```
switch (CrossoverOperator)
            {
                case CrossoverType.PMX:
                    // i1, i2 cut locations
                    // m[] partial map, m[i] => the mapping target of i

                    // define two crossover map for chromosome
                    index_1 = randomizer.Next(numberOfGenes);
                    while (index_2 == index_1)
                    {
                        index_2 = randomizer.Next(numberOfGenes);
                    }
                    // swap two numbers if index_1 is larger
                    if (index_2 < index_1)
                    {
                        temp_num = index_1;
                        index_1 = index_2;
                        index_2 = temp_num;
                    }
                    // initiate mapping
                    mapping = new int[numberOfGenes];
                    for (int i = 0; i < mapping.Length; i++) mapping[i] = -1;
                    // build up the mapping
                    for (int i = index_1; i < index_2; i++)
                    {

                        if (chromosomes[fatherIdx][i] == chromosomes[motherIdx][i])
                        continue;
                        if (mapping[chromosomes[fatherIdx][i]] == -1 &&
                        mapping[chromosomes[motherIdx][i]] == -1)
                        {
                            mapping[chromosomes[fatherIdx][i]] =
                            chromosomes[motherIdx][i];
                            mapping[chromosomes[motherIdx][i]] =
                            chromosomes[fatherIdx][i];
                        }
                        else if (mapping[chromosomes[fatherIdx][i]] == -1)
                        {
                            mapping[chromosomes[fatherIdx][i]] =
                            mapping[chromosomes[motherIdx][i]];
                            try
                            {
                                mapping[mapping[chromosomes[motherIdx][i]]] =
                                chromosomes[fatherIdx][i];
                            }
                            catch (System.IndexOutOfRangeException Exception)
                            {

                            }

                            mapping[chromosomes[motherIdx][i]] = -2;
                        }
                        else if (mapping[chromosomes[motherIdx][i]] == -1)
                        {
                            try
                            {
                                mapping[chromosomes[motherIdx][i]] =
                                mapping[chromosomes[fatherIdx][i]];
                            }
                            catch (System.IndexOutOfRangeException Exception) { }
                            try
                            {
                                mapping[mapping[chromosomes[fatherIdx][i]]] =
                                chromosomes[motherIdx][i];
                            }
                            catch (System.IndexOutOfRangeException Exception) { }
                            try
                            {
                                mapping[chromosomes[fatherIdx][i]] = -2;
```

```
                        }
                        catch (System.IndexOutOfRangeException Exception) { }
                    }
                    else
                    {
                        try
                        {
                            mapping[mapping[chromosomes[motherIdx][i]]] =
                            mapping[chromosomes[fatherIdx][i]];
                        }
                        catch (System.IndexOutOfRangeException Exception) { }

                        try
                        {
                            mapping[mapping[chromosomes[fatherIdx][i]]] =
                            mapping[chromosomes[motherIdx][i]];
                        }
                        catch (System.IndexOutOfRangeException Exception) { }


                        mapping[chromosomes[fatherIdx][i]] = -3;
                        mapping[chromosomes[motherIdx][i]] = -3;
                    }
                }

                // crossover and make two children
                for (int i = 0; i < numberOfGenes; i++)
                {
                    if (index_1 <= i && i < index_2)
                    {
                        chromosomes[child1Idx][i] = chromosomes[motherIdx][i];
                        chromosomes[child2Idx][i] = chromosomes[fatherIdx][i];
                    }
                    else
                    {
                        if (mapping[chromosomes[fatherIdx][i]] < 0)
                        chromosomes[child1Idx][i] = chromosomes[fatherIdx][i];
                        else chromosomes[child1Idx][i] =
                        mapping[chromosomes[fatherIdx][i]];

                        if (mapping[chromosomes[motherIdx][i]] < 0)
                        chromosomes[child2Idx][i] = chromosomes[motherIdx][i];
                        else chromosomes[child2Idx][i] =
                        mapping[chromosomes[motherIdx][i]];
                    }
                }
                // crossover finished.

                // check if all gene are distinct, if not let other chromosome
                replace it
                for (int i = 0; i < populationSize * 3; i++)
                {
                    temp = new int[numberOfGenes];
                    for (int j = 0; j < numberOfGenes; j++)
                    {
                        temp[j] = j;
                    }

                    if (chromosomes[i].Distinct().ToArray().Count() != numberOfGenes)
                    {
                        chromosomes[i] = temp.OrderBy(x =>
                        randomizer.Next()).ToArray();
                    }
                    else
                    {
                        break;
                    }
                }

```

```
123                     break;
124                 case CrossoverType.OX:
125                     // order crossover
126                     // define two crossover map for chromosome
127                     index_1 = randomizer.Next(numberOfGenes);
128                     while (index_2 == index_1)
129                     {
130                         index_2 = randomizer.Next(numberOfGenes);
131                     }
132                     // swap two numbers if index_1 is larger
133                     if (index_2 < index_1)
134                     {
135                         temp_num = index_1;
136                         index_1 = index_2;
137                         index_2 = temp_num;
138                     }
139                     // start creating crossover children
140                     temp_num = 0;
141                     temp_num_2 = 0;
142                     for (int i = index_1; i < index_2; i++)
143                     {
144                         chromosomes[child1Idx][i] = chromosomes[fatherIdx][i];
145                         chromosomes[child2Idx][i] = chromosomes[motherIdx][i];
146                     }
147                     for (int i = 0; i < numberOfGenes; i++)
148                     {
149                         if
                            (!(chromosomes[child1Idx].Contains(chromosomes[motherIdx][i])))
150                         {
151                             if (temp_num == index_1) temp_num = index_2;
152                             chromosomes[child1Idx][temp_num] = chromosomes[motherIdx][i];
153                             temp_num += 1;
154                         }
155                         if
                            (!(chromosomes[child2Idx].Contains(chromosomes[fatherIdx][i])))
156                         {
157                             if (temp_num_2 == index_1) temp_num_2 = index_2;
158                             chromosomes[child2Idx][temp_num] = chromosomes[fatherIdx][i];
159                             temp_num_2 += 1;
160                         }
161                     }
162                     break;
163                 case CrossoverType.POX:
164                     // position-based crossover
165                     temp_num = (int)Math.Round(numberOfGenes * crossoverRate);
166                     temp_1 = new int[temp_num];
167                     // create random index array
168                     for (int i = 0; i < temp_num; i++)
169                     {
170                         temp_1[i] = randomizer.Next(numberOfGenes);
171                     }
172                     // crossover children
173                     for (int i = 0; i < numberOfGenes; i++)
174                     {
175                         if (temp_1.Contains(i))
176                         {
177                             chromosomes[child1Idx][i] = chromosomes[fatherIdx][i];
178                             chromosomes[child2Idx][i] = chromosomes[motherIdx][i];
179                         }
180                         else
181                         {
182                             chromosomes[child1Idx][i] = chromosomes[motherIdx][i];
183                             chromosomes[child2Idx][i] = chromosomes[fatherIdx][i];
184                         }
185
186                     }
187                     break;
188                 case CrossoverType.OSS:
189                     // order-based crossover
```

```
                    temp_num = (int)Math.Round(numberOfGenes * crossoverRate);
                    temp_1 = new int[temp_num];
                    // create random value array
                    for (int i = 0; i < temp_num; i++)
                    {
                        temp_1[i] = randomizer.Next(numberOfGenes);
                    }
                    // crossover children
                    for (int i = 0; i < numberOfGenes; i++)
                    {
                        if (!(temp_1.Contains(chromosomes[fatherIdx][i])))
                        {
                            chromosomes[child1Idx][i] = chromosomes[motherIdx][i];
                        }
                        else {
                            chromosomes[child1Idx][i] = chromosomes[fatherIdx][i];
                        }

                        if (!(temp_1.Contains(chromosomes[motherIdx][i])))
                        {
                            chromosomes[child2Idx][i] = chromosomes[fatherIdx][i];
                        }
                        else
                        {
                            chromosomes[child2Idx][i] = chromosomes[motherIdx][i];
                        }
                    }

                    break;
                case CrossoverType.OCCC:
                    break;
            }
```