CROSSOVERS

Contents

1.1. Alternating-position Crossover

Parameters
none

Alternating-position crossover (AP) is a crossover operator creating an offspring by inserting alternately: elements of the first parent and elements of the second parent, element by element. If a given element already exists in the offspring, its reinsertion is omitted.

Alternating-position crossover is used for ordered chromosome-based problems (please make sure that genes of the individuals do not repeat).

Example

Parent 1:

A	F	B	E	C	D

Parent 2:

C	A	D	B	F	E

Offspring 1:

The first offspring is created starting from the first element of the first parent (A). Then first element of the second parent is inserted (C), preceding insertion of the next element of the first parent (F). The next element of the second parent (A) is omitted as it is already present in the offspring so (B) is derived from the first parent, and so on:

A	C	F	B	D	E

Offspring 2:

The second offspring is created starting from the second parent and following the path as described:

C	A	F	D	B	E

1.2. Cut and Splice Crossover

Parameters
none

Cut and splice crossover is a crossover operator creating an offspring by cutting both parents at random points along their lengths. The cutting point may differ for each parent, therefore the operation may result in offsprings of different lengths.

In the case of out-of-range chromosome lengths the script may fail. It is recommended to use one-point crossover instead (see: chapter 1.4).

Example

Parent 1:

1	0	1	1	1	0

Parent 2:

1	1	0	1	0	0

Offspring 1:

The first parent is randomly divided into two parts by cutting point located to the right of its fourth element (the green part). The second parent is randomly divided between third and fourth element. Obtained parts and crossed producing new offsprings:

1	0	1	1	1	0	0

Offspring 2:

1	0	1	0	0

1.3. Cycle Crossover

Parameters
none

Cycle crossover (CX) is a crossover operator creating an offspring starting from the first element in one of the parents. Cycles are then found. One cycle is a group of elements selected by the following rule:

Two parents X and Y are assumed, each one containing n elements. Therefore, single elements of the parents can be noted as X_i and Y_i where $i = 1, 2, 3, \dots, n$. It is also assumed that the cycle starts from the first parent:

1.3.1. First element of the first parent is selected: X_1,
1.3.2. Second parent is then inspected for the associated value at index $i = 1$: Y_1,
1.3.3. First parent is then analyzed and Y_1 value is looked for in the parent, found at index a: X_a,
1.3.4. Second parent is then inspected again for the associated value at index $i = a$: Y_a,
1.3.5. **If $Y_a = X_1$ cycle is completed**. If not, steps 1.3.3 to 1.3.5 are repeated, but this time considering Y_a, instead of Y_1. The cycle building process runs until $Y_i = X_1$.

After completing the first cycle, next cycles are built, starting from the first element of the first parent that is not included in any of already completed cycles.

Finally, operations are performed alternately for cycles: their elements are maintaining the origin or getting swapped between parents. Cycle crossover is used for ordered chromosome-based problems (please make sure that genes of the individuals do not repeat).

Example

Parent 1:

A	F	B	E	C	D

Parent 2:

C	A	D	B	F	E

The initial cycle starts from the first element of the first parent (**A**), $i = 1$. First element of the second parent is **C**, hence **C** is looked for in the first parent, found at $i = 5$. Fifth element of the second parent is **F**. Since **F** is not equal to the starting value (**A**), the cycle is still running. **F** is looked for in the first parent and found at $i = 2$. The second element of the second parent is **A**, matching the starting value and closing the cycle (elements of the full cycle are marked on green):

🟩 A	🟩 F	B	E	🟩 C	D

🟩 C	🟩 A	D	B	🟩 F	E

Second cycle is then looked for starting from the first out-of-cycle element in the first parent, here: **B**, $i = 3$. The second cycle completed is shown on green below:

A	F	🟩 B	🟩 E	C	🟩 D

C	A	🟩 D	🟩 B	F	🟩 E

All elements of the parents are now included in the cycles. No-swapping and swapping between parents are performed alternately, providing new offsprings as follows:

A	F	D	B	C	E

C	A	B	E	F	D

1.4. One-point Crossover

Parameters	
swapPoint	*Fixed cutting point along the chromosome length.*

One-point crossover, also known as a single-point crossover, is a crossover operator creating an offspring by cutting both parents at a point along their length (*swapPoint*). The cutting point is the same for both parents, therefore, after crossover, when elements are swapped between parents, offsprings maintain their length.

Cutting point is inserted right after i-th element, considering indices of elements starting from index $i = 1$.

Example

Parent 1:

1	0	1	1	1	0

Parent 2:

1	1	0	1	0	0

Offspring 1:

The *swapPoint* was set as **4**, so first four elements are taken to the first part (marked on green). These elements are passed from the first parent to the first offspring. The rest of elements is inherited from the second parent:

1	0	1	1	0	0

Offspring 2:

The *swapPoint* was set as **4**, so first four elements of the second parent are passed to the second offspring (marked on green). The rest of elements is inherited from the first parent:

1	1	0	1	1	0

1.5. Order Based Crossover

Parameters
none

Order based crossover (OX2) is a modification of an ordered crossover method (OX1), see: chapter 1.6. At the beginning, a random number R is obtained in range between 1 and length of a parent, denoting number of elements to be crossed between the parents. Then, R random indices are selected for the crossover. As opposite to the OX1 method, selected elements do not need to form a continuous swath within the parent.

Elements values are then checked at the selected indices in the first parent. The same values are looked for in the second parent. All other values in the second parent remain unchanged. Order of the reset of elements is changed to reflect the order in the first parent.

Second offspring is created by the same rule.

Order based and ordered crossovers are used for ordered chromosome-based problems (please make sure that genes of the individuals do not repeat).

Example

Parent 1:

A	F	B	E	C	D

Parent 2:

C	A	D	B	F	E

Offspring 1:

Let's assume **3** elements were chosen to be crossed. Three random indices were then obtained: $i = 2, 3$ and 5. Looking at the first parent, elements at these indices are: **F**, **B** and **C**, respectively. Looking at the second parent, these elements are found at indices: $i = 5, 4$ and 1, respectively. The rest of elements in the second parent is passed to the first offspring with no change (these are marked on green below). Indices 1, 4 and 5 are then filled with elements from the first parent with **no order changed**, i.e. **F**, **B** and **C** in turn:

F	A	D	B	C	E

Offspring 2:

Random indices used for evaluating the first offspring do not change, i.e. $i = 2, 3$ and 5 are investigated. Looking at the second parent, these indices are occupied by elements: **A**, **D** and **F**, respectively. Looking at the first parent, these elements are found at indices: $i = 1, 6$ and 2, respectively. The rest of elements in the first parent is passed to the second offspring with no change (these are marked on green below). Indices 1, 2 and 6 are then filled with elements from the second parent with **no order changed**, i.e. **A**, **D** and **F** in turn:

A	D	B	E	C	F

1.6. Ordered Crossover

Parameters
none

Ordered crossover (also known as: order crossover, OX1), along with order based crossover (OX2), divides a chromosome into a few parts. These parts are shuffled but, generally, order of elements within each of the parts remains unchanged. The order based crossover (see: chapter 1.5) is a modified version of the ordered crossover.

At first, cutting points are drawn, dividing length of each parent into three parts. Division of both parents does not differ. Elements between the cutting points are passed to the offsprings that the middle part of the first parent is transferred to the first offspring and the middle part of the second parent is transferred to the second offspring, maintaining their original indices. Now copying elements between parents and offspring is performed. The remaining elements of the first parent are copied to the second offspring and the remaining elements of the second parent are copied to the first offspring.

Copying is done in specific order – at first elements to the right of the second cutting point are copied to the right of the middle partition, previously inserted. Repeated elements are omitted. When the end of the parent is reached, elements from its begging are copied in order. Also, when the end of the offspring is reached, successively copied elements are inserted starting from the beginning of the offspring.

Ordered and order based crossovers are used for ordered chromosome-based problems (please make sure that genes of the individuals do not repeat).

Example

Parent 1:

A	F	B	E	C	D

Parent 2:

C	A	D	B	F	E

Offspring 1:

Two cutting points were obtained at random: between first and second element and between fourth and fifth element. Following elements of the first parent are surrounded by the cutting points and will be transferred directly to the first offspring: **F**, **B** and **E** (marked on green below).

Analyzing second parent, **A**, **D** and **B** will be passed to the second offspring. Also from the second parent, starting from the right side of the end cutting point, following elements are considered and copied to the first offspring, if not repeated: F will not be copied as it is already transferred (marked on green); same with E and that is where the end of the second parent is reached. Now, starting from beginning of the parent, following elements are copied to the first offspring in order: **C**, **A** and **D**. The last element – **D** – is inserted at the first index of the offspring as **C** and **A** filled the last two slots to the right of the second cutting point:

D	F	B	E	C	A

Offspring 2:

As mentioned above, **A**, **D** and **B** – elements surrounded by the cutting points in the second parent – are already passed to the second offspring (marked on green). Starting from the right of the end cutting point, empty slots are filled with next elements of the first parent. Repeated elements, already existent in the offspring, are omitted. Therefore, **C** is copied to the offspring as it is not present in the offspring yet, D is omitted, A is omitted, **F** is copied and B is omitted. Finally, **E** is copied to the first slot in the offspring as its right edge was reached:

E	A	D	B	C	F

1.7. Partially-mapped Crossover

Parameters
none

Partially-mapped crossover (**PMX**) was originally proposed by Goldberg and Linge in 1985. At first, two random cutting points are selected, dividing both parents into three parts. When offsprings are created, middle parts are passed to offsprings with no swapping between offsprings and maintaining their original indices – the first offsprings receives the middle part of the first parent and the second offspring receives the middle part of the second parent.

Free offspring slots are then filled with remaining elements transferred directly from their counter-parents. No index shuffling occurs as long as elements in the offspring do not repeat. In case of repetition, a mapping function is used to convert a repeated element to a unique one. The mapping function is described in the example below.

Partially-mapped crossover is used for ordered chromosome-based problems (please make sure that genes of the individuals do not repeat).

Example

Parent 1:

A	F	B	E	C	D

Parent 2:

C	A	D	B	F	E

Offspring 1:

Two cutting points were obtained at random: between first and second element and between fourth and fifth element. Following elements of the first parent are surrounded by the cutting points and will be transferred directly to the first offspring: **F**, **B** and **E** (marked on green below). Analyzing second parent, **A**, **D** and **B** will be passed to the second offspring.

A mapping function can be now defined between associated elements of two offsprings:

	F	B	E		
	↓	↓	↓		
	A	D	B		

Therefore, F is mapped into A, B is mapped into D and E is mapped into B:

$$F \longrightarrow A,\ B \longrightarrow D,\ E \longrightarrow B$$

If a repetition occurs, the repeated element is mapped into new element (e.g. E is changed for B). If the repetition still occurs, the new value is mapped again by the next conversion (in this case: B is changed for D).

In the case of the first offspring, filling the first slot (C) does not generate any repetition. Fifth and sixth slots would be filled with F and E, respectively, but since in both cases repetitions occur, their values are mapped into: F ⟶ A and E ⟶ B ⟶ D, respectively:

C	F	B	E	A	D

Offspring 2:

A, D and B elements are already inserted to the second offspring, as mentioned above. The mapping function is now inverted:

$$A \longrightarrow F,\ D \longrightarrow B,\ B \longrightarrow E$$

In this case, since first insertion in the remaining part leads to repetition of element A, mapping to F is made. Considering fifth slot, C may be copied with no change as no repetition occurs. The last element would be D that generates another repetition, so mapping to B is carried out. Since B is the part of the offspring as well, next conversion was made: D ⟶ B ⟶ E. Finally the offspring consists of:

F	A	D	B	C	E

1.8. Position Based Crossover

Parameters
none

Position based crossover (POS) is similar to order based crossover (OX2) described in chapter 1.5, but in this case, instead of the order of elements, their positions are maintained.

At beginning of the operation a random number R is obtained in range between 1 and length of the parents, obtaining number of elements to be crossed between them. Then, R random indices are selected for the crossover.

Elements values are checked at the selected indices in the second parent and passed directly to the first offspring with no positions changed. For now, all elements in the first parent are investigated, starting from its left edge, and inserted one by one to free slots of the first offspring. If repetition occurs, insertion is omitted.

Similarly, in the case of the second offspring, values at the selected indices are passed directly from the first parent with no positions changed. Then, all elements in second parent are investigated, starting from its left edge, and inserted one by one to free slots of the second offspring. If repetition occurs, insertion is omitted.

Position based crossover is used for ordered chromosome-based problems (please make sure that genes of the individuals do not repeat).

Example

Parent 1:

A	F	B	E	C	D

Parent 2:

C	A	D	B	F	E

Offspring 1:

Let's assume **3** elements were chosen to be crossed. Three random indices were then obtained: $i = 2, 3$ and 5. Looking at the second parent, elements at these indices are: A, D and F, respectively. These elements are passed directly do the first offspring (marked on green).

Now, elements from the first parent are inserted into free slots in their original order, omitting already existent values: A and F are omitted as they were passed to the offspring previously, B is copied to the first free slot ($i = 1$), E is copied to the second free slot ($i = 4$) and C is copied to the last free slot ($i = 6$). D is already existent in the offspring:

B	A	D	E	F	C

Offspring 2:

Random indices used for evaluating the first offspring do not change, i.e. $i = 2, 3$ and 5 are investigated. Looking at the first parent, elements at these indices are: F, B and C, respectively. These elements are passed directly do the second offspring (marked on green).

Now, elements from the second parent are inserted into free slots in their original order, omitting already existent values: C is omitted as it was passed to the offspring previously, A is copied to the first free slot ($i = 1$), D is copied to the second free slot ($i = 4$), B and F are already in the offspring, so finally E is copied to the last free slot ($i = 6$):

A	F	B	D	C	E

1.9. Three Parent Crossover

Parameters
none

Three parent crossover is an operator choosing three random parents out of input generation and producing one offspring. Each element of the first parent is compared with the element of the second parent. If elements are the same, the element is taken for the offspring. Otherwise, the element from the third parent is taken for the offspring.

In general, since offsprings' population size should be equal to the parent's population size, missing elements are reinserted. So far elitist reinsertion method is adopted in the solution, meaning that x best parents are directly taken to fill the gap in the offsprings' population.

Example

Parent 1:

1	0	1	1	1	0

Parent 2:

1	1	0	1	0	0

Parent 3:

1	0	1	0	0	1

Offspring:

Comparing first elements of the first and second parents, equality can be identified, so the element (1) is taken for the offspring. Same rule applies to fourth and sixth elements – 1 and 0 are taken, respectively. In the case of the rest of indices, elements of the first parent are not equal to elements of the second parents, so all the remaining slots are filled with elements of the third parent:

1	0	1	1	0	0

1.10. Two-point Crossover

Parameters	
swapStartPoint	*Fixed initial cutting point along the chromosome length.*
swapEndPoint	*Fixed end cutting point along the chromosome length.*

Two-point crossover is a modification of the one-point crossover. An offspring is created by cutting both parents at given points along their length (*swapStartPoint* and *swapEndPoint*). The cutting points are the same for both parents. The parts between the cutting points are swapped between parents, producing new offsprings. Origin of the rest of the offsprings does not change.

Cutting points are inserted right after i-th elements, considering indices of elements starting from index $i = 1$.

Example

Parent 1:

1	0	1	1	1	0

Parent 2:

1	1	0	1	0	0

Offspring 1:

The *swapStartPoint* was set as **2** and *swapEndPoint* was set as **4**, meaning that elements at third and fourth indices build the middle parts (marked on green). These parts are swapped between parents. The rest of elements is transferred directly to the offsprings, so the first offspring inherits two remaining parts of the first parent and the second offspring inherits two remaining parts of the second parent:

1	0	0	1	1	0

Offspring 2:

1	1	1	1	0	0

1.11. Uniform Crossover

Parameters	
mixProbability	*Probability of inheriting from the first parent in a parents' pair.*

Uniform crossover operates on gene-level, as opposite to the one- or two-point crossovers. The uniform crossover allows to distribute parents' genes to offsprings in accordance with the given inheritance probability (*mixProbability*).

The algorithm goes through length of parents. At each element a random uniform floating point number in range between 0.0 and 1.0 is obtained that is then compared with the given *mixProbability*. If the random number is lower or equal to *mixProbability*, the first offspring inherits the element from the first parent; otherwise, it inherits from the second parent. In the case of the second offspring, if the random number is lower or equal to *mixProbability*, it inherits from the second parent; otherwise – it inherits from the first parent.

Example

Parent 1:

1	0	1	1	1	0

Parent 2:

1	1	0	1	0	0

Offspring 1:

Both parents consist of six elements, hence six random numbers will be obtained – one number per element. It is assumed the *mixProbability* value is 0.50. At the first element, a value of 0.20 was drawn, so the first offspring inherits from the first parent and the second offspring inherits from the second parent. At the second element, a value of 0.30 was drawn, hence no swapping occurs again. Same with third and sixth elements. At fourth and fifth elements, a greater value was obtained in comparison with *mixProbability*, hence inducing gene swaps:

1	0	1	1	0	0

Offspring 2:

1	1	0	1	1	0

1.12. Voting Recombination Crossover

Parameters	
parentsNumber	*Number of parents selected for a single crossover process.*
threshold	*Minimal number of gene occurrences to inherit by the offspring.*

Voting Recombination Crossover (**VR**) is a crossover operator basing on defined number of parents selected from the population. The number of parents inspected within the crossover structure is given as *parentsNumber* and cannot be lower than 2.

Then, considering each index along the chromosome length, the most common element value at the given index is obtained. If the number of element value occurrence is equal or greater than *threshold*, the value is passed to the offspring. This operation is performed for all indices in the chromosome. If threshold limit was not reached at a given index, it is filled with element from the first parent, processed additionally by a uniform mutation operator. It should be noted the *threshold* parameter cannot be greater than *parentsNumber*.

One offspring is produced by this operator. In general, since offsprings' population size should be equal to the parent's population size, missing elements are reinserted. So far elitist reinsertion method is adopted in the solution, meaning that x best parents are directly taken to fill the gap in the offsprings' population.

Example

We assume that the number of parents selected for the crossover (*parentsNumber*) is **4**. The threshold value was set as **3**.

Parent 1:

1	0	1	1	1	0

Parent 2:

1	1	0	1	0	0

Parent 3:

1	0	0	0	1	1

Parent 4:

1	0	0	1	0	1

Offspring:

Starting from the first index, element in the first parent is **1**, in the second parent – **1**, in the third parent – **1** and in the fourth parent – **1** as well. Since number of occurrences is higher than the defined threshold, the value of **1** is passed to the offspring. At the second index, most common value among the parents is **0** – three occurrences. Since number of **0**'s is equal to the threshold, it is also passed to the offspring. Same for third and fourth index. All of these are marked on green below.

For the last two indices, $i = 5$ and **6**, no required number of occurrences was reached, hence **1** and **0** are taken from the first parent. Due to uniform mutation imposed on these values, final genes may be slightly different:

1	0	0	1	1	1

References

Hussain, A., Muhammad, Y. S., Sajid, M. N., Hussian, I., Shoukry, A. M., Gani, S.: ***Genetic Algorithm for Traveling Salesman Problem with Modified Cycle Crossover Operator***. Computation Intelligence and Neuroscience, vol. 2017, August 2017, pp. 1 – 7.

Larrañaga, P., Kuijpers, C. M. H., Murga, R. H., Inza, I., Dizdarevic, S.: ***Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators***. Artificial Intelligence Review, vol. 13, issue 2, April 1999, pp. 129 – 170.

Umbarkar, A. J., Sheth, P. D.: ***Crossover Operators in Genetic Algorithms: A Review***. ICTACT Journal on Soft Computing, vol. 6, issue 1, October 2015, pp. 1083 – 1092.