

REPRESENTING THE PROBLEM DOMAIN IN STOCHASTIC SEARCH ALGORITHMS

Lifu Zhang and John Miles, Cardiff University

Address

MilesJC@cf.ac.uk

Abstract: Stochastic search algorithms such as genetic algorithms are being increasingly used in Engineering to find good solutions to problems. Typically, the problem is represented as a set of parameters which describe the main features of the problem and many good solutions have been discovered using this approach. However, when looking at certain classes of problems, particularly some forms of engineering design, it is possible that using a parametric representation limits the search. Any restrictions on the scope of the search have the potential to result in significantly sub-optimal solutions. Examples from topological search and other design problems are examined. Non-parametric forms of representation also allow the use of novel crossover mechanisms in genetic algorithms which can be tailored to suit the problem. These can have a significant impact on how the search progresses and the form of solution that is obtained. Examples are presented for the topological optimisation of the cross section of a beam using a voxel-based representation.

1. INTRODUCTION

Genetic algorithms...

2. REPRESENTATION...

Parameterised representation approach...

2.1. Representation...1

Parameterised representation approach...1

2.2. Representation...2

Parameterised representation approach...2

3. NOVEL CROSSOVER MECHANISMS

The voxel representation approach for plane shape optimisation frees the searching space from being restricted by the experience-oriented parameters. On one hand, it makes the search algorithms, such as GAs, much more possible to locate the global optimum. On the other hand, a relatively unlimited searching space will consequentially result in an increase in the workload for a specified optimisation task. For GAs, some novel crossover mechanisms have been developed to assist the algorithms in locating the optimum quickly and stably while maintaining the size of the searching space in order to keep the robustness.

3.1. 2D crossover with positional swap

With the voxel representation approach, we can translate the shapes into computer-manageable objects by simply assigning 1 or 0 (and more options for multi-material) to each voxel to indicate if it is “ON” or not (, and if “ON”, what material it is). The genetic operator such as crossover will then operate on these numeralized objects (chromosomes) instead of the shapes. The ready-to-use crossover methods from the genetic algorithms literature can be divided up into two categories. One is the standard crossover method, like one-point crossover based on a randomly selected breaking point, and uniform crossover based on a randomly generated crossover mask. The other is the 2D crossover method by which the operator interchanges two parent chromosomes within a random extracted continuous area as shown in Fig. 1. It should be noted that the interchanging area could be any shape to satisfy the needs of a specified task.

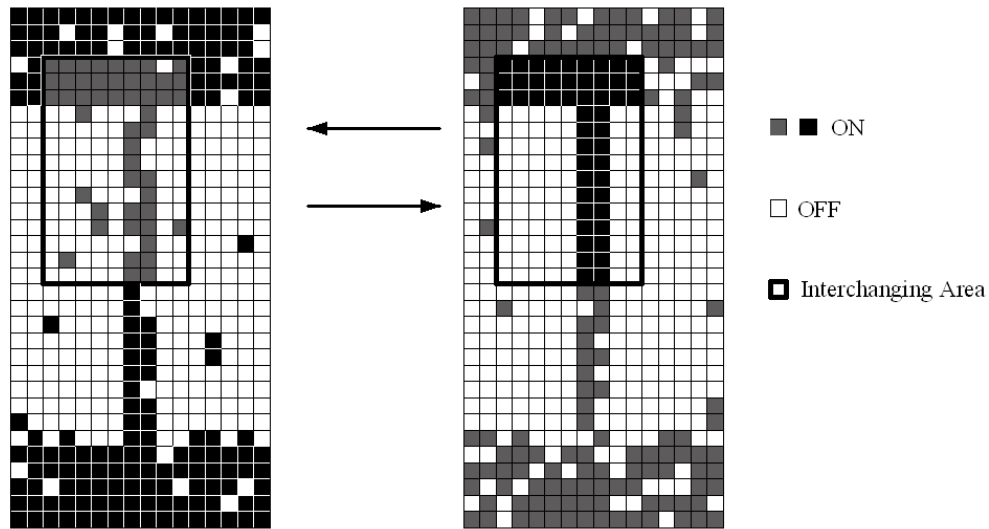


Fig. 1 2D crossover illustration

The idea is whether the numeralized object is considered a 1D *string* or a 2D *array*. Standard crossover method such as one-point crossover or uniform crossover substantially treats the object as a *string* with no regard to what it represents for as a whole. By contraries, 2D crossover emphasizes the 2D property of the object and operates on it from a 2D point of view. Both standard and 2D crossover methods have certain merits that favour the shape optimisation problems represented by voxel.

For each crossover operation, standard crossover method such as uniform crossover generates a *mask* according to which two new individuals will be produced from the parent chromosomes. The *mask* is randomly generated with the equal opportunity for every element, as the voxel within a shape, to be 0 or 1. In that case, as it shows in Fig. 2, each new individual contains “stuff” from both of its parents within a relatively short length of its body. This crossover property ensures that, when it operates on parent chromosomes, the possibility of replacing all the elements of one chromosome within a certain area with the ones from the other is quite small. That means within this certain area, a new individual will look like both of its parents. It has great advantages when we need to modify the chromosome mildly and slightly.

Parent 1: ...1 1 0 0 1 0 1 1 0 0 1... **New 1:** ...1 1 0 0 1 1 0 1 0 0 1...
Mask: ...0 1 0 0 1 1 1 1 0 1 0...
Parent 2: ...0 1 1 0 1 1 0 1 1 0 1... **New 2:** ...0 1 1 0 1 0 1 1 1 0 1...

Fig. 2 The mildness of uniform crossover

Unlike the uniform crossover method, the 2D crossover method operates much more roughly on the object. Since all of the elements within the extracted interchanging area are treated as a whole and exchanged together between parent chromosomes to form new individuals, the parents' properties of that area will be completely inherited by their children. To inherit the parents' properties in a continuous area manner is essential for shape optimisation. When we talk about the shape in most engineering cases, we mean a whatever object that is formed by countless points getting together, and it is continuous, rather than a large set of randomly spread meaningless points. The 2D crossover method emphasizes this shape characteristic and helps to maintain the good schema within the parent chromosomes. By applying 2D crossover, it has greater possibility that a new individual may inherit most of its parents' merits and hence form a much better solution. It makes an initial individual, usually in the form of randomly spread points, evolve into a relatively continuous shape, a relatively good solution much quickly.

Despite the merits that standard and 2D crossover each has, they have a big drawback in common when cooperating with voxel representation. The problem lies in the diversity of the population. The voxel representation approach can set up an extremely huge discrete searching space. Take a 32*64 voxel representation (design domain divided into 32*64 voxels) for example, the searching space is composed of 2^{2048} possible solutions. The size and discreteness of the searching space is a great problem for search algorithms including GAs. Keeping the diversity of the population becomes the most important issue. Applying GAs with a population, which has little diversity, has no difference with just throwing one blind searching robot into the Milky Way galaxy to locate the Earth. Original standard and 2D crossover method cannot help in keeping the population diversity. As a matter of fact, they just make the population converge into similar individuals too soon to locate the global optimum.

One possible way of inducing diversity into the population is mutation. Mutation do has the ability to keep the diversity; however, problems still

exist. Ordinary rate and scale is just not powerful enough and has limited contribution in the situation mentioned above, while by imply increasing the rate and scale may result in an unfavourable disturbance during the optimisation implementation. To avoid the embarrassment, a novel 2D crossover with swap method is developed. It plays the main role in inducing population diversity for voxel represented optimisation problem.

The swap is first introduced in GAs literature as a mutation operator. By swap, two elements within an object exchange their position with each other as it shows in Fig. 3.

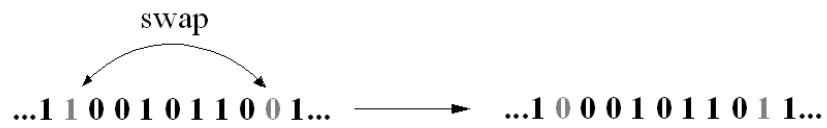


Fig. 3 Swap mutation operator

Apparently, just applying the swap mutation is not going to have any obvious effect on the problem. However, it will be different if we adopt the swap mutation idea to the 2D crossover. Fig. 4 illustrates the idea of how to integrate the swap into the 2D crossover method.

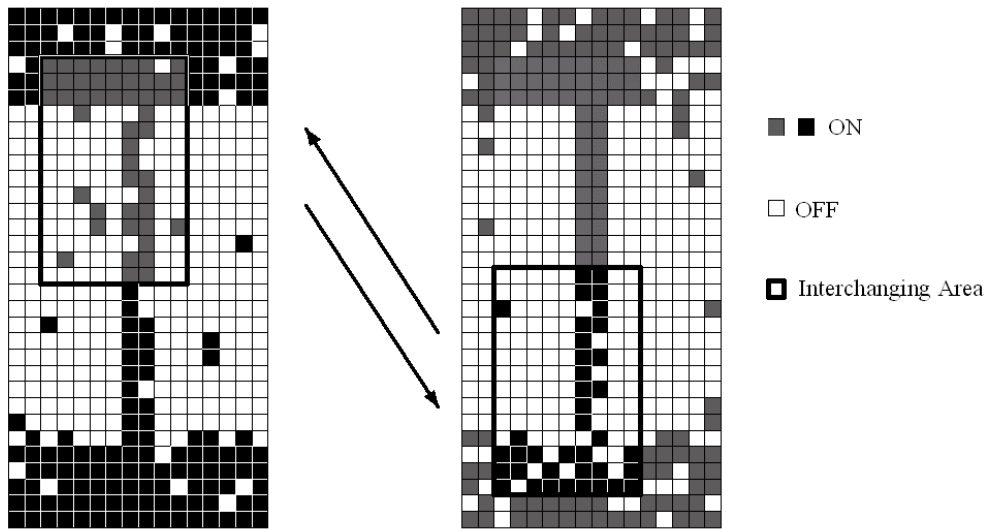


Fig. 4 2D crossover with swap

With the idea of doing element swap during crossover, we randomly abstract the interchanging area within each parent chromosome respectively. That means it is very likely that the voxel, which is going to replace the original one, comes from a different position of the other parent chromosome. This special element swap has three significant properties. Unlike the original swap as a mutation operator, the swap in 2D crossover is:

1. Large-scale swap: the swap happens in a relatively large number of elements rather than two. A new individual will have more chances to look much more dissimilar with its parents as well as with the other new individuals in the same generation, thereby maintain the diversity.
2. Inter-object swap: the swap of element happens between two parent chromosomes rather than one new individual as swap mutation does. This inter-object mechanism also helps to induce diversity to the population.
3. 2D swap: both of the above-mentioned properties have certain mechanism to some extent disturb the optimization process while trying to keep the population diversity. With the ability of maintaining the good schema of a certain area, 2D crossover method can prevent this mechanism from being too disturbing. The 2D property inherited from the 2D crossover is just as a valve added to the mechanism to control the disturbance to make sure the process is properly disturbed.

Original GAs mechanism constitutionally discourages population diversity. It simply gives better individuals who have higher fitness the greater chance to produce without considering the diversity problem, which makes the population converge on a few so-called good solutions, usually local optimum, too early. The early convergence diminishes the GAs' merit as global search algorithms. The 2D crossover with swap maintains the population diversity by inducing new diversity into the population. Meanwhile, the individual fitness is also considered for parent selection and the best individual of a generation is always kept in the population. Thus formed a vivid searching environment for GAs: one individual is located at the best solution ever found during the optimisation process and left all the other ones dispersedly distributed all over the searching space to search for better solutions. By this means, 2D crossover with swap ensures the GAs to locate the global optimum.

3.2. Fitting the crossover for specified problem

As has been mentioned early in this chapter, the interchanging area for 2D crossover can be whatever shape that suits the specified problem. Adopting the right interchanging area shape strategy can greatly improve GAs' performance. A simple *shape-guessing* program has been developed to demonstrate the influences of different interchanging area shapes on the optimisation process. It should be noted that a real world shape optimisation problem is usually much more than a shape-guessing problem; however, it is enough for us to look inside the different strategies through this simple problem.

The program is coded according to the voxel representation approach we discussed in this paper. The number of voxels in horizontal and vertical directions is 32 and 64 respectively. The initial population is set to 100. Four voxels for every new individual produced in each generation are going to mutate randomly.

Suppose that we are the puzzle maker and a shape shown in Fig. 5 (a) is given for the program to guess. For each fitness evaluation, only the number of correct voxel is known by the program. The program will start guessing from the randomly generated shapes, initial population, one of which may look like the one in Fig. 5 (b). For each kind of interchanging area strategy, the total number of generation, which reflects the operator's efficiency with the specified problem, will be examined.

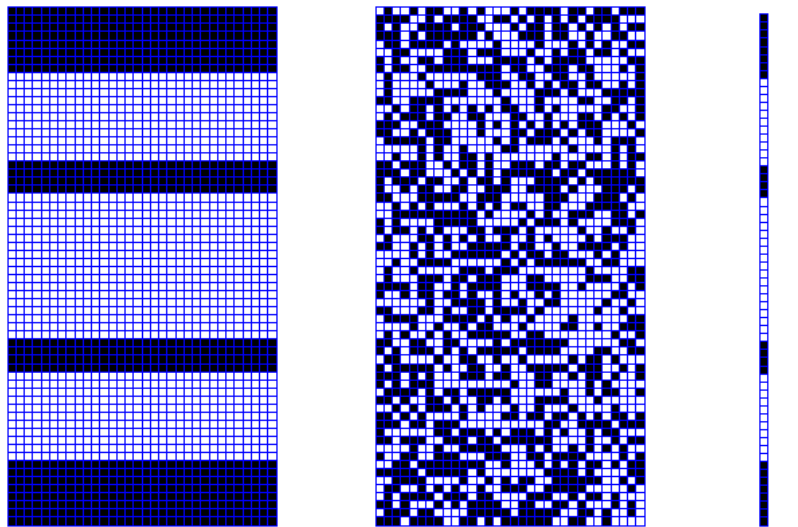


Fig. 5 (a) Puzzle shape (b) Initial Individual (c) Unit

In order to demonstrate how the crossover method could be tailored to fit a specified problem, a special 2D crossover with swap is developed. The interchanging area is specialized into a strip that runs through the domain rather than a simple rectangle. (Fig. 6) By doing this, the swap is restricted in horizontal direction.

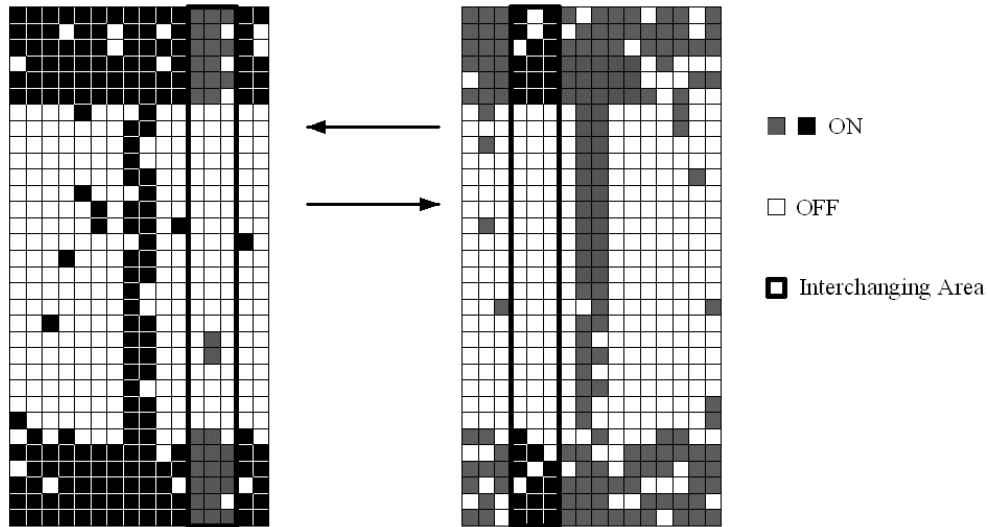


Fig. 6 Tailoring the crossover to limit the swap freeness

Table 1 shows the average number of generation (with the highest and the lowest in the parentheses) of 20 runs for each kind of crossover strategy to get the correct result. Here are the three strategies that had been tested:

- uniform crossover
- 2D crossover with swap (simple rectangle interchanging area)
- tailored 2D crossover with swap

Table 1. Average generations for different crossover strategies

Strategy	Uniform	2D	Tailored 2D
Number of generation	585 (778, 425)	>1000	215 (252, 189)

[average (highest, lowest)]

As it shows in Table 1, the tailored 2D crossover with swap performed much better than the other two in dealing with this specified problem. The uniform crossover got the right shape as well, while it took two times as

many generations as the winner did. The general 2D crossover was just unable to achieve the correct shape within 1000 generation.

Why did the tailored 2D crossover perform so well on this problem? The reason lies in the structure of the puzzle shape (Fig. 5 (a)). A characteristic unit (Fig. 5(c)) can be abstracted from the puzzle shape. The shape can be formed by arraying the unit horizontally within the domain. The tailored crossover method, which has the swap restricted horizontally, favours this property of the puzzle shape. It substantially keeps the good unit and transplants it to the other part of the domain. The uniform crossover slows down the process because it smashes the characteristic unit. The general 2D crossover is just too disrupting at the later stage of the process, which makes it unable to get the final correct answer.

3.3. Multiple crossover strategy

The problem arises when we deal with the real world shape optimisation is how we could ever know what the result will be as if we are giving the puzzle shape. In fact, we will never know beforehand what the result is or even how, for some complicated problem, it may look like. Then, how to tailor the crossover method in order to fit the problem?

Suppose that we have a set of crossover strategies in our hands, the best way to decide which one suits the problem best is to try. The intrinsic mechanism of GAs offers us a clever way of doing this. One of the most important characteristics of GAs is that it deals with a relatively large population of individuals at the same time. In that case, we can randomly apply different crossover strategies each time when GAs try to crossover two individuals. Whenever the right strategy is applied at the right time to the right individuals, a good new individual will be produced and kept in the population. We can still use the shape-guessing program to demonstrate how the multiple crossover strategy affects the optimisation process.

The puzzle shape for this test is shown in Fig. 7. This shape is composed of two characteristic units as unit 1 and unit 2. As in the former test, if we know about these beforehand, we can just use two tailored 2D crossover methods, each of which respectively restricts the swap horizontally and vertically, to achieve a good performance. Note that this is multiple crossover strategy as well. To avoid this coincidence, we use two other crossover methods, uniform and general 2D (with swap), to form the multiple crossover strategy.

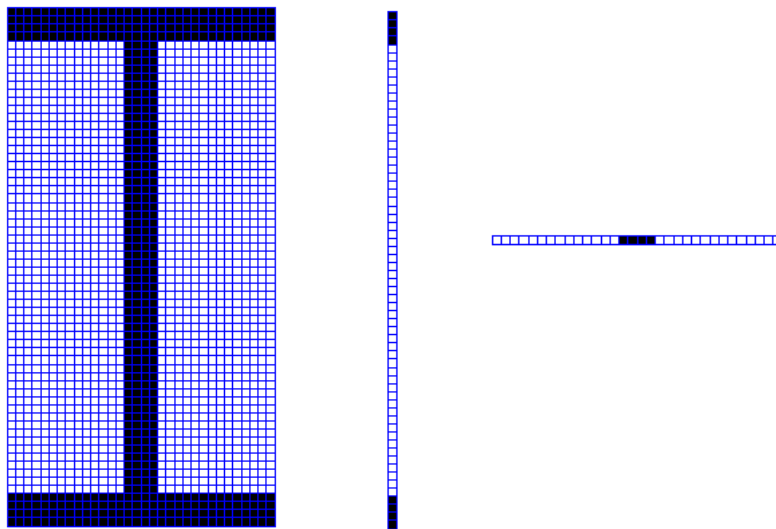


Fig. 7 Puzzle shape Unit 1 Unit 2

As well as examining the necessary number of generations to achieve the correct shape, we need to look a step deeper into the optimisation process. The best fitness of each generation is recorded in order to trace the performance all along the optimisation process and to compare the performances of different crossover methods at the same generation. Fig. 8 shows the result of the test.

Comparison of crossover strategies

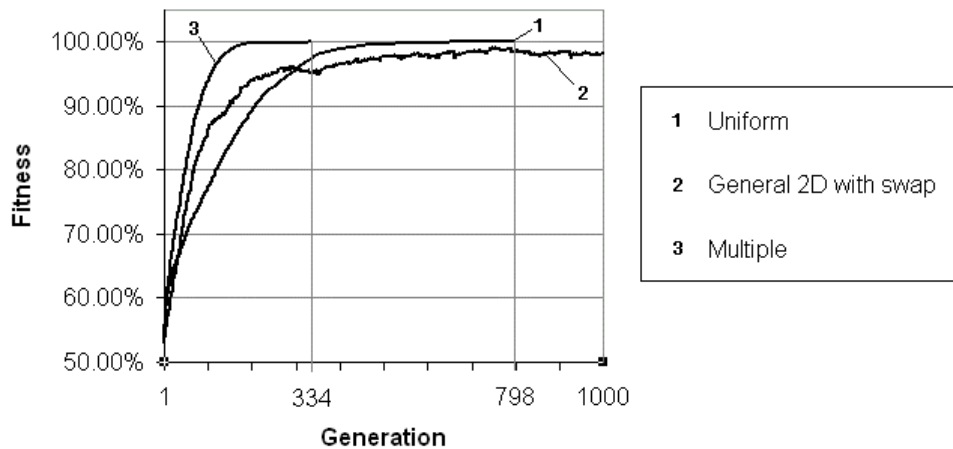


Fig. 8 Test result of multiple crossover strategy

From Fig. 8 we know that:

1. Uniform crossover method has a very stable performance. It drives the optimisation process reposefully and finally achieves the correct shape.
2. Due to its relatively disturbing nature, the general 2D crossover with swap is unable to get the perfect shape at the final phase of the whole process. However, it has an outstanding performance in the early phase. That means it can get a “good” but not “perfect” shape much quickly.
3. The multiple crossover strategy performs the best.

By “combining” two crossover methods, we form a better-performed crossover strategy with the both stable and quick property. In fact, both of the two methods are still working respectively.

The reason why the performance could be improved by multiple crossover strategy can be illustrated by the following example. Suppose that a group of people are set out looking for a target in an area and you are one of them. The area is so large that the group has to be spread around to search for the target. It is time for you to make a move. You can either jump or mince and the only information you can get is whether you become further or closer to the target after a move. Sometime you may jump because you think the target is still far away from you. This is usually a good idea at the beginning of the search because, in a large searching area, it is almost impossible for you to be very close to the target at this stage. Jumping can be allimportant when you are close to the cliff. It can save your life by jumping over it; however, you may not realize what has just happened. Whenever you jump, you are using the crossover methods like the general 2D crossover with swap we are discussing in this paper. They just give you a great change in position. Of course, jumping is not always the best choice especially when you get close to the target. It is almost impossible for you to land exactly on the target by jumping because you cannot control the distance or even the direction you jump. Now, it is time to use the uniform crossover method, to mince. By this kind of move, you will not miss the target easily. If by any chance the direction of the target is known, everybody will jump to that direction by using the tailored crossover methods. We are not always that lucky in the real world. Since you cannot control the direction for most of the time when you make the move, you always have the risk of jumping or mincing away from the target, or even worse, into the abysm. Do not be depressed! Your teammates are still there

and the best move will always be kept. Besides, you may find a new way to the target from this never-explored place, and this time, you just induce the diversity to the population.

In general, by using multiple crossover strategy, GAs becomes much cleverer when dealing with real word problem. This is especially true with multi-objective problems in which the objectives may from time to time have contradiction with each other in different ways. GAs can dexterously escape from the embarrassment by using the right crossover method that is available in the multiple crossover strategy.

Two issues are essential for the multiple crossover strategy. First, a large enough population should be provided to ensure the possibility of using the right crossover method at the right time to the right individuals. Second, the best individual must be kept for the next generation or at least being automatically selected to crossover.

4. Conclusions

Stochastic search algorithms...

References

- [1] John Miles...
- [2] ...