IF3170 Intelegensi Artifisial

Tugas Besar 1 IF3170 Inteligensi Artifisial

Pencarian Solusi Diagonal Magic Cube dengan Local Search

Oleh:

Kelompok 13

Muhammad Atpur Rafif	**13522086**
Muhamad Rafli Rasyiidin	**13522088**
M. Hanief Fatkhan Nashrullah	**13522100**
Indraswara Galih Jayanegara	**13522119**

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

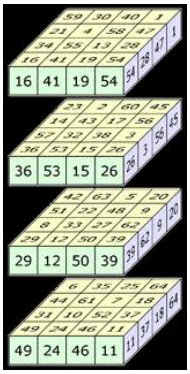
2024

Daftar Isi

Deskripsi Persoalan

Magic Cube merupakan sebuah kubus yang tersusun dari angka 1 hingga n^3 tanpa ada pengulangan dengan n adalah panjang sisi pada kubus tersebut. Angka-angka pada kubus tersusun sedemikian rupa sehingga properti-properti berikut terpenuhi:

- Terdapat satu angka yang merupakan magic number dari kubus tersebut (Magic number tidak harus termasuk dalam rentang 1 hingga n^3, magic number juga bukan termasuk ke dalam angka yang harus dimasukkan ke dalam kubus)
- Jumlah angka-angka untuk setiap baris sama dengan magic number
- Jumlah angka-angka untuk setiap kolom sama dengan magic number
- Jumlah angka-angka untuk setiap tiang sama dengan magic number
- Jumlah angka-angka untuk seluruh diagonal ruang pada kubus sama dengan magic number
- Jumlah angka-angka untuk seluruh diagonal pada suatu potongan bidang dari kubus sama dengan magic number



Most magic 4x4x4 cube of Walter Trump

Sumber : https://www.magischvierkant.com/three-dimensional-eng/magic-features/

Pada tugas ini, akan diselesaikan permasalahan Diagonal Magic Cube berukuran 5x5x5. Initial state dari suatu kubus adalah susunan angka 1 hingga 5^3 secara acak. Kemudian, tiap iterasi pada algoritma local search, langkah yang boleh dilakukan adalah menukar posisi dari 2 angka pada kubus tersebut.

Pembahasan

Objective Function

Terdapat dua fungsi objektif yang kami miliki, fungsi objektif pertama adalah fungsi objektif “analog” yang memperhitungkan jumlah perbedaan antara setiap tiang atau diagonal dengan magic number. Berikut adalah fungsi objektif analog

$$
\begin{aligned}
f(S,n) = {} & \left(\sum_{x=1}^{n}\sum_{y=1}^{n}\left|\left(\sum_{z=1}^{n} S(x,y,z)\right) - g(n)\right|\right) + \left(\sum_{x=1}^{n}\sum_{z=1}^{n}\left|\left(\sum_{y=1}^{n} S(x,y,z)\right) - g(n)\right|\right) \\
& + \left(\sum_{y=1}^{n}\sum_{z=1}^{n}\left|\left(\sum_{x=1}^{n} S(x,y,z)\right) - g(n)\right|\right) \\
& + \left(\sum_{x=1}^{n}\left|\left(\sum_{i=1}^{n} S(x,i,i)\right) - g(n)\right|\right) + \left(\sum_{x=1}^{n}\left|\left(\sum_{i=1}^{n} S(x,i,n-i)\right) - g(n)\right|\right) \\
& + \left(\sum_{y=1}^{n}\left|\left(\sum_{i=1}^{n} S(i,y,i)\right) - g(n)\right|\right) + \left(\sum_{y=1}^{n}\left|\left(\sum_{i=1}^{n} S(i,y,n-i)\right) - g(n)\right|\right) \\
& + \left(\sum_{z=1}^{n}\left|\left(\sum_{i=1}^{n} S(i,i,z)\right) - g(n)\right|\right) + \left(\sum_{z=1}^{n}\left|\left(\sum_{i=1}^{n} S(i,n-i,z)\right) - g(n)\right|\right) \\
& + \left|\left(\sum_{i=1}^{n} S(i,i,i)\right) - g(n)\right| + \left|\left(\sum_{i=1}^{n} S(i,i,n-i)\right) - g(n)\right| \\
& + \left|\left(\sum_{i=1}^{n} S(i,n-i,i)\right) - g(n)\right| + \left|\left(\sum_{i=1}^{n} S(i,n-i,n-i)\right) - g(n)\right|
\end{aligned}
$$

dengan S adalah kubus, $S(x,y,z)$ adalah nilai elemen dari kubus sesuai dengan posisi $x, y,$ dan z pada kubus dan $g(n)$ adalah *magic number*. Nilai n yang digunakan pada kubus berukuran $5 \times 5 \times 5$ adalah 5.

Penentuan *magic number* dilakukan dengan rumus sebagai berikut,

$$g(n) = \frac{n(n^3+1)}{2}$$

dengan n adalah panjang sisi dari kubus.

Karena *magic number* untuk kubus dengan panjang sisi n ditentukan oleh $g(n)$, dan n yang digunakan bernilai 5, maka target dari pencarian ini adalah suatu angka yang mendekati $g(5) = 315$. Untuk dapat mencari hal tersebut, kami menggunakan jumlah dari selisih setiap properti dengan *magic number* $g(5)$. Ketika setiap jumlah komponen memiliki nilai yang sama dengan *magic number*, maka selisihnya akan bernilai 0 dan *state* tersebut adalah *goal state* dari permasalahan *Diagonal Magic Cube*.

Untuk alternatif *objective function* yang akan kami gunakan pada pengujian persoalan ini, kami membuat objective function lain yang kami beri nama *objective function digital* yang merupakan banyaknya persamaan yang memiliki hasil berjumlah *magic number* dari sebuah *state*. Bentuk *state* disini adalah cara pengisian nilai angka pada kubus. Persamaan yang dimaksud disini adalah penjumlahan, baik baris, kolom, tiang, maupun diagonal bidang dan ruang. Fungsi ini akan menghasilkan nilai maksimal apabila susunan angka pada kubus telah berhasil menjadi *magic cube*, atau dengan kata lain seluruh jumlah persamaannya adalah *magic number*. Pencarian akan memaksimalkan fungsi ini untuk mendapatkan solusi. Sedangkan alasan dari pemilihan fungsi ini karena posisi nilai maksimal merupakan *magic cube* dan merupakan hal yang *straightforward* terhadap persoalan yang diberikan.

Kedua fungsi tersebut merupakan alternatif fungsi yang kami pilih sebagai objective function. Fungsi tersebut kami pilih karena pada dasarnya, setiap kubus yang memiliki panjang sisi sebesar n, akan memiliki *pre-determined magic number*. Setiap kubus yang memiliki panjang sisi sebesar n memiliki *magic number* yang bersesuaian dengan panjang sisi tersebut.

Implementasi Algoritma Local Search

Abstraksi

State

Tipe data yang mengimplementasi *constraint* ini. Dapat menghasilkan nilai *objective function*.

Genetic

Tipe data yang mengimplementasi *constraint* ini. Dapat menggabungkan dua state untuk mendapatkan state baru.

iterateIO

Abstraksi dari iterasi. Fungsi ini membutuhkan state sebagai state awal lalu menjalankan fungsi menggunakan state tersebut. Saat fungsi tersebut menghasilkan state baru, maka fungsi tersebut akan dijalankan menggunakan state baru tersebut. Namun, jika tidak menghasilkan apapun, maka fungsi ini akan mengembalikan state terakhir.

Hill-climbing

Source Code

`run :: (State s) => Algorithm Parameter () s`
`run a _ = HCWS.run na (HCWS.Parameter 1)`
`  where`
`  na (s, _) = a (s, ())`

Deskripsi
Fungsi ini merupakan kasus spesifik dari Hill-climbing with sideways move, dengan pembatasan perpindahan state sebanyak satu kali.

Hill-climbing with Sideways Move

Source Code
`run :: (State s) => Algorithm Parameter () s`

```haskell
run a p s = snd <$> iterateIO (maxIteration p, s) f
  where f (0, _) = return Nothing
        f (ci, cs) = do
              a (cs, ())
              ns <- pickRandom $ neighbor cs
              case getPoint ns `compare` getPoint cs of
                  LT -> return Nothing
                  EQ -> return $ Just (ci - 1, ns)
                  GT -> return $ Just (maxIteration p, ns)
```

Deskripsi

Fungsi ini mengambil neighbor pada setiap iterasinya. Terdapat tiga kasus yaitu ketika *objective function* dari *next state* kurang dari *current state* maka algoritma akan berhenti. Kemudian saat *objective function* bernilai sama algoritma mengurangi iterasi yang diperbolehkan dan lanjut menggunakan state baru. Terakhir, saat *objective function* lebih besar dibandingkan dengan *current state*, *state* akan berpindah ke *state* baru dan iterasi yang diperbolehkan kembali ke nilai awal. Selain itu, saat jumlah iterasi yang diperbolehkan sudah nol, maka algoritma akan berhenti.

Random Restart Hill-climbing

Source Code

```haskell
run :: forall s. (State s) => Algorithm Parameter Data s
run a p s = snd <$> iterateIO (maxRestart p, s) f
  where
       f (0, _) = return Nothing
       f (cr, cs) = do
       let na :: (s, ()) -> IO ()
            na (is, _) = a (is, Data {restartCount = maxRestart p - cr})
       rs <- randomState cs
       ns <- HC.run na () rs
       if isGlobalMaxima cs
            then return Nothing
            else return $ Just (cr - 1, if getPoint ns > getPoint cs then ns else
cs)
```

Deskripsi

Fungsi ini menggunakan hill-climbing dengan maksimal iterasi sebanyak parameter yang diberikan. Pada setiap iterasinya akan dibuat *state* baru secara *random*.

Stochastic Hill-climbing

Source Code

```
run :: (State s) => Algorithm Parameter () s
run a p s = snd <$> iterateIO (maxIteration p, s) f
  where f (0, _) = return Nothing
        f (ci, cs) = do
             a (cs, ())
             ns <- nextRandomState cs
             return $ Just (ci - 1, if getPoint ns > getPoint cs then ns else cs)
```

Deskripsi

Fungsi ini akan membentuk *state* baru secara *random* sebanyak parameter yang diberikan. State akan berpindah ketika *objective function state* baru bernilai lebih baik.

Simulated Annealing

Source Code

```
run :: (State s) => Algorithm Parameter Data s
run a p s = snd <$> iterateIO (sd, s) f
  where
      sd =
           Data
                { temperature = initialTemperature p,
                      probabilityThreshold = 0,
                      stuckCount = 0
      }
      f (Data 0 _ _, _) = return Nothing
      f (d, cs) = do
      ns <- nextRandomState cs
      let de = getPoint ns - getPoint cs
            ct = temperature d
            pv = exp $ (fromIntegral de :: Double) / ct
            nt = g ct
            nd = d {probabilityThreshold = pv, temperature = nt}
      a (cs, nd)
```

```
      if de > 0
      then return $ Just (nd, ns)
      else do
            r <- randomRIO (0, 1) :: IO Double
            let nnd = nd {stuckCount = stuckCount nd + 1}
            if r < pv
            then return $ Just (nnd, ns)
            else return $ Just (nnd, cs)

      g :: Double -> Double
      g t = case function p of
      Exponential d -> if t < 1e-10 then 0 else t / d
      Linear m -> t - m
```

Deskripsi

Fungsi ini akan menerima temperatur awal dan melakukan set stuckCount ke nol. Jika temperatur mencapai nol, maka algoritma berhenti. Kemudian akan dibuat suatu *state* baru dan dihitung selisih *objective function* dari *current state* dengan *state* yang baru dibentuk. Jika *state* baru lebih baik, maka *state baru* tersebut akan dipilih. Jika tidak, maka stuckCount akan ditambah satu kemudian diperiksa probabilitasnya apakah *state* baru tersebut akan dipilih atau tidak.

Genetic Algorithm

Source Code

```
run :: forall s. (State s, Genetic s) => Algorithm Parameter Data s
run a p s = do
  pp <- replicateM (populationSize p) $ randomState s
  let f :: (Int, [s]) -> IO (Maybe (Int, [s]))
        f (0, _) = return Nothing
        f (i, ss) = do
        let ssw = if all ((0 ==) . snd) l then fe <$> l else l
                where
                l = ft <$> ss
                ft v = (v, toRational $ getPoint v)
                fe (v, _) = (v, 1)
        let rn = R.fromList ssw
        avg = realToFrac (sum $ getPoint <$> ss) / realToFrac (populationSize p) :: Double
        ms = foldr1 g ss
                where g s1 s2 = if getPoint s1 > getPoint s2 then s1 else s2
        a (ms, Data {pointAverage = avg})
        nss <- forM ss $ \_ -> do
        p1 <- rn
```

```
        p2 <- rn
        m <- randomIO :: IO Bool
        c <- combineGenes p1 p2
        if m
        then nextRandomState c
        else return c
        return $ Just (i - 1, nss)
  ss <- iterateIO (maxIteration p, pp) f
  return $ (foldr1 h . snd) ss
  where
        h c r = if getPoint c > getPoint r then c else r
```

Deskripsi

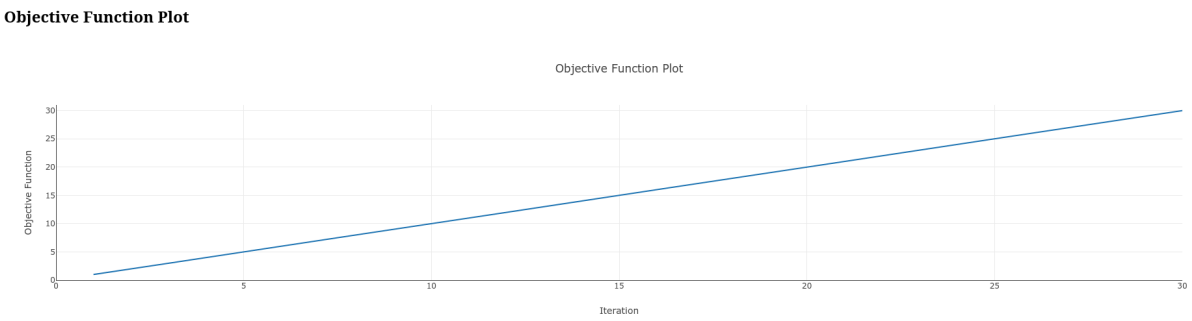
Fungsi ini melakukan generasi populasi awal secara random. Pada setiap iterasi, fitness function diperiksa, kemudian dipilih kandidat terbaiknya untuk menghasilkan generasi baru. Langkah ini diulang sampai batas maksimal iterasi tercapai atau tujuan tercapai. Terakhir, dikembalikan solusi terbaik.

Hasil Eksperimen dan Analisis

Hill-climbing (Steepest-ascent)

Percobaan ke-1

Plot



State Awal

64	103	53	15	7
67	78	25	30	18
51	82	115	122	27
95	45	110	66	23
65	93	123	4	36

113	8	2	100	92
98	48	44	107	112
17	11	72	97	84
85	41	87	96	59
46	80	69	99	37

33	106	109	119	24
71	28	120	79	39
34	88	9	29	81
90	32	12	70	125
49	55	91	38	40

43	54	116	101	68
62	14	94	5	73
75	105	3	47	77
22	19	118	52	111
1	83	56	76	26

57	13	16	121	86
114	58	50	104	42
74	89	102	21	35
60	10	6	61	63
31	108	124	117	20

State Akhir

79	52	96	15	7
67	78	25	20	125
17	93	63	115	27
95	10	69	66	23
91	82	123	4	36

113	8	2	100	92
1	48	44	107	18
51	11	72	97	84
85	41	110	73	59
49	55	87	80	37

16	68	109	26	24
71	28	120	57	39
83	88	9	35	81
90	32	12	70	112
46	99	65	38	40

43	54	116	101	106
62	103	77	5	53
75	34	3	47	94
22	19	118	45	111
98	105	56	117	121

64	13	33	119	86
114	58	50	104	42
74	89	102	21	29
60	14	6	61	122
31	108	124	76	30

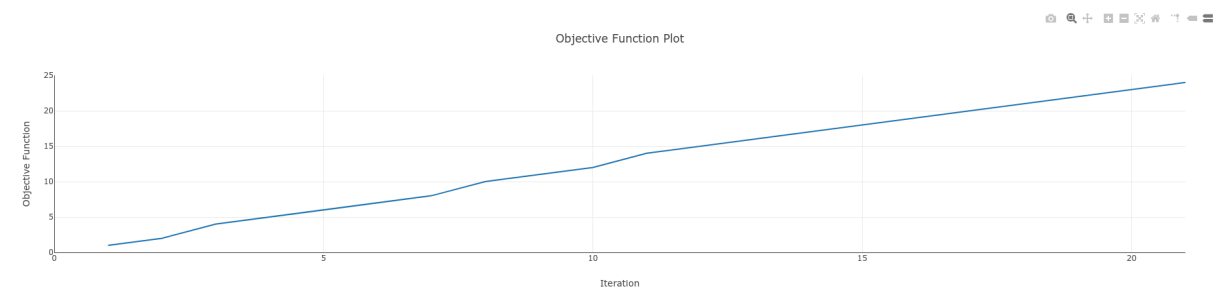
Final objective func value: 31

Durasi: 474.50009600000004 ms

Banyak Iterasi: 30

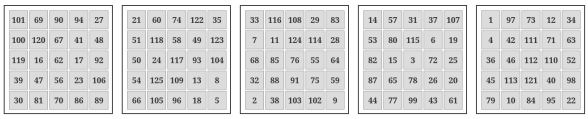
Percobaan ke-2

Plot



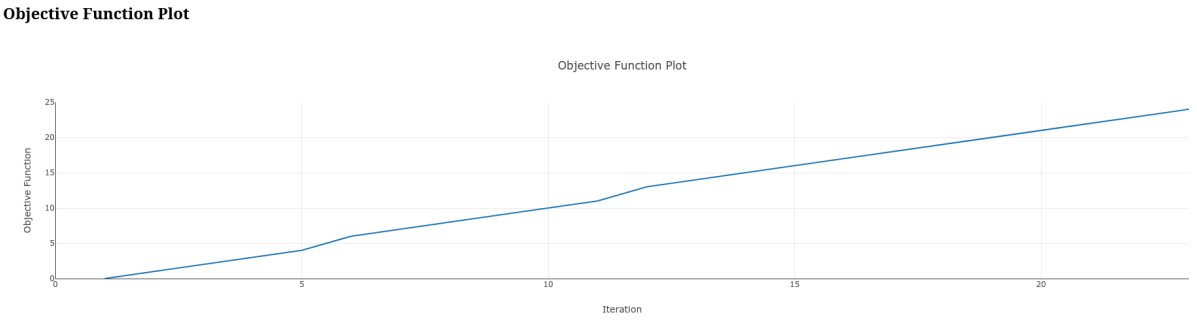
State Awal

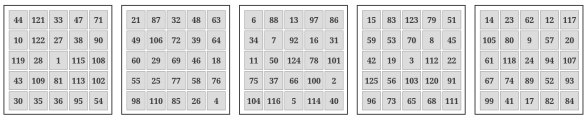
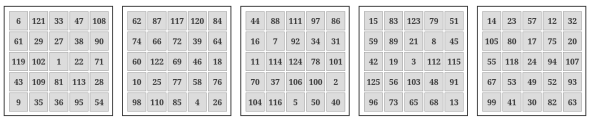
State Akhir

	
Final objective func value: 25	
Durasi: 348.06185700000003 ms	Banyak Iterasi: 21

Percobaan ke-3

Plot



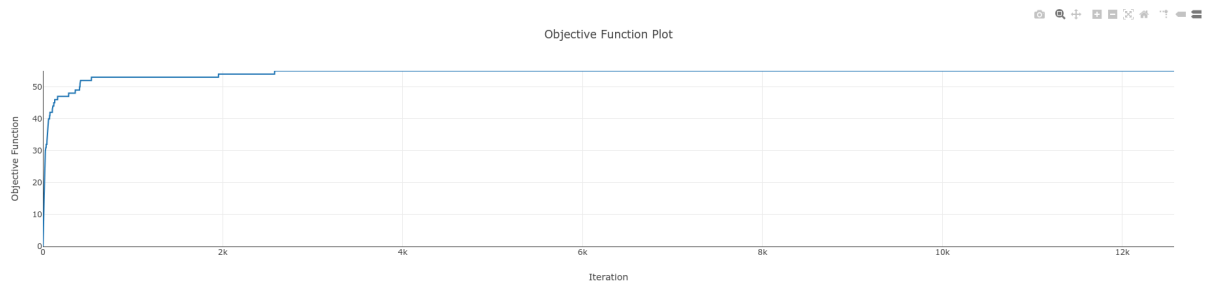
State Awal 	State Akhir 
Final objective func value:25	
Durasi: 381.671038 ms	Banyak Iterasi: 23

Hill-climbing with Sideways Move

Percobaan ke-1

Plot

Objective Function Plot



State Awal

82 30 53 14 71 41 52 36 8 100 77 61 27 106 33 70 28 64 56 58 104 102 44 59 88	43 123 94 19 35 47 108 37 26 16 49 119 57 83 50 24 69 81 112 84 3 62 2 60 111	34 38 21 113 66 120 40 63 31 54 4 18 99 11 93 23 116 98 92 67 96 101 29 65 78	75 90 42 46 12 95 122 6 17 107 73 115 109 125 7 39 91 20 25 97 32 5 103 68 114	124 80 76 117 121 105 48 10 74 79 45 15 22 13 89 55 51 110 85 1 9 87 118 72 86

State Akhir

34 114 93 3 71 105 28 66 108 8 33 61 26 89 106 110 6 52 56 91 104 15 88 59 27	47 49 83 19 58 41 51 37 99 87 1 119 112 45 38 24 69 81 57 84 5 102 2 63 48	100 97 21 86 11 120 32 96 31 36 43 82 4 65 67 92 70 98 94 111 22 101 68 39 85	10 80 76 90 54 40 103 72 17 16 116 35 30 109 25 75 55 64 23 78 74 62 73 29 77	124 46 42 117 121 9 123 44 60 79 122 18 118 7 50 14 115 20 113 53 95 13 107 125 12

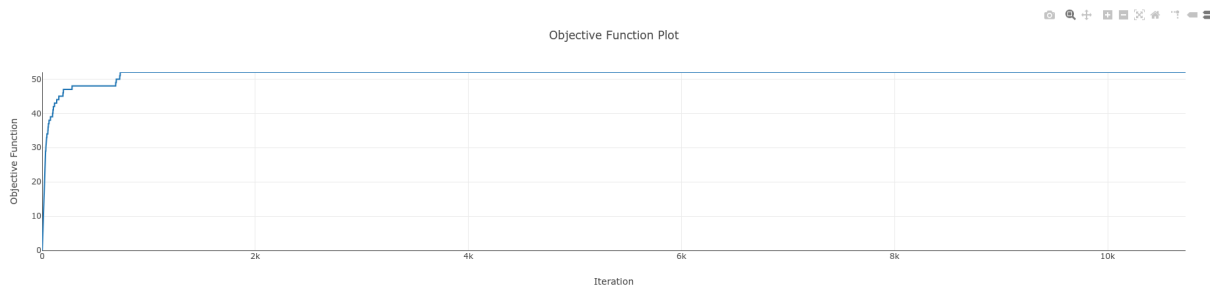
Final objective func value:55

Durasi: 203669.94681700002 ms	Max sideways move: 10000 Banyak Iterasi: 12574

Percobaan ke-2

Plot

Objective Function Plot



State Awal

120 111 65 72 52 35 33 3 54 82 18 61 50 59 17 97 10 106 66 58 64 119 55 4 108	25 49 122 74 44 20 21 68 56 73 107 85 13 110 19 46 94 14 41 69 77 57 9 123 11	67 78 100 125 60 116 27 40 89 83 90 112 88 36 80 12 99 118 117 102 101 79 43 34 104	23 37 28 121 92 87 86 53 95 91 16 76 2 103 109 48 32 1 15 29 7 5 63 6 93	8 84 30 47 31 22 38 105 39 114 96 62 113 24 75 71 70 42 115 51 45 124 81 98 26

State Akhir

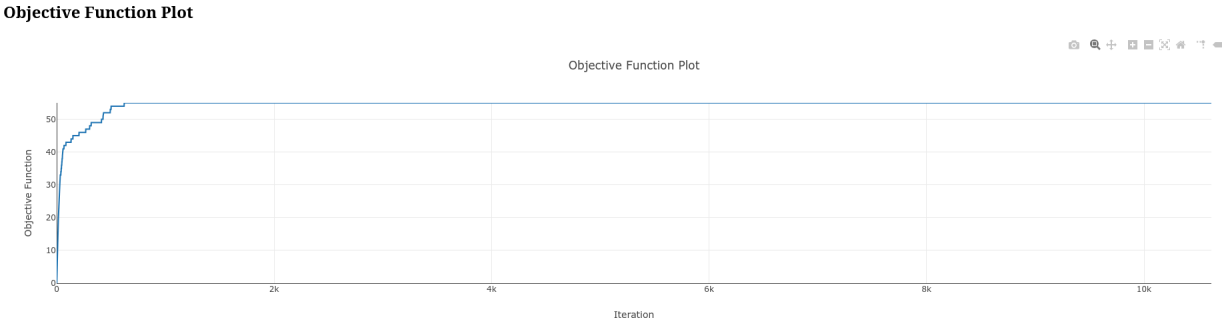
120 3 103 24 65 57 18 68 56 41 64 88 59 85 19 58 115 50 10 82 16 51 35 9 108	77 116 4 74 44 12 106 102 39 20 107 112 98 93 109 46 61 71 110 27 73 38 13 125 66	67 76 111 49 86 1 79 52 89 94 90 33 43 54 80 91 7 81 22 114 84 34 78 101 104	23 37 42 121 92 30 87 53 95 69 117 8 2 21 32 48 29 99 15 124 97 70 119 63 11	28 83 55 47 31 96 25 40 36 118 5 60 113 62 75 72 6 14 123 100 45 122 105 17 26

Final objective func value:52

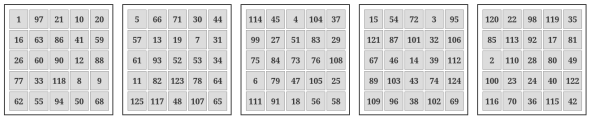
Durasi: 171688.318963 ms	Max sideways move:10000 Banyak Iterasi: 10731

Percobaan ke-3

Plot



State Awal



State Akhir

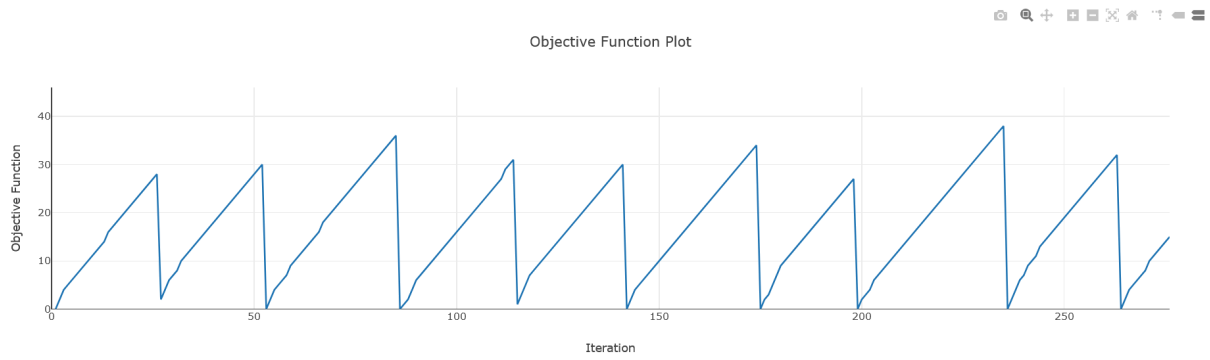
Final objective func value:55

Durasi: 179870.101929 ms	Max sideways move: 10000 Banyak Iterasi: 10619

Random Restart Hill-climbing

Percobaan ke-1

Plot



State Awal

79 84 100 102 14 103 59 46 12 18 73 83 85 6 34 1 108 4 8 105 117 113 115 25 90	60 24 35 38 65 125 122 92 89 112 86 97 9 40 123 72 21 63 61 13 87 27 26 23 106	101 10 20 124 19 2 11 28 69 53 44 56 95 78 50 119 62 58 111 91 118 110 31 39 82	55 109 33 17 54 116 114 5 49 52 88 94 120 22 75 16 99 45 93 32 3 96 76 98 30	121 51 74 7 15 81 71 66 67 43 68 36 48 80 57 70 104 77 47 37 64 41 42 107 29

State Akhir

57 114 125 71 10 63 21 103 28 100 76 9 48 101 51 46 37 24 36 27 7 35 56 79 97	116 93 47 94 89 81 53 25 6 23 2 59 120 95 39 98 96 64 13 44 4 14 88 107 102	50 84 119 118 117 85 109 74 83 54 121 90 110 115 60 104 75 38 30 68 32 62 123 82 16	22 52 5 15 3 26 55 66 91 77 105 8 49 111 42 108 78 43 65 106 72 122 1 33 87	40 112 19 17 31 80 73 41 29 34 11 20 61 99 124 113 18 45 69 70 67 92 12 58 86

Final objective func value: 31

Durasi: 37401.774690000006 ms

Banyak Iterasi: 10

Restart Iterasi 1: 0 iterasi

Restart Iterasi 2: 25 iterasi

Restart Iterasi 3: 25 iterasi

Restart Iterasi 4: 32 iterasi

Restart Iterasi 5: 28 iterasi

Restart Iterasi 6: 26 iterasi

Restart Iterasi 7: 32 iterasi

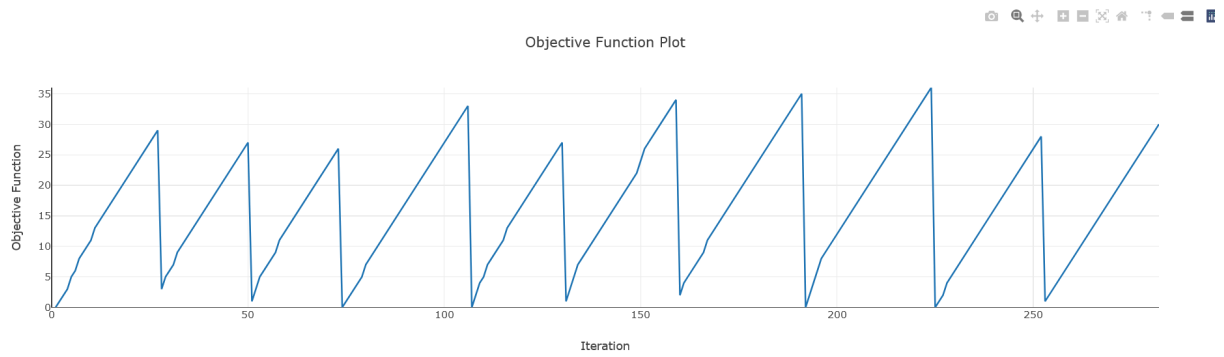
Restart Iterasi 8: 23 iterasi

Restart Iterasi 9: 36 iterasi

Restart Iterasi 10: 27 iterasi

Percobaan ke-2

Plot



State Awal

State Akhir

Final objective func value: 31

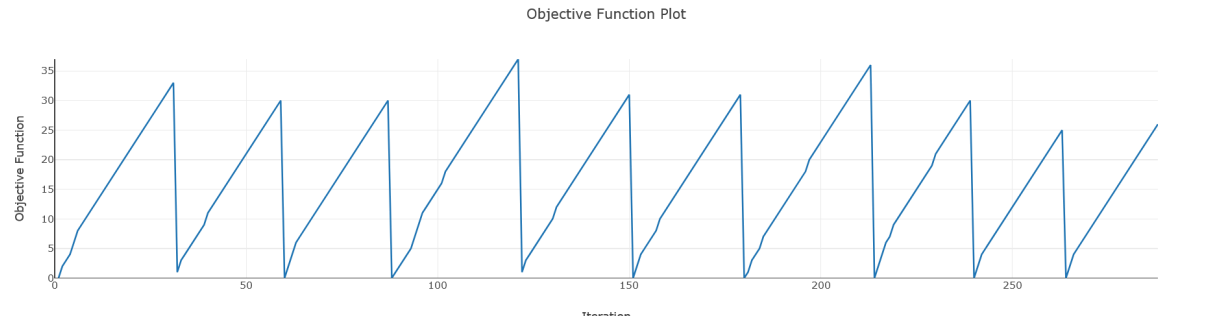
Durasi: 3788.7569420000004 ms

Banyak Iterasi:

- Restart Iterasi 1: 0 iterasi
- Restart Iterasi 2: 26 iterasi
- Restart Iterasi 3: 22 iterasi
- Restart Iterasi 4: 22 iterasi
- Restart Iterasi 5: 32 iterasi
- Restart Iterasi 6: 23 iterasi
- Restart Iterasi 7: 28 iterasi
- Restart Iterasi 8: 31 iterasi
- Restart Iterasi 9: 32 iterasi
- Restart Iterasi 10: 27 iterasi

Percobaan ke-3

Plot



State Awal

State Akhir

<table>
<tr>
<td>

<table>
<tr><td>1</td><td>115</td><td>109</td><td>40</td><td>30</td></tr>
<tr><td>75</td><td>55</td><td>21</td><td>82</td><td>17</td></tr>
<tr><td>65</td><td>22</td><td>121</td><td>123</td><td>11</td></tr>
<tr><td>31</td><td>93</td><td>20</td><td>89</td><td>100</td></tr>
<tr><td>34</td><td>16</td><td>119</td><td>91</td><td>81</td></tr>
</table>
<table>
<tr><td>24</td><td>2</td><td>70</td><td>57</td><td>15</td></tr>
<tr><td>43</td><td>66</td><td>79</td><td>69</td><td>102</td></tr>
<tr><td>49</td><td>33</td><td>27</td><td>97</td><td>62</td></tr>
<tr><td>25</td><td>37</td><td>14</td><td>61</td><td>78</td></tr>
<tr><td>9</td><td>4</td><td>26</td><td>8</td><td>50</td></tr>
</table>
<table>
<tr><td>113</td><td>106</td><td>87</td><td>104</td><td>23</td></tr>
<tr><td>42</td><td>39</td><td>105</td><td>110</td><td>90</td></tr>
<tr><td>5</td><td>72</td><td>13</td><td>71</td><td>44</td></tr>
<tr><td>10</td><td>41</td><td>19</td><td>46</td><td>51</td></tr>
<tr><td>56</td><td>77</td><td>76</td><td>29</td><td>99</td></tr>
</table>
<table>
<tr><td>35</td><td>52</td><td>107</td><td>120</td><td>84</td></tr>
<tr><td>80</td><td>74</td><td>48</td><td>59</td><td>122</td></tr>
<tr><td>114</td><td>47</td><td>94</td><td>117</td><td>53</td></tr>
<tr><td>101</td><td>38</td><td>67</td><td>108</td><td>28</td></tr>
<tr><td>96</td><td>103</td><td>7</td><td>116</td><td>95</td></tr>
</table>
<table>
<tr><td>73</td><td>58</td><td>12</td><td>85</td><td>118</td></tr>
<tr><td>68</td><td>36</td><td>60</td><td>124</td><td>64</td></tr>
<tr><td>98</td><td>88</td><td>18</td><td>86</td><td>112</td></tr>
<tr><td>83</td><td>63</td><td>92</td><td>54</td><td>3</td></tr>
<tr><td>6</td><td>45</td><td>32</td><td>111</td><td>125</td></tr>
</table>
</td>
<td>

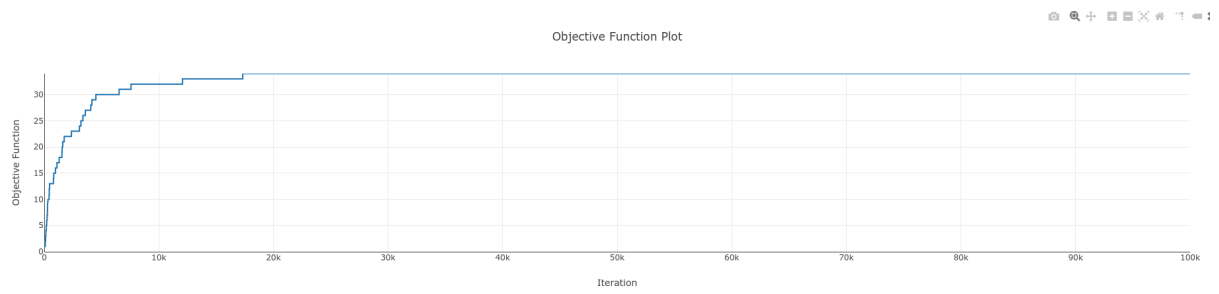
<table>
<tr><td>94</td><td>82</td><td>58</td><td>31</td><td>50</td></tr>
<tr><td>32</td><td>57</td><td>103</td><td>63</td><td>60</td></tr>
<tr><td>66</td><td>108</td><td>29</td><td>109</td><td>38</td></tr>
<tr><td>11</td><td>45</td><td>86</td><td>43</td><td>78</td></tr>
<tr><td>112</td><td>23</td><td>39</td><td>89</td><td>92</td></tr>
</table>
<table>
<tr><td>42</td><td>90</td><td>41</td><td>27</td><td>115</td></tr>
<tr><td>73</td><td>12</td><td>84</td><td>98</td><td>48</td></tr>
<tr><td>87</td><td>96</td><td>6</td><td>119</td><td>72</td></tr>
<tr><td>107</td><td>105</td><td>104</td><td>69</td><td>81</td></tr>
<tr><td>44</td><td>122</td><td>80</td><td>2</td><td>125</td></tr>
</table>
<table>
<tr><td>74</td><td>36</td><td>111</td><td>3</td><td>91</td></tr>
<tr><td>100</td><td>121</td><td>52</td><td>30</td><td>116</td></tr>
<tr><td>14</td><td>20</td><td>123</td><td>93</td><td>65</td></tr>
<tr><td>49</td><td>47</td><td>37</td><td>114</td><td>25</td></tr>
<tr><td>83</td><td>54</td><td>85</td><td>75</td><td>18</td></tr>
</table>
<table>
<tr><td>17</td><td>102</td><td>19</td><td>106</td><td>101</td></tr>
<tr><td>15</td><td>10</td><td>40</td><td>120</td><td>4</td></tr>
<tr><td>46</td><td>21</td><td>113</td><td>59</td><td>76</td></tr>
<tr><td>24</td><td>7</td><td>33</td><td>22</td><td>118</td></tr>
<tr><td>97</td><td>61</td><td>110</td><td>8</td><td>16</td></tr>
</table>
<table>
<tr><td>88</td><td>5</td><td>9</td><td>35</td><td>79</td></tr>
<tr><td>95</td><td>34</td><td>68</td><td>77</td><td>99</td></tr>
<tr><td>51</td><td>70</td><td>62</td><td>28</td><td>53</td></tr>
<tr><td>124</td><td>56</td><td>55</td><td>67</td><td>13</td></tr>
<tr><td>26</td><td>117</td><td>1</td><td>71</td><td>64</td></tr>
</table>
</td>
</tr>
<tr>
<td colspan="2">Final objective func value: 27</td>
</tr>
<tr>
<td>Durasi: 3782.6614940000004 ms</td>
<td>Banyak Iterasi:
- Restart Iterasi 1: 0 iterasi
- Restart Iterasi 2: 30 iterasi
- Restart Iterasi 3: 27 iterasi
- Restart Iterasi 4: 27 iterasi
- Restart Iterasi 5: 33 iterasi
- Restart Iterasi 6: 28 iterasi
- Restart Iterasi 7: 28 iterasi
- Restart Iterasi 8: 33 iterasi
- Restart Iterasi 9: 25 iterasi
- Restart Iterasi 10: 23 iterasi</td>
</tr>
</table>

Stochastic Hill-climbing

Percobaan ke-1

Plot

Objective Function Plot



State Awal

49 114 71 66 120 69 12 51 108 55 10 2 45 100 79 117 107 86 44 85 113 47 58 42 20	63 33 60 16 74 122 6 116 93 46 81 11 59 56 9 29 110 1 123 30 121 7 36 19 78	35 23 65 8 119 48 50 125 62 75 94 112 18 32 72 57 27 82 28 67 22 40 76 24 70	104 99 115 105 52 97 53 4 14 17 34 102 15 84 89 80 95 64 31 26 5 109 96 21 118	3 101 61 68 77 124 43 90 98 39 111 25 92 106 38 83 91 88 37 73 54 41 13 87 103

State Akhir

13 114 15 66 108 69 12 111 68 55 10 65 71 91 78 19 107 60 44 85 113 118 58 9 17	63 33 86 16 74 122 6 116 93 46 94 11 59 56 95 29 110 1 123 52 121 7 36 117 34	35 23 89 8 119 48 125 50 62 30 81 112 18 32 72 57 27 82 67 28 22 40 76 20 70	104 99 64 105 75 92 53 4 14 97 79 102 80 115 2 45 42 84 31 90 5 109 96 101 47	3 106 61 120 77 124 43 26 98 24 51 25 87 21 38 83 100 88 37 73 54 41 49 39 103

Final objective func value:34

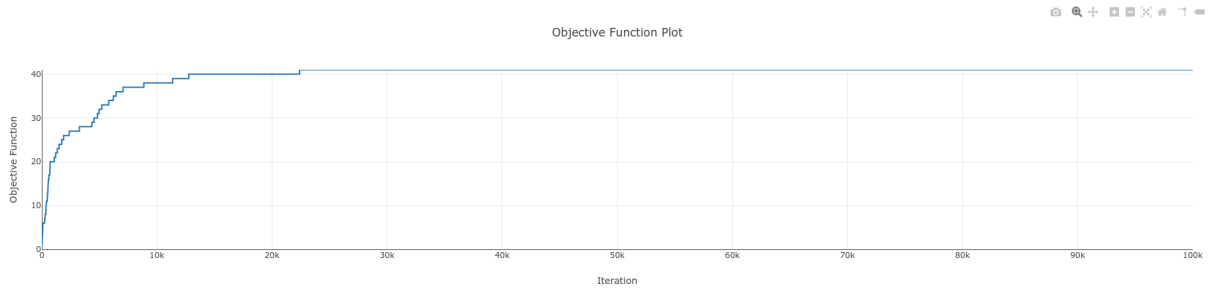
Durasi: 1424.5817390000002 ms

Banyak Iterasi: 100000

Percobaan ke-2

Plot

Objective Function Plot



State Awal

58 49 68 113 16 36 118 92 1 65 115 99 52 89 48 77 20 13 104 45 73 43 94 23 11	102 50 71 111 62 87 12 67 14 7 26 119 66 107 40 30 21 122 29 110 18 96 8 98 79	27 100 6 41 105 70 123 120 10 46 19 81 15 25 3 82 75 35 109 33 85 31 97 83 9	34 108 116 28 42 76 93 61 106 57 69 47 64 103 44 5 59 90 63 121 101 80 117 53 56	51 24 74 60 112 54 39 124 114 55 88 22 32 2 37 78 84 72 95 86 17 125 4 91 38

State Akhir

58 49 16 124 68 36 12 6 1 65 71 118 52 7 25 77 22 13 110 93 73 114 94 23 11	102 50 103 27 62 87 39 67 33 89 26 76 66 107 40 30 21 105 79 56 125 96 8 57 29	70 100 4 5 122 59 123 120 10 3 19 54 101 108 46 82 75 35 109 14 85 31 55 83 61	34 92 116 28 45 86 42 9 80 98 69 47 64 91 44 48 111 90 63 121 15 18 117 53 112	51 24 74 60 106 81 99 113 43 97 88 20 32 2 37 78 84 72 95 119 17 104 41 115 38

Final objective func value:41

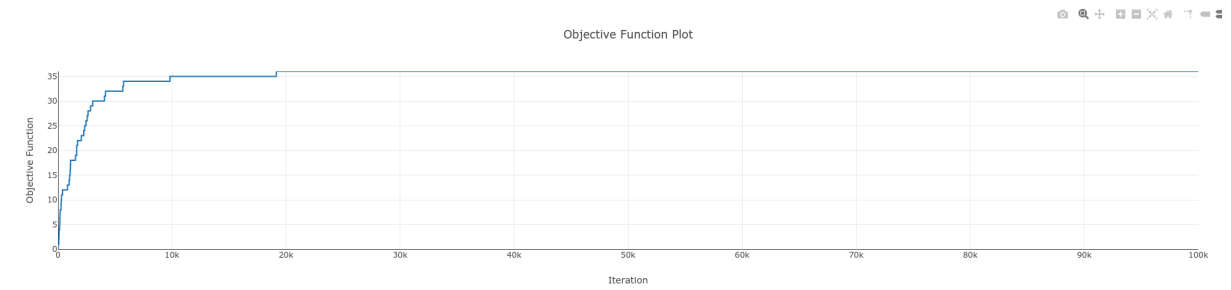
Durasi: 1428.692362 ms

Banyak Iterasi: 100000

Percobaan ke-3

Plot

Objective Function Plot



State Awal

9 44 97 14 43 11 22 19 2 101 80 6 56 12 17 25 53 118 108 99 15 106 103 51 29	5 115 3 33 95 82 119 124 32 74 37 21 34 81 72 71 46 63 41 75 100 49 122 105 10	123 90 18 16 117 69 98 54 107 50 109 83 20 61 96 68 77 111 67 27 57 65 78 70 84	120 73 47 94 110 60 55 36 26 58 62 42 30 4 121 52 86 76 8 91 79 45 39 116 24	31 87 88 64 112 40 1 13 113 93 89 125 38 85 114 92 66 23 28 59 7 48 35 104 102

State Akhir

9 83 97 14 43 52 42 19 2 51 6 80 56 84 89 5 53 118 108 99 15 57 103 107 33	25 115 3 111 95 82 119 93 32 74 37 21 34 81 72 71 11 63 41 87 100 49 122 105 10	123 31 24 16 121 69 98 45 101 59 120 44 20 61 96 109 77 35 67 27 114 65 78 70 12	68 38 47 110 94 60 55 116 26 58 62 102 30 4 117 39 86 76 8 28 79 92 46 36 18	90 48 13 64 112 40 1 88 113 73 124 125 17 85 106 54 66 23 91 50 7 75 29 104 22

Final objective func value:36

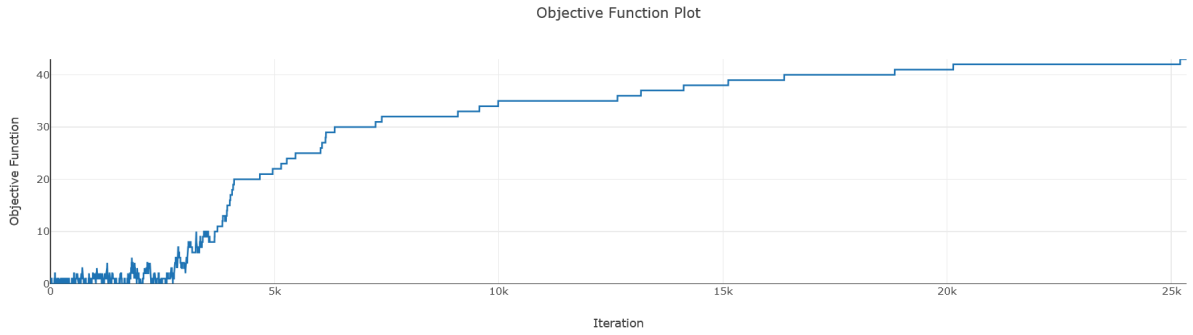
Durasi: 1428.944492 ms

Banyak Iterasi: 100000

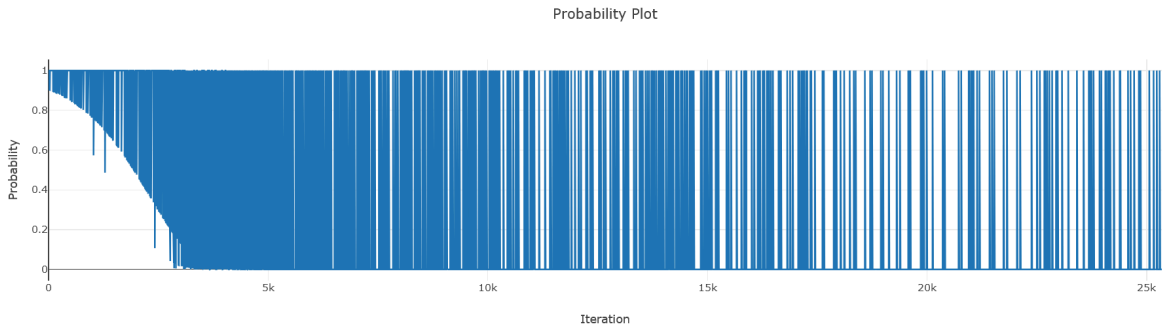
Simulated Annealing

Percobaan ke-1

Plot objective func/iteration



Plot $e^{\frac{\Delta E}{T}}$ /iteration



State Awal

State Akhir

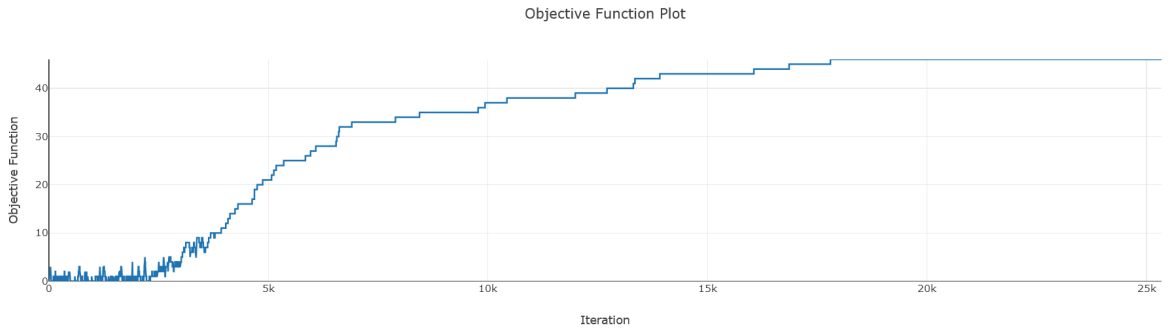
Final objective func value: 43

Frekuensi stuck: 25164

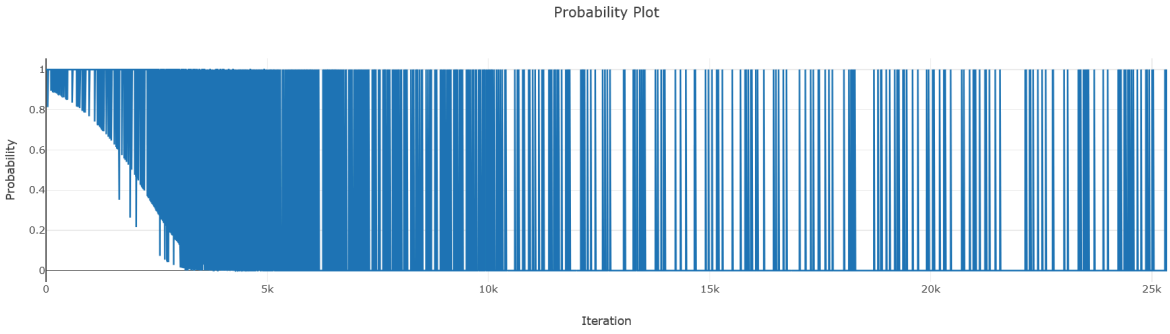
Durasi: 319.34565200000003 ms

Percobaan ke-2

Plot objective func/iteration



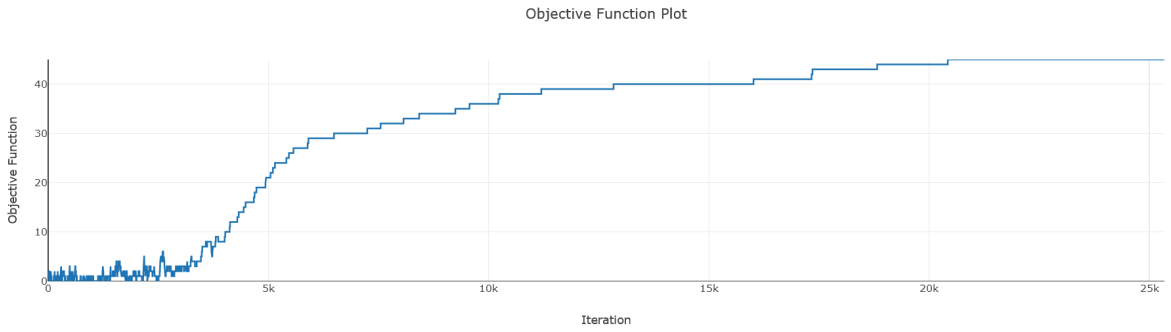
Plot $e^{\frac{\Delta E}{T}}$/iteration



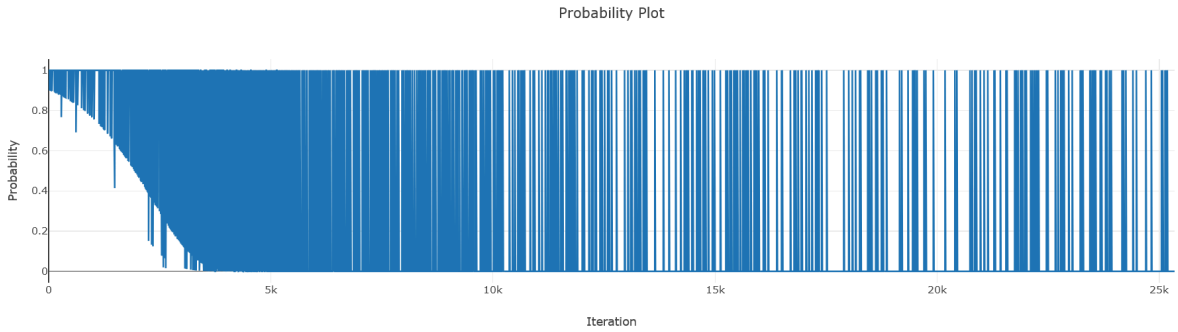
State Awal	State Akhir
	
Final objective func value: 46	Frekuensi stuck: 25185
Durasi: 314.07101800000004 ms	

Percobaan ke-3

Plot objective func/iteration



Plot $e^{\frac{\Delta E}{T}}$ /iteration



State Awal

61 77 97 120 50 93 98 92 107 80 82 64 125 69 104 96 90 19 73 74 60 53 95 37 18	22 49 23 91 84 15 55 20 33 99 16 12 121 54 89 56 43 62 116 13 118 86 102 83 94	112 101 67 2 88 21 17 119 81 52 57 40 42 45 68 44 75 105 87 6 30 59 124 28 4	106 34 32 117 11 103 58 110 72 7 108 46 39 71 109 5 78 51 113 79 25 76 36 8 14	123 1 26 47 111 29 27 41 24 85 114 65 70 3 10 122 115 9 48 35 31 63 38 100 66

State Akhir

35 69 77 73 61 47 116 64 10 78 8 52 113 97 20 15 18 108 28 30 111 7 67 107 23	102 87 21 13 92 62 58 91 38 94 95 70 82 125 105 115 63 46 90 124 40 37 75 49 76	53 93 100 101 89 117 79 45 33 41 31 27 50 123 14 59 114 1 12 66 29 2 119 44 121	71 83 85 19 57 4 98 11 122 80 86 9 36 88 96 42 110 72 26 65 22 68 103 60 17	54 120 32 109 74 3 106 104 112 43 118 16 48 5 99 84 25 24 34 6 56 51 39 55 81

Final objective func value: 45

Frekuensi stuck: 25167

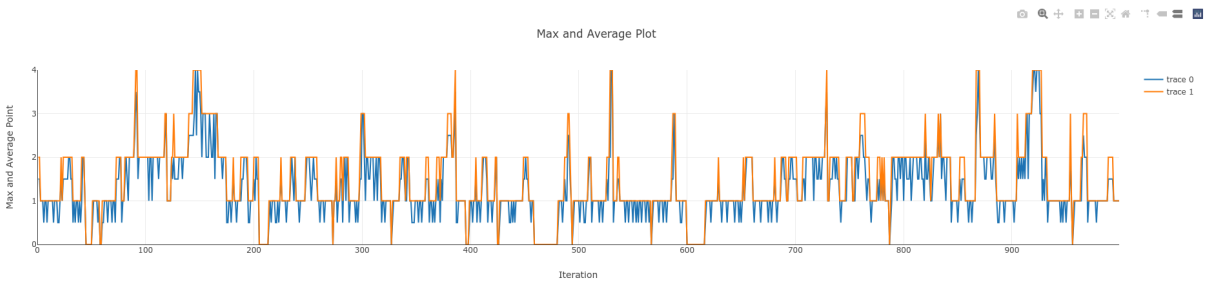
Durasi: 318.05748600000004 ms

Genetic Algorithm

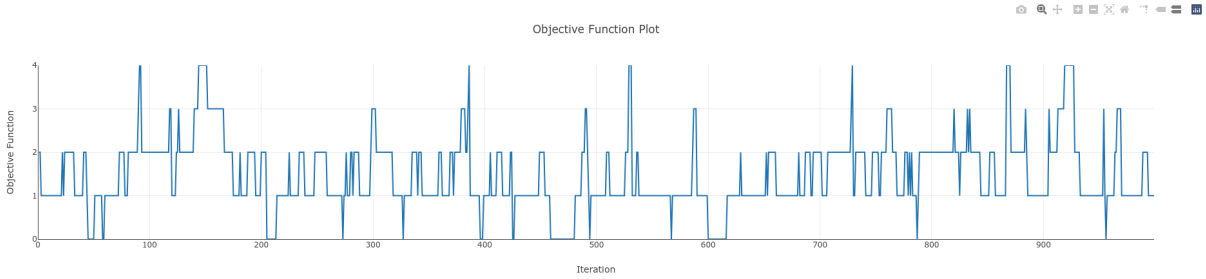
Iteration Control (Population Variable)

Iteration Control (2 population Variable, Test 1)

Plot objective func/iteration



Objective Function Plot



Jumlah Populasi: 2

State Awal

116	99	109	85	117
83	9	50	98	62
66	122	82	84	16
73	49	19	59	94
68	106	23	47	123

6	54	53	3	124
48	69	46	13	2
70	36	89	63	96
79	114	88	108	92
100	7	22	25	125

24	45	104	21	86
38	90	18	76	81
93	41	80	52	27
31	71	26	72	60
107	44	55	51	43

17	33	110	30	102
57	101	10	111	32
5	29	118	113	15
14	58	39	56	112
115	61	4	74	75

12	97	77	67	11
91	42	40	28	87
64	103	20	65	8
120	105	1	121	34
119	37	78	95	35

State Akhir

94	32	90	95	73
109	14	89	59	35
45	13	15	79	102
21	80	16	75	41
42	5	1	49	26

119	124	22	86	115
96	40	72	55	71
6	43	30	4	36
104	101	67	44	107
3	10	123	117	78

121	31	83	34	39
7	74	88	97	18
99	62	125	100	93
81	60	85	84	8
2	27	46	24	38

52	20	37	58	64
70	77	25	61	110
92	120	118	56	48
17	98	122	63	33
57	66	28	103	69

76	47	91	65	111
106	112	29	82	114
19	11	53	12	116
9	108	23	51	113
87	54	68	105	50

Final Objective Function Value: 1	
Durasi: 433.908382 ms	Banyak Iterasi: 1000

Iteration Control (2 population Variable, Test 2)

Plot objective func/iteration

Jumlah Populasi: 2

State Awal

99	67	35	4	112
1	100	48	54	121
15	88	8	39	6
30	17	27	62	73
125	52	24	102	85

9	25	118	95	3
110	40	119	69	86
93	87	94	12	117
28	106	108	60	38
115	72	114	91	65

37	120	98	50	124
36	20	49	32	101
116	66	80	77	74
78	104	18	68	111
22	56	81	7	46

63	89	58	113	23
21	55	47	92	26
64	83	75	45	79
61	84	123	10	29
107	76	105	97	2

44	13	34	31	53
43	33	82	70	11
41	57	16	14	19
5	103	96	51	90
122	42	59	71	109

State Akhir

76	43	46	56	14
96	124	19	97	5
74	12	57	92	89
1	28	23	94	34
69	25	120	27	49

62	77	107	33	99
83	54	91	102	38
101	114	52	58	41
64	2	36	88	45
118	15	63	4	103

30	21	70	121	123
95	26	68	42	39
115	93	35	24	55
98	9	6	3	100
112	82	125	90	78

111	72	106	59	119
65	75	113	71	85
117	67	109	32	20
104	60	80	47	53
17	81	66	108	79

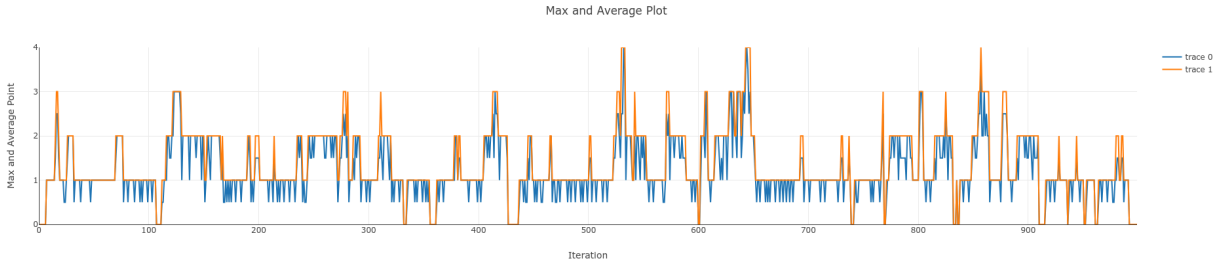
7	73	122	11	86
116	22	61	105	51
110	84	48	40	16
37	10	44	18	8
13	29	31	50	87

Final Objective Function Value: 0

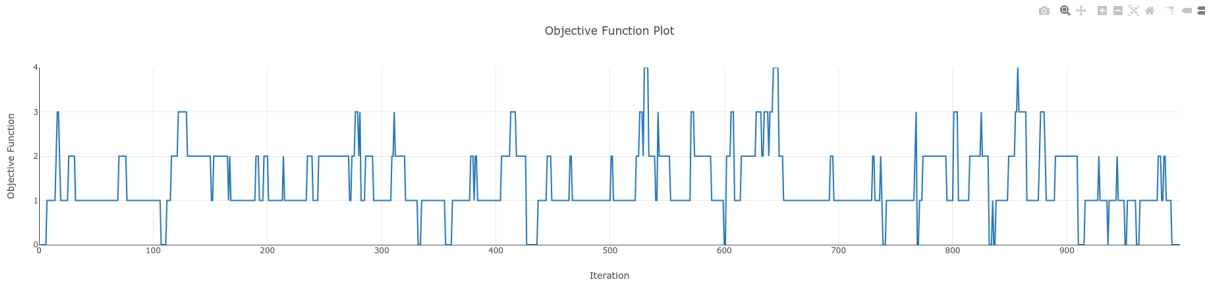
Durasi: 459.728993 ms	Banyak Iterasi: 1000

Iteration Control (2 population Variable, Test 3)

Plot objective func/iteration



Objective Function Plot



Jumlah Populasi: 2

State Awal

112	9	11	93	108
64	29	120	55	48
14	38	47	113	96
20	23	110	15	7
13	107	36	5	71

90	66	118	61	25
67	91	41	6	65
59	8	103	35	111
42	31	95	100	106
10	39	123	81	89

51	117	54	50	37
17	44	74	78	104
82	33	125	22	105
85	115	43	97	73
68	84	77	63	34

26	124	3	12	116
45	76	57	70	18
86	94	87	101	92
102	32	60	72	30
62	52	83	119	98

40	24	69	56	19
79	80	49	88	1
27	114	21	122	121
46	75	4	28	99
53	58	16	109	2

State Akhir

16	109	123	36	19
102	122	81	37	113
55	18	115	56	77
87	73	119	78	95
1	86	62	12	39

89	15	11	103	93
84	83	57	68	2
28	99	88	32	105
34	3	76	74	13
46	8	29	69	67

64	30	35	54	106
21	6	80	82	4
96	7	9	117	79
116	23	53	41	40
100	26	92	10	44

42	121	94	58	20
47	60	97	66	5
85	25	51	61	90
124	65	98	27	43
114	33	108	118	110

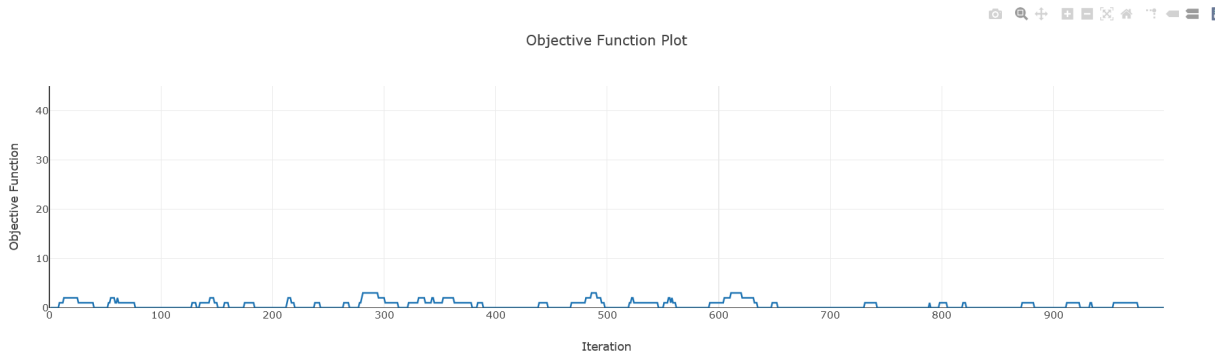
104	111	70	75	45
72	59	71	22	50
49	31	107	17	101
125	120	24	14	52
112	63	38	48	91

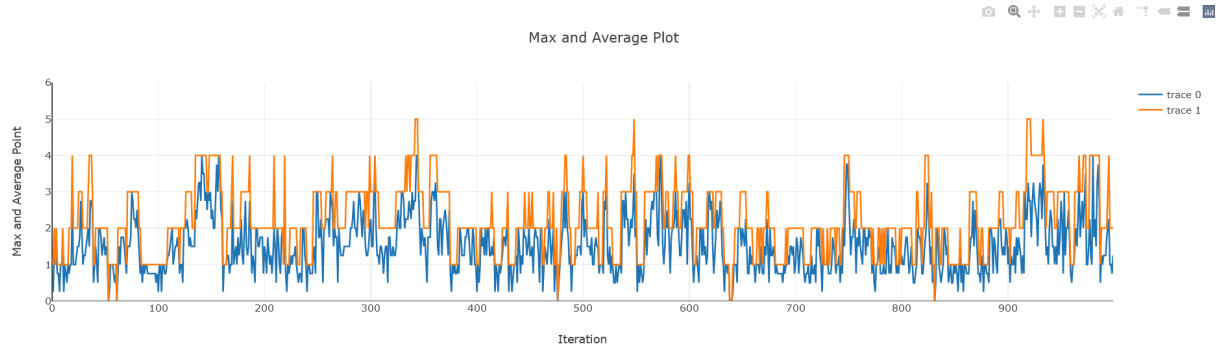
Final Objective Function Value: 1

Durasi: 444.791708 ms	Banyak Iterasi: 1000

Iteration Control (4 population Variable, Test 1)

Plot objective func/iteration





Jumlah Populasi 4

State Awal

72	33	18	73	1
29	8	105	67	34
120	41	108	75	58
95	81	28	122	106
83	109	59	116	94

66	57	47	60	53
77	39	97	61	16
26	9	102	62	7
51	17	80	12	48
14	37	64	56	36

52	110	27	125	85
113	55	63	114	88
90	115	38	30	76
22	3	10	32	100
70	118	96	31	6

89	103	98	112	50
84	71	43	49	44
4	20	104	2	91
111	117	42	13	35
92	24	123	25	69

23	40	86	78	124
19	45	46	65	82
11	79	68	121	15
93	54	21	101	99
5	119	74	107	87

State Akhir

60	63	36	18	89
79	90	115	80	26
33	41	13	100	61
118	95	32	45	16
30	58	67	125	107

29	116	77	87	4
101	21	68	92	112
34	46	42	74	64
70	12	88	108	123
86	43	66	6	44

28	31	111	98	17
93	81	104	71	3
75	82	122	51	85
117	59	2	110	53
23	8	49	11	48

22	94	78	113	97
56	105	73	47	124
103	50	15	25	55
40	72	106	5	38
35	27	83	10	9

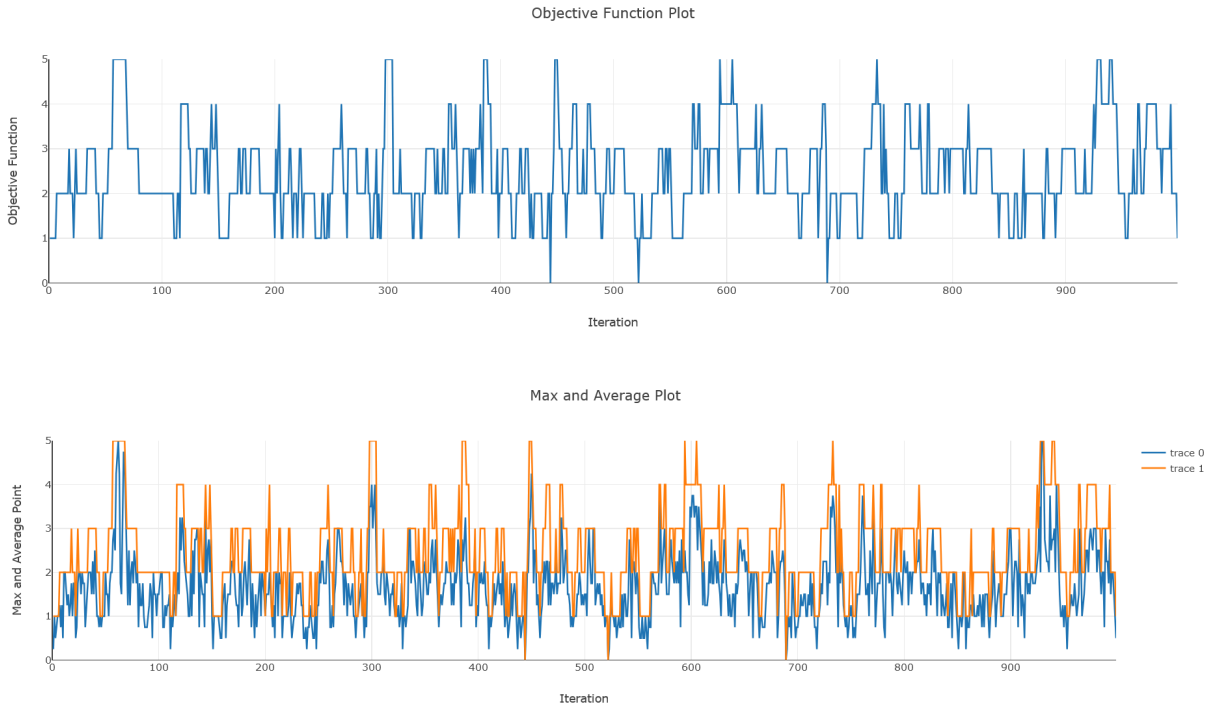
20	91	1	52	19
120	114	109	37	76
102	14	99	62	65
69	54	39	7	119
57	121	96	24	84

Final Objective Function Value: 2

Durasi: 618.538316 ms	Banyak Iterasi: 1000

Iteration Control (4 population Variable, Test 2)

Plot objective func/iteration



Jumlah Populasi 4

State Awal

96	118	4	53	77
51	124	115	95	9
85	57	45	106	21
50	11	68	8	69
39	62	20	99	120

67	88	78	26	29
93	80	100	7	17
14	60	24	32	76
102	90	12	61	3
94	58	15	101	116

47	6	119	66	81
109	54	31	73	103
27	108	55	92	70
13	87	28	35	114
105	43	16	38	1

65	52	49	41	25
71	19	5	40	79
89	33	18	42	10
23	64	122	107	30
104	112	46	82	2

63	113	22	75	72
97	121	56	91	59
98	74	125	84	48
123	86	37	110	117
83	111	34	44	36

State Akhir

35	16	28	54	71
40	101	47	23	30
75	37	53	31	8
70	39	52	4	65
115	100	25	36	76

32	120	7	41	81
102	6	82	26	1
83	72	88	58	111
9	18	112	90	113
95	108	55	44	63

61	92	98	2	86
49	80	27	124	121
67	43	73	57	97
22	45	24	13	125
46	17	122	119	5

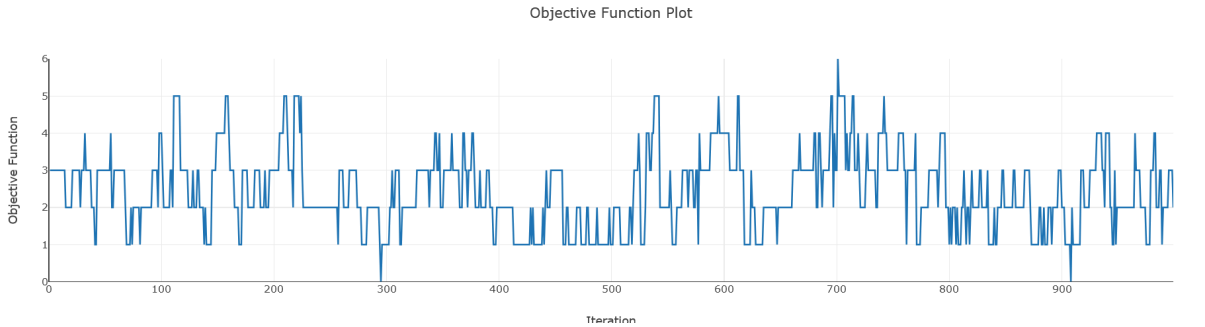
3	33	38	42	79
74	118	114	117	34
89	66	110	11	85
20	94	51	64	116
106	104	107	93	105

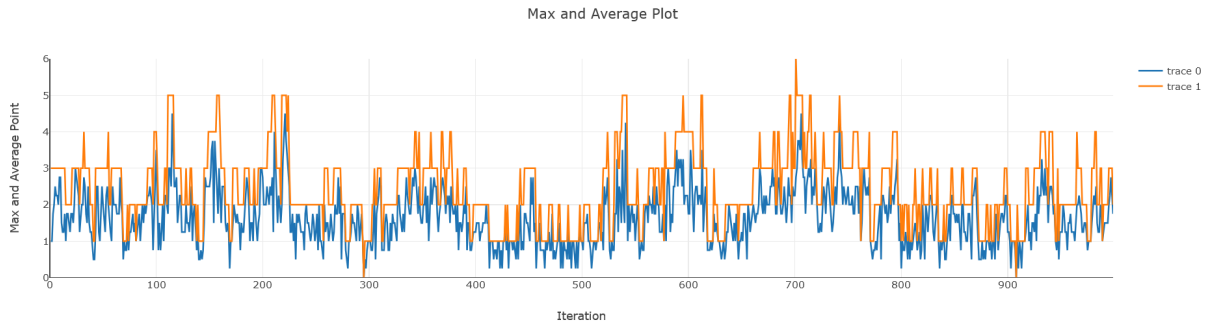
15	59	62	123	14
68	99	50	29	77
103	78	69	109	84
60	91	10	19	96
48	56	12	87	21

Final Objective Function Value: 2	
Durasi: 617.3901000000001 ms	Banyak Iterasi: 1000

Iteration Control (4 population Variable, Test 3)

Plot objective func/iteration





Jumlah Populasi 4

State Awal

99	42	89	88	32
61	24	51	122	67
29	18	102	119	116
15	83	109	28	8
92	40	95	36	75

97	74	70	54	46
10	13	56	38	90
68	41	31	113	112
58	5	34	111	121
3	85	52	50	48

79	53	23	101	33
60	123	4	2	77
106	80	72	125	94
65	39	117	57	49
43	120	27	71	17

22	93	114	21	45
107	91	20	1	6
59	47	98	11	105
84	44	76	25	86
62	9	96	30	16

81	14	103	12	64
78	100	82	104	73
108	55	37	26	110
87	19	118	7	63
69	66	115	35	124

State Akhir

41	82	105	8	79
11	42	68	75	43
98	102	95	113	39
67	116	88	76	61
90	50	30	52	7

122	21	48	34	35
29	33	9	125	4
58	45	77	5	60
56	121	13	65	89
49	16	46	81	63

40	100	93	91	22
110	84	24	69	72
38	123	106	28	23
51	114	62	104	115
55	19	14	97	109

3	57	20	80	25
117	119	70	17	108
112	120	99	59	66
118	37	71	103	78
10	124	94	36	107

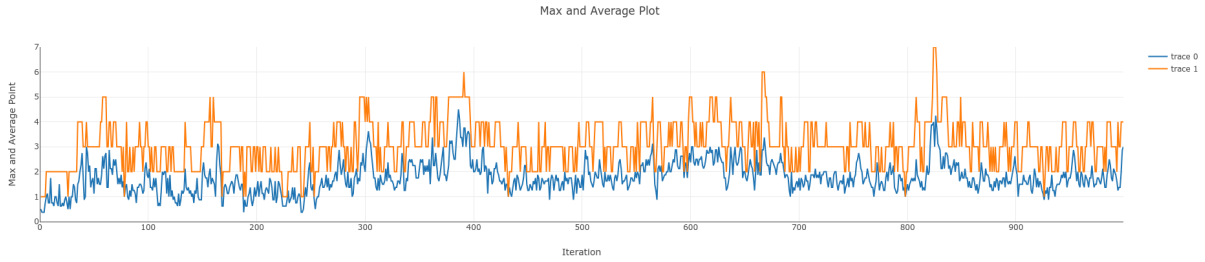
18	31	47	85	6
83	26	54	96	27
74	111	15	32	87
73	64	1	2	86
92	44	53	101	12

Final Objective Function Value: 2

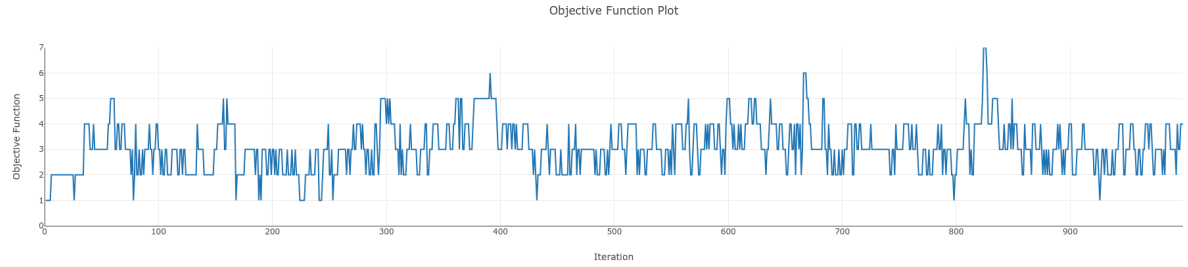
Durasi: 606.9865090000001 ms	Banyak Iterasi: 1000

Iteration Control (8 population Variable, Test 1)

Plot objective func/iteration



Objective Function Plot



Jumlah Populasi: 8

State Awal

85	102	79	83	25
29	37	38	2	27
9	109	125	71	23
22	124	58	47	122
62	43	92	90	7

49	14	21	60	74
118	3	117	119	46
45	24	5	86	100
50	103	77	80	106
20	114	34	76	65

72	81	26	73	48
108	116	52	70	110
123	95	66	82	93
30	13	67	68	104
18	75	107	44	19

11	121	78	10	88
101	28	84	64	42
56	55	1	16	35
4	69	91	59	98
41	94	115	12	17

112	6	97	89	99
32	51	113	31	120
8	111	61	15	36
39	33	40	96	53
63	87	54	105	57

State Akhir

58	36	93	82	73
26	94	40	99	39
27	92	81	85	71
55	41	45	51	123
125	52	75	37	56

22	21	20	67	10
111	117	103	62	70
47	76	46	78	116
87	100	89	28	23
60	38	44	101	32

2	19	3	5	108
118	12	14	61	98
95	4	54	59	34
91	11	113	64	42
105	90	110	102	57

74	63	114	83	72
97	6	119	29	48
79	18	96	33	17
124	1	109	69	49
15	112	25	68	121

16	13	35	8	80
50	24	30	107	84
77	66	31	120	88
86	7	115	122	65
53	9	104	106	43

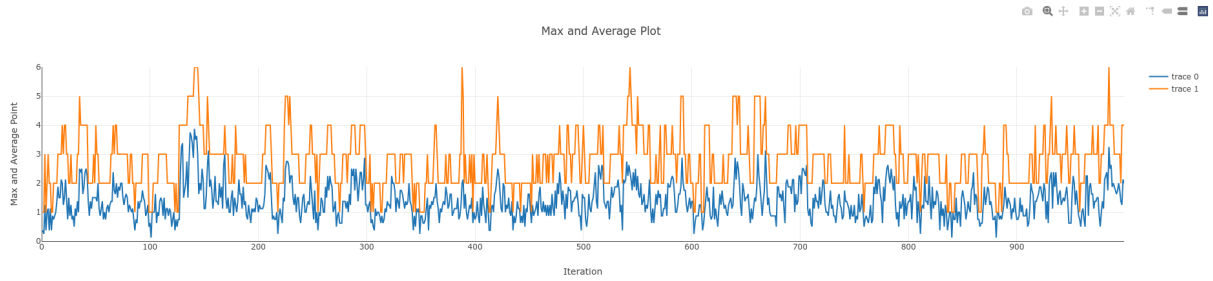
Final Objective Function Value: 4

Durasi: 1691.175807 ms

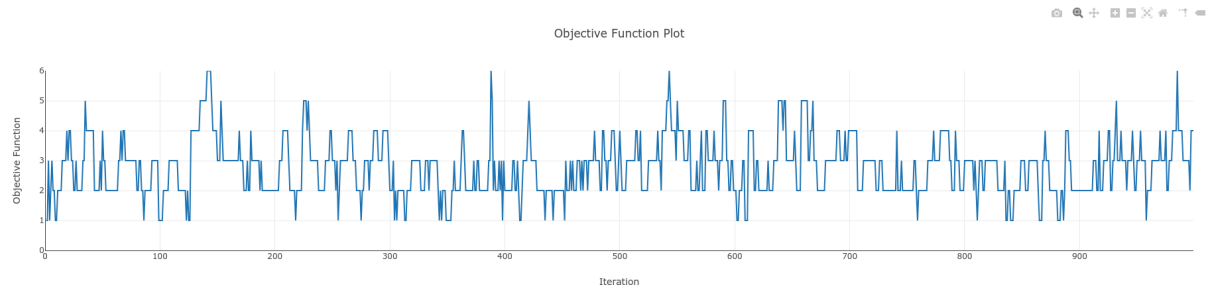
Banyak Iterasi: 1000

Iteration Control (8 population Variable, Test 2)

Plot objective func/iteration



Objective Function Plot



Jumlah Populasi: 8

State Awal

101	4	66	109	100
45	58	81	86	41
35	46	26	18	95
75	67	22	34	88
52	47	113	120	40

59	69	91	93	27
68	78	48	38	25
7	23	98	104	13
21	80	9	33	111
29	11	122	83	94

62	114	50	73	15
97	115	44	57	125
107	19	72	89	31
118	121	24	74	116
106	14	39	17	117

123	85	10	55	96
56	87	84	28	76
6	108	43	65	63
53	12	37	92	2
102	119	3	1	79

77	82	103	20	70
112	60	71	51	90
54	61	110	5	36
105	99	42	30	124
49	32	64	8	16

State Akhir

29	95	54	110	74
84	99	12	67	7
66	124	9	11	105
86	101	90	115	21
100	6	41	22	108

87	20	73	119	8
2	23	93	50	80
91	72	4	125	17
14	1	26	51	77
123	118	85	49	65

63	36	116	122	64
13	25	55	15	53
46	98	112	44	19
59	88	89	40	109
57	113	30	76	107

58	43	102	56	94
68	42	34	33	83
52	45	114	38	3
117	121	103	81	32
39	18	27	97	96

106	79	37	61	120
62	78	48	70	71
111	5	10	92	60
47	69	28	31	16
104	24	75	82	35

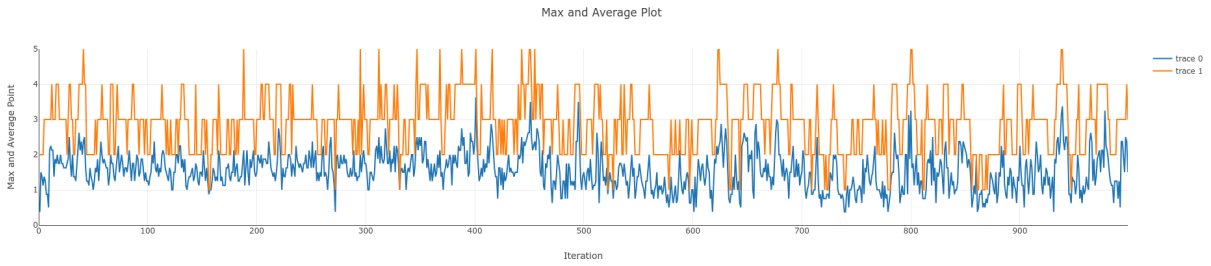
Final Objective Function Value: 2

Durasi: 1716.3622520000001 ms

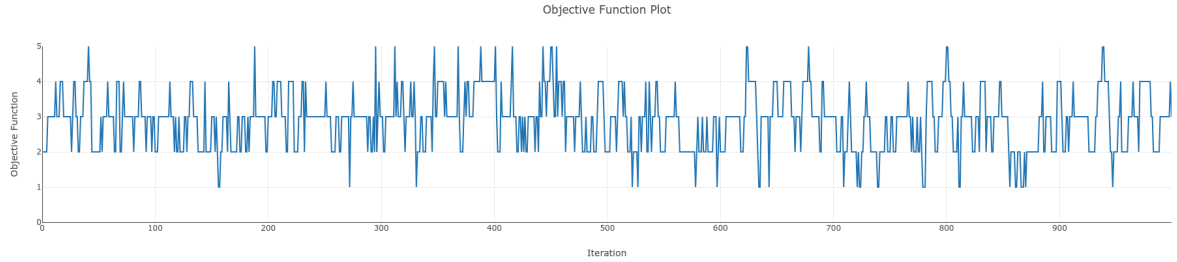
Banyak Iterasi: 1000

Iteration Control (8 population Variable, Test 3)

Plot objective func/iteration



Objective Function Plot



Jumlah Populasi: 8

State Awal

78	62	51	115	29
38	4	108	47	23
112	31	8	12	91
59	72	56	39	6
113	44	93	2	74

24	88	81	68	10
3	36	18	40	94
79	107	53	16	109
114	102	30	103	104
119	118	99	52	89

37	41	82	35	124
71	75	105	46	42
98	85	101	83	28
86	9	120	87	122
92	111	106	96	34

20	22	64	32	49
17	14	25	63	19
60	50	117	11	48
15	123	43	5	90
58	95	69	55	26

27	67	61	7	110
73	57	21	76	80
125	65	84	33	45
121	1	70	100	77
66	13	54	116	97

State Akhir

10	112	65	61	89
82	91	108	18	41
75	122	124	73	33
47	118	14	107	94
99	120	77	74	93

87	84	78	49	44
71	29	121	102	57
80	28	42	55	24
70	23	67	68	30
85	22	98	35	26

4	113	19	63	101
46	111	45	96	90
79	58	43	76	66
103	2	15	11	106
9	72	123	81	117

6	100	38	5	8
83	1	95	34	109
48	37	125	7	88
31	52	3	119	110
69	92	60	21	13

16	40	12	97	62
64	104	25	59	27
54	114	32	20	53
86	39	105	115	116
17	56	51	36	50

Final Objective Function Value: 3

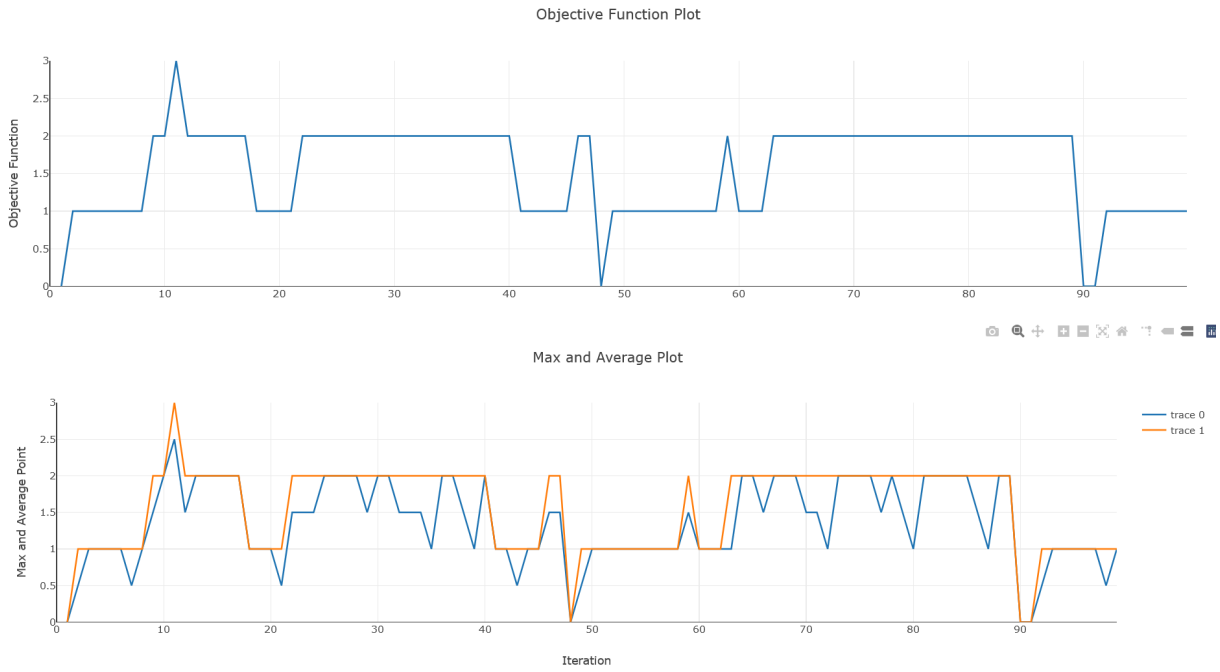
Durasi: 1775.602055 ms

Banyak Iterasi: 1000

Population Control (Iteration Variable)

Population Control (Iteration Variable 1, Test 1)

Plot objective func/iteration



Jumlah Populasi: 2

State Awal

27	56	96	91	43
124	92	32	42	106
29	52	33	114	47
67	112	28	7	119
45	111	50	8	95

49	6	77	20	41
48	102	115	73	40
2	117	100	39	55
105	103	70	68	15
99	118	11	104	81

37	109	116	25	94
21	5	54	97	53
23	1	78	17	59
86	80	71	83	34
14	101	85	113	75

107	46	13	57	121
19	123	76	72	60
63	79	22	58	62
87	98	89	65	108
64	69	12	18	9

88	125	93	82	36
24	38	3	30	122
84	51	90	66	10
31	120	35	74	61
4	16	110	44	26

State Akhir

94	79	21	59	113
90	40	69	110	28
118	23	77	86	56
125	62	33	6	35
54	80	106	124	65

72	92	2	32	82
38	89	121	100	66
98	53	26	1	57
84	31	20	114	109
115	24	63	17	49

36	119	67	123	3
19	103	4	107	95
30	70	52	74	122
117	39	27	50	60
76	116	96	108	64

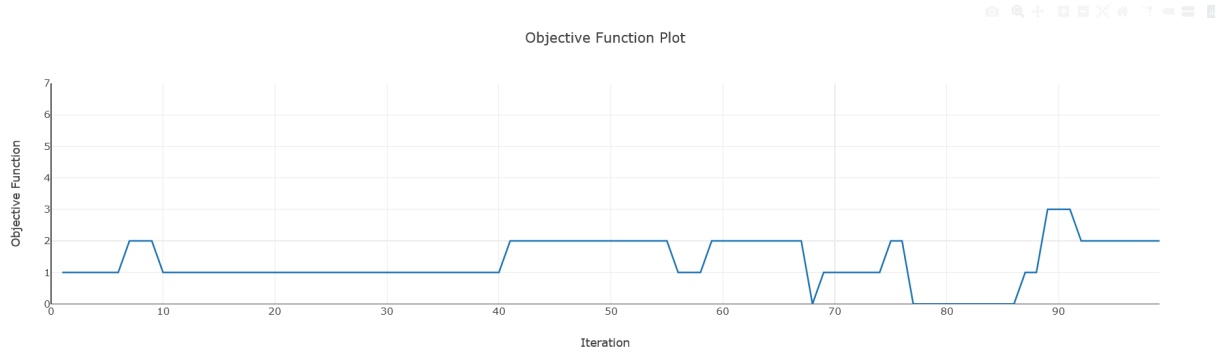
43	99	48	7	88
73	8	75	37	12
61	93	45	34	42
104	87	120	16	105
41	46	15	97	29

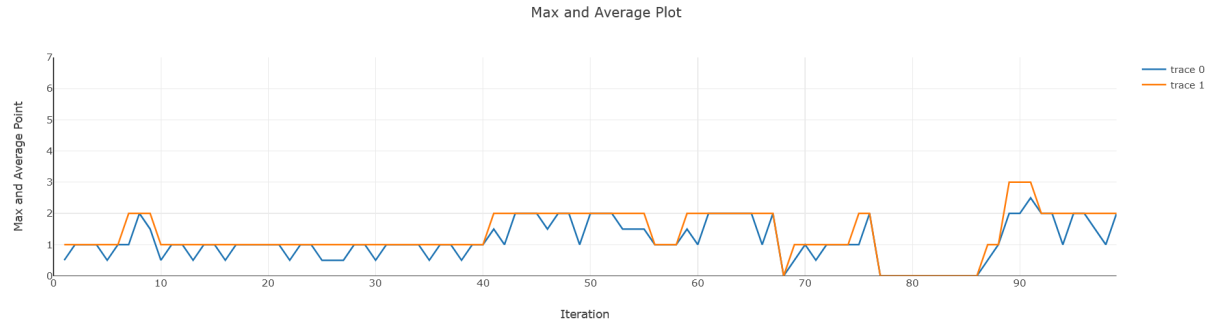
83	14	51	91	111
13	47	71	102	85
44	5	78	9	58
18	11	55	68	101
25	10	112	22	81

Final Objective Function Value: 1	
Durasi: 30.611866000000003 ms	Banyak Iterasi: 100

Population Control (Iteration Variable 1, Test 2)

Plot objective func/iteration





Jumlah Populasi: 2

State Awal

119	93	78	26	9
46	103	60	104	50
124	64	117	69	112
76	107	16	70	53
73	75	68	4	49

38	58	10	120	15
99	123	40	88	42
3	45	109	35	54
111	90	81	89	25
20	28	61	55	97

22	56	106	100	92
67	8	114	118	18
37	27	48	47	62
1	84	86	83	101
39	82	115	34	43

44	80	71	19	105
122	125	57	116	2
23	30	95	17	24
113	74	41	108	6
102	66	59	36	121

14	5	12	72	31
85	32	91	7	63
96	21	51	110	33
29	65	11	79	87
13	77	98	94	52

State Akhir

54	78	72	112	36
118	27	13	103	68
102	19	88	111	32
17	114	18	98	42
63	46	65	4	84

66	122	124	37	56
95	116	51	74	21
6	79	30	2	107
43	93	53	109	7
12	99	105	69	96

110	101	83	14	16
75	71	38	11	3
45	86	39	26	119
47	94	29	81	55
70	57	59	5	23

58	80	60	9	91
100	76	49	82	87
90	15	113	48	92
104	64	31	35	125
52	50	24	25	120

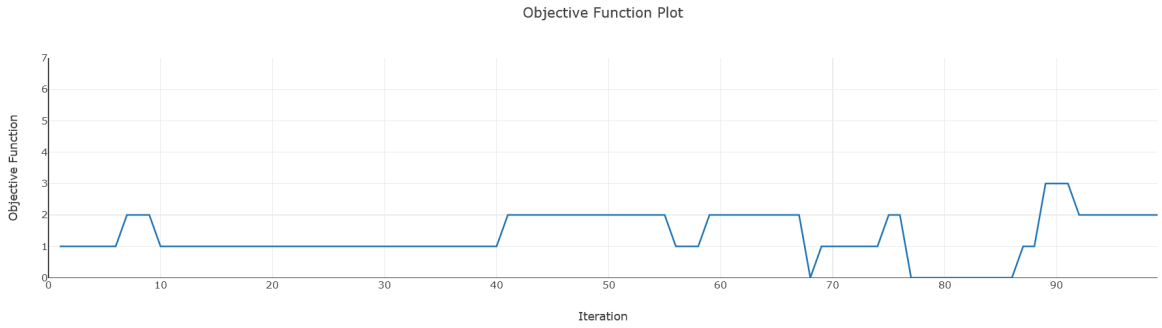
40	67	1	28	108
20	10	106	62	121
123	89	61	97	41
77	33	117	22	73
44	115	34	85	8

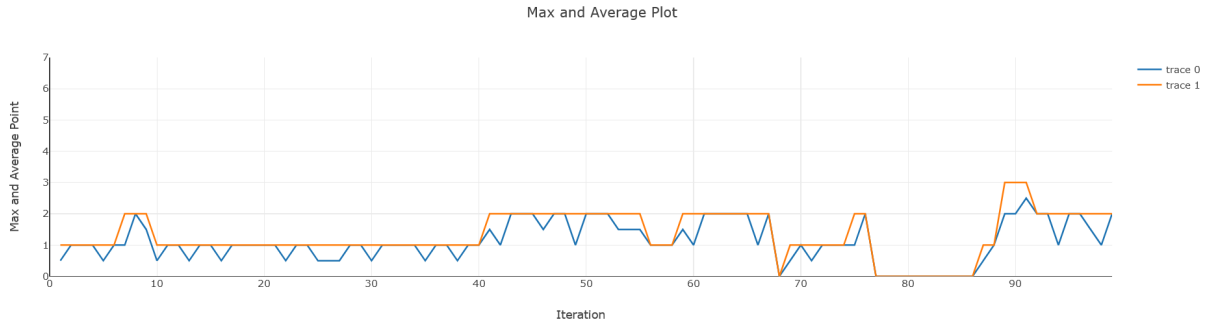
Final Objective Function Value: 1

Durasi: 30.552316 ms	Banyak Iterasi: 100

Population Control (Iteration Variable 1, Test 3)

Plot objective func/iteration





Jumlah Populasi 2

State Awal

90	81	68	80	50
121	84	46	11	24
60	67	7	101	115
21	70	49	32	94
36	117	26	96	108

5	12	93	91	20
2	104	105	82	88
102	109	14	63	15
16	44	72	95	87
51	69	22	23	45

28	103	99	122	76
125	107	65	58	100
30	37	19	85	52
116	41	8	13	71
56	77	54	33	38

34	114	97	73	113
9	83	29	106	39
47	110	40	55	31
43	27	86	120	35
98	62	48	1	118

78	57	111	74	59
6	75	17	42	79
3	4	112	92	18
66	53	124	61	89
10	119	64	25	123

State Akhir:

74	123	81	57	8
48	19	25	70	29
72	88	94	87	47
89	40	56	125	2
55	3	119	33	1

50	27	5	107	32
90	71	98	46	43
36	41	52	9	116
122	24	26	20	80
105	16	75	63	23

58	96	109	97	37
64	67	45	38	18
86	82	51	34	85
66	22	21	28	39
101	76	115	10	118

112	14	92	79	84
117	7	102	68	114
103	30	11	110	120
95	69	104	53	124
77	54	83	35	31

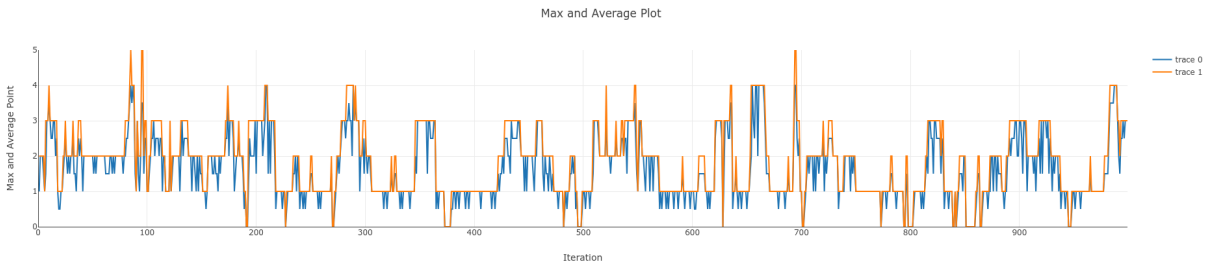
100	12	6	91	42
49	108	113	13	15
73	111	61	4	65
93	78	106	60	17
121	99	62	59	44

Final Objective Function Value: 1

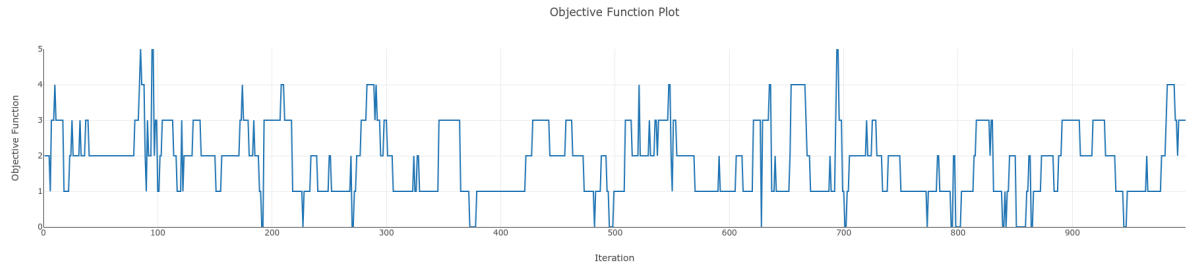
Durasi: 30.035064000000002 ms	Banyak Iterasi: 100

Population Control (Iteration Variable 2, Test 1)

Plot objective func/iteration



Objective Function Plot



Jumlah Populasi: 2

State Awal

94	82	53	79	7
36	26	41	28	51
125	118	76	16	116
85	61	81	102	123
78	1	56	100	52

67	15	33	124	46
45	35	17	50	44
111	107	58	37	104
9	71	91	113	92
106	20	32	70	63

73	99	98	23	119
30	21	10	74	59
68	5	2	54	19
77	69	80	60	75
40	72	6	25	112

90	31	42	22	29
115	12	66	48	108
64	18	27	88	96
8	110	122	62	86
87	101	14	3	117

47	93	65	95	120
89	13	34	24	114
4	57	97	39	103
55	109	121	83	38
49	43	105	11	84

State Akhir

94	82	53	79	7
36	26	41	28	51
125	118	76	16	116
85	61	81	102	123
78	1	56	100	52

67	15	33	124	46
45	35	17	50	44
111	107	58	37	104
9	71	91	113	92
106	20	32	70	63

73	99	98	23	119
30	21	10	74	59
68	5	2	54	19
77	69	80	60	75
40	72	6	25	112

90	31	42	22	29
115	12	66	48	108
64	18	27	88	96
8	110	122	62	86
87	101	14	3	117

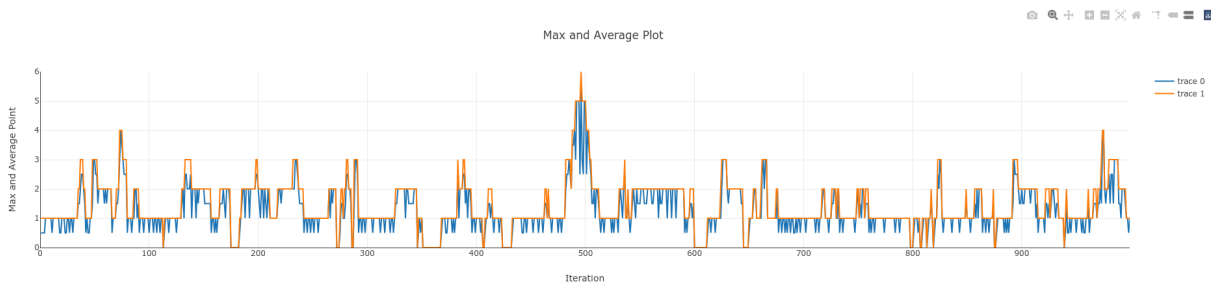
47	93	65	95	120
89	13	34	24	114
4	57	97	39	103
55	109	121	83	38
49	43	105	11	84

Final Objective Function Value: 3

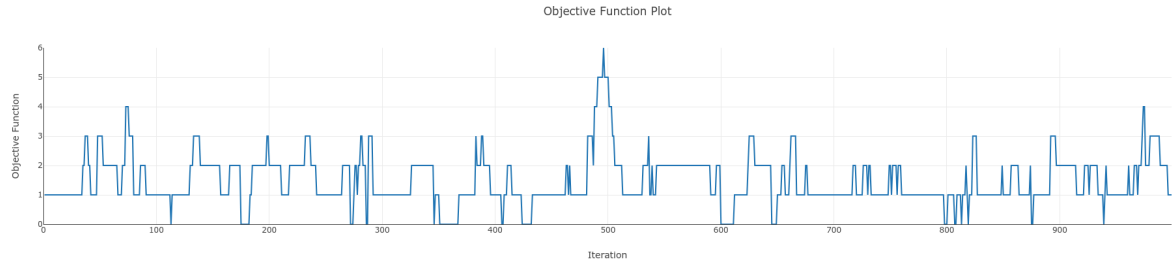
Durasi: 426.966453 ms	Banyak Iterasi: 1000

Population Control (Iteration Variable 2, Test 2)

Plot objective func/iteration



Objective Function Plot



Jumlah Populasi 2

State Awal

7	3	94	111	105
47	5	14	85	107
17	74	8	61	11
45	69	106	113	116
49	10	37	41	118

73	55	83	39	109
12	119	27	79	44
92	110	89	54	101
82	81	91	78	123
76	124	30	62	71

120	36	99	72	53
32	102	115	50	77
24	46	31	93	28
51	57	112	125	48
6	122	16	114	64

34	96	80	38	66
87	29	58	4	25
26	22	33	67	103
90	68	84	104	117
65	70	21	1	56

97	2	19	43	13
20	40	121	23	35
86	98	100	52	108
42	59	63	88	9
75	95	18	60	15

State Akhir

23	48	118	115	10
19	24	74	89	50
97	125	2	41	22
96	57	70	77	35
95	58	116	5	93

59	73	104	103	1
15	99	12	112	55
42	111	80	8	107
34	4	85	18	110
17	28	16	105	90

108	109	69	33	94
113	86	76	51	72
61	31	39	56	60
54	100	87	102	68
20	9	124	119	3

6	26	83	7	98
65	43	30	79	117
47	14	40	122	106
44	64	91	21	121
27	78	52	13	67

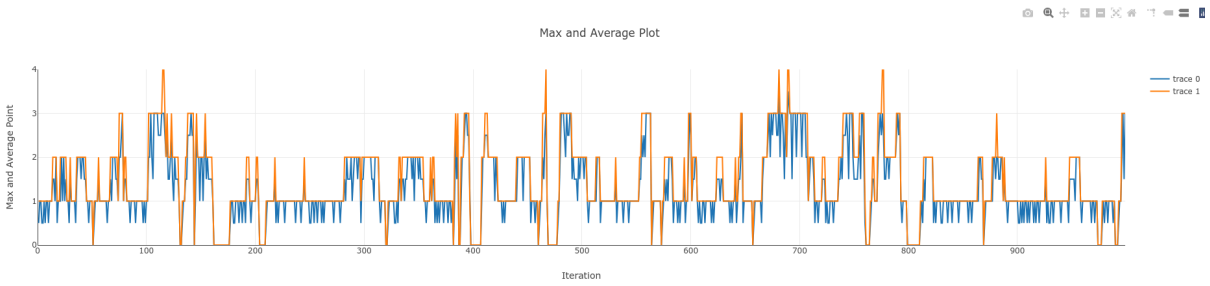
11	120	45	123	66
29	38	37	88	82
25	81	92	84	63
46	49	53	32	101
71	114	75	62	36

Final Objective Function Value: 1

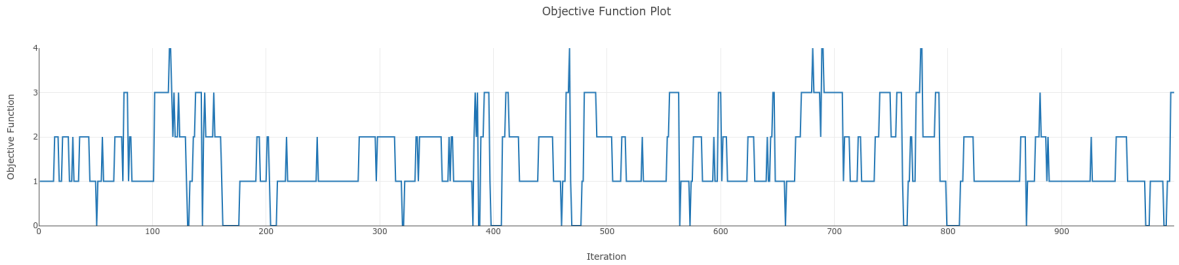
Durasi: 433.85607100000004 ms	Banyak Iterasi: 1000

Population Control (Iteration Variable 2, Test 3)

Plot objective func/iteration



Objective Function Plot



Jumlah Populasi 2

State Awal

9	80	52	43	123
113	75	6	34	39
50	74	125	31	1
16	62	100	72	58
54	119	114	44	51

115	83	65	22	3
110	95	79	92	82
91	57	40	77	121
45	29	87	32	4
93	81	14	71	17

41	124	36	15	109
53	7	2	116	24
20	18	25	69	35
103	68	105	5	96
23	108	37	46	107

10	78	112	59	85
102	38	26	86	104
61	106	64	67	8
101	49	97	120	111
99	27	42	89	19

48	33	90	28	55
56	94	98	122	73
30	118	84	60	117
12	11	47	13	76
21	70	63	88	66

State Akhir

110	61	21	52	87
4	53	34	39	65
81	114	15	20	17
121	82	83	23	14
99	90	68	66	103

29	8	92	78	108
119	69	12	26	122
2	30	86	48	24
16	88	96	28	38
47	76	101	5	91

118	67	64	104	50
107	100	25	93	31
59	57	42	63	94
85	10	62	35	113
70	27	125	117	51

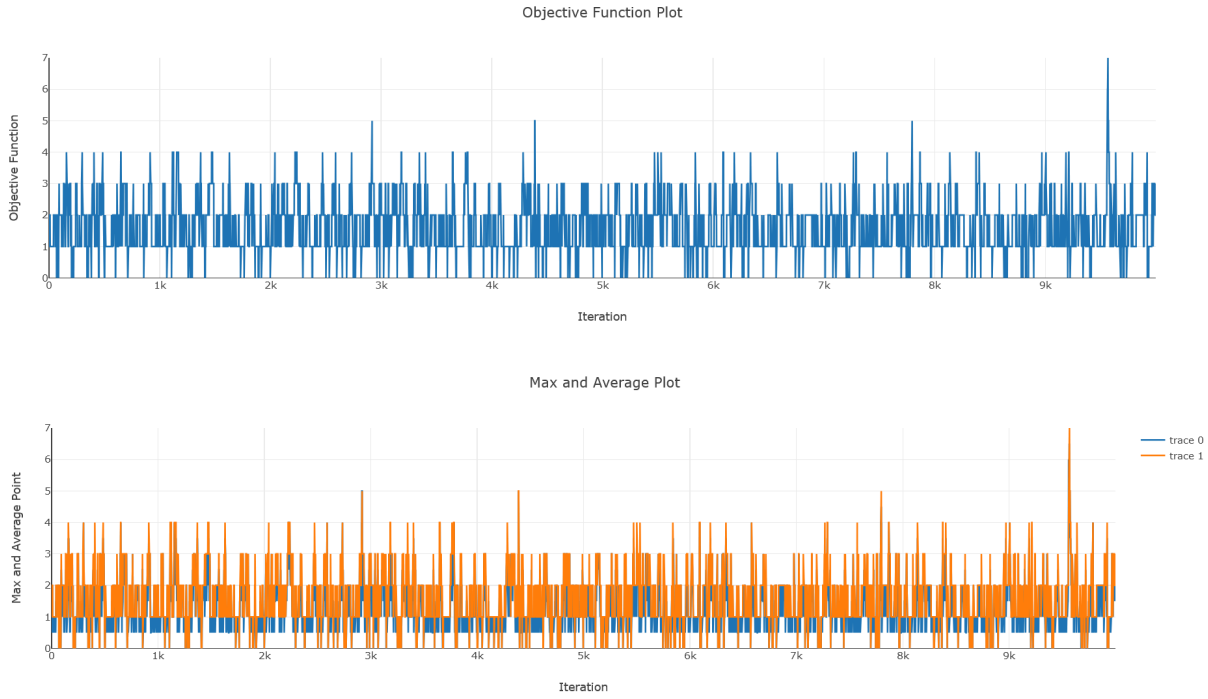
95	55	89	1	44
72	98	22	43	105
60	49	37	73	46
56	71	36	19	80
54	109	40	102	3

124	112	6	111	106
120	9	33	32	75
77	97	79	11	116
84	123	115	45	18
58	13	41	7	74

Final Objective Function Value: 3	
Durasi: 440.621817 ms	Banyak Iterasi: 1000

Population Control (Iteration Variable 3, Test 1)

Plot objective func/iteration



Jumlah Populasi 2

State Awal

64	38	114	46	14
90	23	124	16	17
122	55	56	95	43
44	1	80	98	120
84	79	115	111	69

8	77	57	22	75
68	18	50	105	108
53	97	52	117	96
48	6	86	33	10
11	71	100	94	76

31	60	29	67	36
19	82	106	85	107
93	99	102	59	83
89	7	62	125	63
74	54	35	112	2

51	113	104	78	30
109	121	47	66	21
72	5	37	27	116
70	13	45	42	15
28	65	34	101	73

91	49	88	58	81
39	103	4	25	123
40	20	41	24	87
110	9	3	61	119
118	26	92	12	32

State Akhir

10	90	84	38	67
26	88	83	118	87
50	102	76	21	124
99	41	123	114	52
54	24	74	53	14

79	94	80	43	45
95	121	12	110	116
9	11	93	28	1
46	70	103	65	101
113	122	40	78	19

2	92	120	117	62
44	17	30	51	3
73	13	119	69	98
104	59	105	91	4
33	63	20	106	32

56	68	77	39	111
25	55	15	85	36
107	8	75	6	109
61	27	48	34	86
81	97	31	96	125

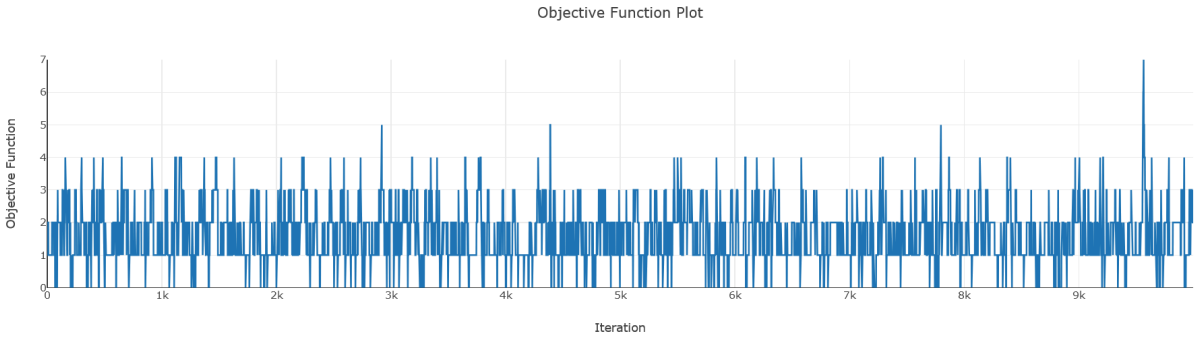
57	23	64	60	66
22	5	72	49	82
108	47	16	29	37
35	42	112	71	18
7	100	89	58	115

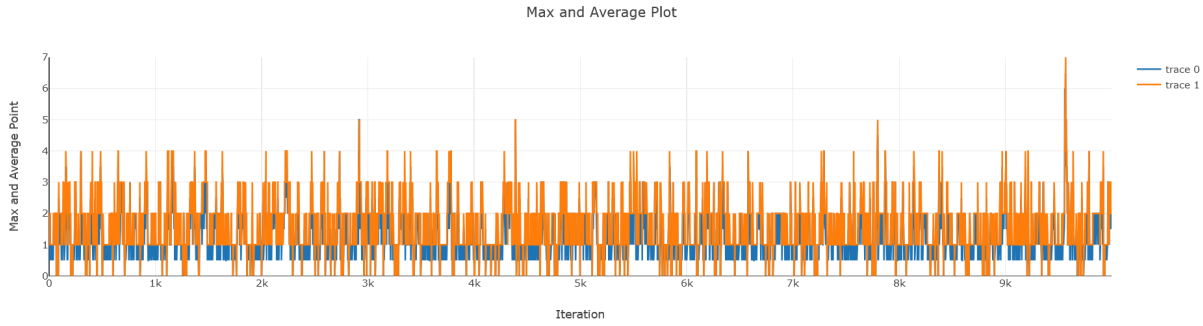
Final Objective Function Value: 1

Durasi: 3108.1004040000003 ms	Banyak Iterasi: 10000

Population Control (Iteration Variable 3, Test 2)

Plot objective func/iteration





Jumlah Populasi 2

State Awal:

39	90	80	81	48
27	50	23	84	120
18	12	115	14	76
51	113	4	109	75
49	99	114	22	8

47	125	106	34	9
117	74	92	104	58
59	83	29	25	89
73	91	123	72	119
101	71	41	33	96

3	15	11	64	1
46	82	60	122	102
98	69	78	62	57
63	2	40	110	6
55	86	77	16	20

87	44	118	38	67
121	37	7	88	32
66	19	124	28	103
108	65	95	13	85
94	52	68	105	21

43	53	111	97	107
54	17	112	61	93
56	26	30	35	42
24	70	5	116	10
45	100	36	31	79

State Akhir

117	118	40	72	25
84	7	35	94	12
124	104	79	21	65
89	3	115	46	64
30	62	122	109	77

107	22	66	49	125
53	8	28	114	44
120	110	39	9	81
105	29	86	60	59
97	31	69	55	37

100	10	26	51	50
13	56	17	23	103
121	15	1	108	76
34	80	101	57	99
36	113	98	58	52

88	27	78	43	41
123	102	106	6	19
38	2	95	42	116
48	4	91	111	61
11	83	33	54	119

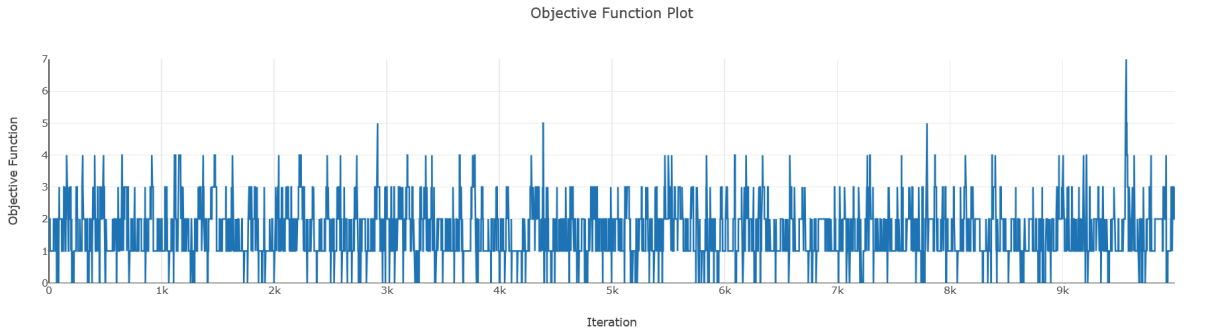
75	16	68	63	92
82	5	67	70	73
90	112	45	47	93
74	85	71	14	96
20	32	87	18	24

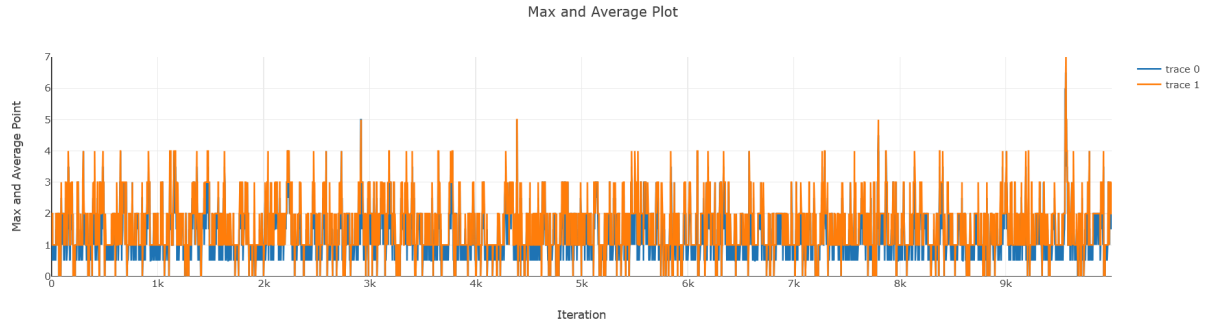
Final Objective Function Value: 2

Durasi: 3135.84121 ms	Banyak Iterasi: 10.000

Population Control (Iteration Variable 3, Test 3)

Plot objective func/iteration





Jumlah Populasi 2

State Awal

60	30	75	28	117
64	45	80	63	37
40	91	50	27	6
13	90	41	96	18
44	21	52	65	111

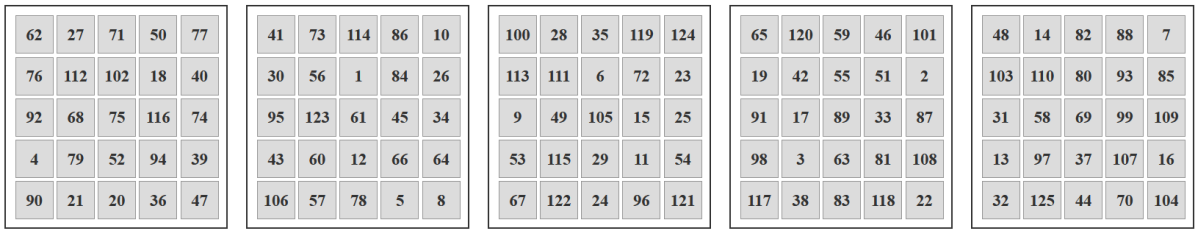
125	82	116	93	103
94	38	85	121	72
24	9	55	97	70
48	104	108	69	57
56	16	92	67	14

8	112	66	122	3
110	5	76	32	118
35	25	47	73	12
61	20	68	87	46
81	11	78	2	100

88	98	105	59	74
89	99	31	17	33
86	77	15	36	123
58	101	102	34	84
49	71	107	7	1

115	51	39	54	109
26	23	62	22	83
95	113	114	43	29
79	53	106	124	119
120	42	10	4	19

State Akhir



62	27	71	50	77
76	112	102	18	40
92	68	75	116	74
4	79	52	94	39
90	21	20	36	47

41	73	114	86	10
30	56	1	84	26
95	123	61	45	34
43	60	12	66	64
106	57	78	5	8

100	28	35	119	124
113	111	6	72	23
9	49	105	15	25
53	115	29	11	54
67	122	24	96	121

65	120	59	46	101
19	42	55	51	2
91	17	89	33	87
98	3	63	81	108
117	38	83	118	22

48	14	82	88	7
103	110	80	93	85
31	58	69	99	109
13	97	37	107	16
32	125	44	70	104

Final Objective Function Value: 2	
Durasi: 3134.626032 ms	Banyak Iterasi: 10.000

Analisis

Algoritma yang paling mendekati global maxima adalah algoritma Hil-climbing with sideways move. Algoritma tersebut memiliki *objective function* maksimum tertinggi yaitu 55 point. Meskipun demikian, point yang dicapai ini masih cukup jauh dibandingkan *global maxima* yaitu 109 point. Algoritma ini memiliki hasil demikian karena algoritma ini memperbolehkan eksplorasi pada *state* yang memiliki *objective function* yang sama atau lebih baik. Namun, sebelum akhir pencarian, algoritma ini mencapai *plateau* yang sangat besar (lebih dari 10.000 sideways move) sehingga hasil pencarian *stuck* pada 55 point.

Algoritma Steepest Ascent Hill-climbing memiliki hasil *objective function* yang masih cukup jauh dari global maxima. Algoritma ini memiliki *objective function* tertinggi bernilai 31 point. Algoritma ini memiliki hasil demikian karena tidak seperti hill-climbing with sideways move, algoritma ini terbatas jika sudah tidak menemukan *successor* yang memiliki *objective function* yang lebih baik dibandingkan *current state*.

Algoritma random restart hill-climbing memiliki hasil *objective function* yang masih cukup jauh dari *global optima*, mirip seperti steepest ascent hill-climbing, algoritma ini memiliki nilai maksimum 31 point. Hasil tersebut didapatkan karena penukaran yang terjadi belum mendapat kesempatan yang cukup banyak untuk mencapai 109 point.

Algoritma stochastic hill-climbing memiliki hasil *objective function* tertinggi pada 41 point. Sama seperti hasil-hasil lainnya, hasil ini masih cukup jauh dari global optima. Algoritma ini memiliki hasil demikian karena *state random* yang dihasilkan kebetulan tidak ada yang memiliki hasil yang lebih baik ketika algoritma ini menemukan *plateau*. Sehingga algoritma ini stuck pada *plateau* tersebut dan hanya mencapai sekitar 41 point.

Algoritma simulated annealing memiliki hasil point yang cukup tinggi dibandingkan algoritma lainnya hanya tertinggal di belakang algoritma hill-climbing with sideways move. Algoritma ini memiliki point tertinggi pada 46 point. Hasil ini didapatkan karena cooling factor yang cukup besar meskipun tidak cukup besar untuk mencapai global optima.

Algoritma Genetic Algorithm bukanlah sebuah algoritma yang cocok untuk permasalahan ini. Hal ini dikarenakan perubahan satu buah nilai pada kubus sangat berpengaruh terhadap nilai fungsi. Pada saat penggabungan dua state, banyak nilai kubus yang saling terhubung, namun harus terpisah dikarenakan penggabungan tersebut. Sehingga pada akhir iterasi tidak mendapatkan hasil yang baik.

Seluruh algoritma membutuhkan waktu tergantung dengan parameter yang dibutuhkan. Salah satu hal yang bisa dibandingkan adalah Hill Climbing dengan Random Restart. Keduanya sama-sama menggunakan algoritma yang sama. Pada random restart, digunakan Hill Climbing berulang kali, namun hasil yang diberikan tidak sebanding dengan waktu yang dibutuhkan.

Genetic Algorithm memiliki hasil yang konsisten pada setiap test yang dilakukan. Stochastic memiliki hasil yang lumayan konsisten pada setiap test yang dilakukan yakni, 36, 41, dan 34. Pada Simulated Annealing hasilnya tidak jauh berbeda dengan nilai 43, 46, dan 45. Pada Random Restart Hill Climbing hasilnya tidak jauh berbeda juga yakni, 31,31 dan 37. Pada hill climbing with sideways hasilnya lebih bagus daripada yang lain dan konsisten pada nilai, 55, 52, dan 55. Pada hill climbing steepest ascent nilai yang didapatkan tidak jauh berbeda bahkan dua nilai memiliki nilai yang sama yaitu, 31, 31, dan 25.

Dalam percobaan yang kami lakukan, *Genetic Algorithm* tidak menghasilkan nilai yang signifikan. Hasil yang diperoleh sangat rendah dan tidak menunjukkan perbedaan yang berarti antara satu tes dengan tes lainnya.

Kesimpulan dan Saran

a. kesimpulan

Pada kasus dan parameter yang digunakan oleh kelompok kami, algoritma hill-climbing with sideways move merupakan algoritma yang terbaik untuk setidaknya mendapatkan hasil yang paling optimal meskipun hasil belum mencapai global optima. Hal ini didapatkan karena parameter dan *retry* yang cukup sempit sedangkan algoritma seperti simmulated annealing atau genetic algorithm membutuhkan kelonggaran dan kesempatan yang lebih luas untuk bereksplorasi dan menemukan solusi optimal.

b. saran

Saran bagi orang-orang yang ingin mengerjakan proyek ini lebih lanjut adalah jangan menggunakan bahasa yang kurang familiar oleh anggotanya apalagi hanya salah satu orang saja yang mengerti bahasanya. Gunakanlah bahasa yang kalian paling kuasai. Selain itu, bagi periset selanjutnya yang ingin melanjutkan mungkin bisa memperbanyak populasi dan iterasi yang ada jika memiliki banyak waktu

Pembagian Tugas

NIM	Nama	Tugas
13522086	Muhammad Atpur Rafif	Perancangan objective function, Implementasi code algoritma
13522088	Muhamad Rafli Rasyiidin	Perancangan objective function, Implementasi Integrasi server dengan UI
13522100	M. Hanief Fatkhan Nashrullah	Perancangan objective function, Eksperimen dan analisis
13522119	Indraswara Galih Jayanegara	Perancangan objective function, Eksperimen dan analisis

Referensi

Magisch Vierkant. (n.d.). *Features of the magic cube*. Retrieved October 1, 2024, from https://www.magischvierkant.com/three-dimensional-eng/magic-features/

OEIS Foundation Inc. (n.d.). *A027441*. Retrieved October 1, 2024, from https://oeis.org/A027441

Trump, W. (2003, June 19). *The successful search for the smallest perfect magic cube* (last modified: 2005, February 25). https://www.trump.de/magic-squares/magic-cubes/cubes-1.html