

```

1
2  switch (CrossoverOperator)
3  {
4      case CrossoverType.PMX:
5          // i1, i2 cut locations
6          // m[] partial map, m[i] => the mapping target of i
7
8          // define two crossover map for chromosome
9          index_1 = randomizer.Next(numberOfGenes);
10         while (index_2 == index_1)
11         {
12             index_2 = randomizer.Next(numberOfGenes);
13         }
14         // swap two numbers if index_1 is larger
15         if (index_2 < index_1)
16         {
17             temp_num = index_1;
18             index_1 = index_2;
19             index_2 = temp_num;
20         }
21         // initiate mapping
22         mapping = new int[numberOfGenes];
23         for (int i = 0; i < mapping.Length; i++) mapping[i] = -1;
24         // build up the mapping
25         for (int i = index_1; i < index_2; i++)
26         {
27
28             if (chromosomes[fatherIdx][i] == chromosomes[motherIdx][i]) continue;
29             if (mapping[chromosomes[fatherIdx][i]] == -1 &&
30                 mapping[chromosomes[motherIdx][i]] == -1)
31             {
32                 mapping[chromosomes[fatherIdx][i]] = chromosomes[motherIdx][i];
33                 mapping[chromosomes[motherIdx][i]] = chromosomes[fatherIdx][i];
34             }
35             else if (mapping[chromosomes[fatherIdx][i]] == -1)
36             {
37                 mapping[chromosomes[fatherIdx][i]] = mapping[chromosomes[motherIdx][i]];
38                 try
39                 {
40                     mapping[mapping[chromosomes[motherIdx][i]]] =
41                         chromosomes[fatherIdx][i];
42                 }
43                 catch (System.IndexOutOfRangeException Exception)
44                 {
45
46                 }
47                 mapping[chromosomes[motherIdx][i]] = -2;
48             }
49             else if (mapping[chromosomes[motherIdx][i]] == -1)
50             {
51                 try
52                 {
53                     mapping[chromosomes[motherIdx][i]] =
54                         mapping[chromosomes[fatherIdx][i]];
55                 }
56                 catch (System.IndexOutOfRangeException Exception) { }
57                 try
58                 {
59                     mapping[mapping[chromosomes[fatherIdx][i]]] =
60                         chromosomes[motherIdx][i];
61                 }
62                 catch (System.IndexOutOfRangeException Exception) { }
63                 try
64                 {
65                     mapping[chromosomes[fatherIdx][i]] = -2;
66                 }
67                 catch (System.IndexOutOfRangeException Exception) { }
68             }
69         }
70     }
71 }

```

```

66         else
67         {
68             try
69             {
70                 mapping[mapping[chromosomes[motherIdx][i]]] =
                    mapping[chromosomes[fatherIdx][i]];
71             }
72             catch (System.IndexOutOfRangeException Exception) { }
73
74             try
75             {
76                 mapping[mapping[chromosomes[fatherIdx][i]]] =
                    mapping[chromosomes[motherIdx][i]];
77             }
78             catch (System.IndexOutOfRangeException Exception) { }
79
80
81             mapping[chromosomes[fatherIdx][i]] = -3;
82             mapping[chromosomes[motherIdx][i]] = -3;
83         }
84     }
85
86     // crossover and make two children
87     for (int i = 0; i < numberOfGenes; i++)
88     {
89         if (index_1 <= i && i < index_2)
90         {
91             chromosomes[child1Idx][i] = chromosomes[motherIdx][i];
92             chromosomes[child2Idx][i] = chromosomes[fatherIdx][i];
93         }
94         else
95         {
96             if (mapping[chromosomes[fatherIdx][i]] < 0) chromosomes[child1Idx][i] =
                chromosomes[fatherIdx][i];
97             else chromosomes[child1Idx][i] = mapping[chromosomes[fatherIdx][i]];
98
99             if (mapping[chromosomes[motherIdx][i]] < 0) chromosomes[child2Idx][i] =
                chromosomes[motherIdx][i];
100            else chromosomes[child2Idx][i] = mapping[chromosomes[motherIdx][i]];
101        }
102    }
103    // crossover finished.
104
105    // check if all gene are distinct, if not let other chromosome replace it
106    for (int i = 0; i < populationSize * 3; i++)
107    {
108        temp = new int[numberOfGenes];
109        for (int j = 0; j < numberOfGenes; j++)
110        {
111            temp[j] = j;
112        }
113
114        if (chromosomes[i].Distinct().ToArray().Count() != numberOfGenes)
115        {
116            chromosomes[i] = temp.OrderBy(x => randomizer.Next()).ToArray();
117        }
118        else
119        {
120            break;
121        }
122    }
123
124    break;
125 case CrossoverType.OX:
126     // order crossover
127     // define two crossover map for chromosome
128     index_1 = randomizer.Next(numberOfGenes);
129     while (index_2 == index_1)
130     {

```

```

131         index_2 = randomizer.Next(numberOfGenes);
132     }
133     // swap two numbers if index_1 is larger
134     if (index_2 < index_1)
135     {
136         temp_num = index_1;
137         index_1 = index_2;
138         index_2 = temp_num;
139     }
140     // start creating crossover children
141     temp_num = 0;
142     temp_num_2 = 0;
143     for (int i = index_1; i < index_2; i++)
144     {
145         chromosomes[child1Idx][i] = chromosomes[fatherIdx][i];
146         chromosomes[child2Idx][i] = chromosomes[motherIdx][i];
147     }
148     for (int i = 0; i < numberOfGenes; i++)
149     {
150         if (!(chromosomes[child1Idx].Contains(chromosomes[motherIdx][i])))
151         {
152             if (temp_num == index_1) temp_num = index_2;
153             chromosomes[child1Idx][temp_num] = chromosomes[motherIdx][i];
154             temp_num += 1;
155         }
156         if (!(chromosomes[child2Idx].Contains(chromosomes[fatherIdx][i])))
157         {
158             if (temp_num_2 == index_1) temp_num_2 = index_2;
159             chromosomes[child2Idx][temp_num_2] = chromosomes[fatherIdx][i];
160             temp_num_2 += 1;
161         }
162     }
163     break;
164 case CrossoverType.POX:
165     // position-based crossover
166     temp_num = (int)Math.Round(numberOfGenes * crossoverRate);
167     temp_1 = new int[temp_num];
168     // create random index array
169     for (int i = 0; i < temp_num; i++)
170     {
171         temp_1[i] = randomizer.Next(numberOfGenes);
172     }
173     // crossover children
174     for (int i = 0; i < numberOfGenes; i++)
175     {
176         if (temp_1.Contains(i))
177         {
178             chromosomes[child1Idx][i] = chromosomes[fatherIdx][i];
179             chromosomes[child2Idx][i] = chromosomes[motherIdx][i];
180         }
181         else
182         {
183             chromosomes[child1Idx][i] = chromosomes[motherIdx][i];
184             chromosomes[child2Idx][i] = chromosomes[fatherIdx][i];
185         }
186     }
187     break;
188 case CrossoverType.OSS:
189     // order-based crossover
190     temp_num = (int)Math.Round(numberOfGenes * crossoverRate);
191     temp_1 = new int[temp_num];
192     // create random value array
193     for (int i = 0; i < temp_num; i++)
194     {
195         temp_1[i] = randomizer.Next(numberOfGenes);
196     }
197     // crossover children
198     for (int i = 0; i < numberOfGenes; i++)

```

```

200     {
201         if (!(temp_1.Contains(chromosomes[fatherIdx][i])))
202         {
203             chromosomes[child1Idx][i] = chromosomes[motherIdx][i];
204         }
205         else
206         {
207             chromosomes[child1Idx][i] = chromosomes[fatherIdx][i];
208         }
209
210         if (!(temp_1.Contains(chromosomes[motherIdx][i])))
211         {
212             chromosomes[child2Idx][i] = chromosomes[fatherIdx][i];
213         }
214         else
215         {
216             chromosomes[child2Idx][i] = chromosomes[motherIdx][i];
217         }
218     }
219
220     break;
221 case CrossoverType.OCCC:
222     break;
223 }
224
225
226
227 switch (CrossoverOperator)
228 {
229     case CrossoverType.OnePointCut:
230         for (int i = 0; i < numberOfGenes; i++)
231         {
232             if (i < temp_num)
233             {
234                 chromosomes[child1Idx][i] = chromosomes[fatherIdx][i];
235                 chromosomes[child2Idx][i] = chromosomes[motherIdx][i];
236             }
237             else
238             {
239                 chromosomes[child2Idx][i] = chromosomes[fatherIdx][i];
240                 chromosomes[child1Idx][i] = chromosomes[motherIdx][i];
241             }
242         }
243         break;
244     case CrossoverType.TwoPointCut:
245         for (int i = 0; i < numberOfGenes; i++)
246         {
247             if (i < temp_num)
248             {
249                 chromosomes[child1Idx][i] = chromosomes[fatherIdx][i];
250                 chromosomes[child2Idx][i] = chromosomes[motherIdx][i];
251             }
252             else if (temp_num < i && i < temp_num_2)
253             {
254                 chromosomes[child2Idx][i] = chromosomes[fatherIdx][i];
255                 chromosomes[child1Idx][i] = chromosomes[motherIdx][i];
256             }
257             else
258             {
259                 chromosomes[child1Idx][i] = chromosomes[fatherIdx][i];
260                 chromosomes[child2Idx][i] = chromosomes[motherIdx][i];
261             }
262         }
263         break;
264     }
265 }
266
267

```