On the Ability of the One-Point Crossover Operator to Search the Space in Genetic Algorithms

Zbigniew Pliszka[1] and Olgierd Unold[2(✉)]

[1] Wroclaw Public Library, Sztabowa 95, 53-310 Wroclaw, Poland
[2] Department of Computer Engineering, Faculty of Electronics
Wroclaw University of Technology, Wyb. Wyspianskiego 27, 50-370 Wroclaw, Poland
olgierd.unold@pwr.edu.pl
http://olgierd.unold.staff.iiar.pwr.edu.pl/

Abstract. In this paper we study the search abilities of binary one-point crossover (1ptc) operator in a genetic algorithm (GA). We show, that under certain conditions, GA is capable of using only a 1ptc operator to explore the entire search space, fighting premature convergence. Further, we prove that to restore the entire space from any two binary chromosomes, each of length n, at least $2^{n-1}-1$ one-point crossover operations is needed. This number can serve as a measure for comparing the search speed of the different algorithms. Moreover, we propose an algorithm spanning the search space in the minimal number of crossovers.

Keywords: Evolutionary Computation · Genetic Algorithm · One-point Crossover · Premature Convergence

1 Introduction

The genetic algorithm (GA), invented in 1960s by Holland [10], seems to be one of the most studied topics in evolutionary algorithm (EA) literature. GAs are robust search and optimization algorithms based on natural selection in environments and natural genetics in biology. What is interesting, GA examines not just one solution, but a pool of probable solutions simultaneously, which are organized as chromosomes and form a population. GA incrementally generates new chromosomes by applying selection, crossover and mutation operators, until the population finally reaches a state, where diversity is minimal (so called convergence). The set of all possible chromosomes forms the search space. The most common way of encoding chromosome in GA is a fixed binary string. It is well-known that crossover operator plays a key role in the evolutionary process, especially in preserving the genetic diversity [3,31]. Various crossover operators are used in GA, and among them the simplest one is the one-point crossover belonging to so-called *mask-based crossover operators* [30]. This operator selects a crossover point within a chromosome then interchanges two parent chromosomes at this point to produce two new offspring. Note that any multi-point crossover

L. Rutkowski et al. (Eds.): ICAISC 2015, Part I, LNAI 9119, pp. 361–369, 2015.
DOI: 10.1007/978-3-319-19324-3_33

can be seen as 1ptcs assembling [22]. Moreover, in [24] we proved the theorem which says that each crossover exchanging can be represented as a composition of 1ptcs. It means, that the maximum exploration opportunities within a class of exchanging crossovers has one-point crossover, and each other type of crossover operator can retain–at best–this ability!

In this paper we show, that under certain conditions, GA is capable to span (explore) the entire search space adequately, using only one-point crossover operator.

Lets stress here and now that any evolutionary algorithm, including GA, usually tries to avoid exploring the entire space of possible solutions. However, knowing the search abilities of genetic operators is essential for the proper design of the algorithm and to prevent undesirable properties, such as a premature convergence.

Further, we propose a measure for comparing the search speed of a set by different algorithms. For a n-element set this measure is the minimal number of 1ptcs necessary to explore the entire binary space, i.e. $2^{n-1} - 1$. We show how to construct an algorithm spanning the space of binary chromosomes in the minimal number of steps. The use both of the measure and the algorithm was illustrated by the problem of finding palindromes.

2 Related Work

The issue posed by this paper has been a long time study in the field of genetic algorithms [6,29], but the problem of spanning the space of binary chromosomes seems to be still insufficiently explored in the literature and certainly prematurely abandoned.

In [21] the conditions for one-point crossover operator in GA were defined which must be met by the operator for exploring all the search space of binary chromosomes. A somewhat related problem is the problem of too early convergence of GA (convergence refers to some measure of the genetic diversity of the population). A number of attempts were undertaken to avoid this undesirable phenomenon (for a comparative survey see [19]). The authors presented a wide range of solutions. Some of them argue that sufficient operator is a mutation [9] or that such discussion is pointless without reference to particular fitness function or coding method [15]. Another group of authors proposed adaptive probabilities of genetic operators [2,4,5] or drew attention to the impact of selective pressure [1,14]. It is worth noticing a bit controversial proposal of inserting a random individual into a pool [11].

The dynamics of evolutionary algorithms expressed by the NK model, only partly related to the problem of binary search space, has been recently studied [13,32].

What is interesting, while most of the research prove the usefulness of the crossover operator in EAs [7,8,12,17,18], [26] showed the problem, called Ignoble Trails, in which mutation-only EA finds solutions much faster than using crossing-over.

Recently, the influence of crossover in multi-objective EAs is studied [16,27].

3 Some Properties of the Space of Binary Chromosomes

Subject of this study is the following set:

$A^n = \{(a_n, a_{n-1}, \ldots, a_i, \ldots, a_1) : \forall i \in \{1, 2, \ldots, n\}, a_i \in \{0, 1\}\}$.

Its elements represent all possible binary chromosomes of equal length n, where n is a natural number higher than 1. Only one-point crossover is performed on the pairs of elements of this set. This limitation is more apparent than real, however. In [22] we showed that any multi-point crossover is a combination of one-point crossovers. The proof is to be found in [24].

In the previous works [20,21,22,23,24] some definitions and properties of the binary space under study was introduced. In the following, the key definitions and results are summarized.

Definition 1. *An initial or primary population is a set of chromosomes (elements) from the A^n space.*

We assume that all elements of such a set take part in the first selection process for the parent pool.

Definition 2. *We say that the A^n space is ancestral, if all its elements can be obtained from a primary population of repeatedly applying finite number of times only one-point crossover operators.*

Note that the above definition does not reject the chromosomes received from the primary population by the other genetic operations (like mutation or inversion). The only requirement is that there is a potential ability to generate all chromosomes from the space using only 1ptcs.

Definition 3. *Two chromosomes a_t and a_k in A^n are called polar chromosomes if and only if for each coordinate these chromosomes have opposite values.*

For example two chromosomes: (01100) and (10011) are polar in A^5 space. Note that the distance between polar chromosomes is constant and equals n. Some other properties of polar chromosomes were given in [23].

Theorem 1. *The whole space A^n is ancestral if and only if there are the elements in the primary population P, which have the following properties:*
for each position (locus), we have two elements from P having different (in terms of dual opposing) values.

This theorem is proven in [20] for binary Hadamard space, which is isomorphic with A^n space (see [23]).

Theorem 2 follows from Theorem 1 immediately.

Theorem 2. *If a primary population $P \subseteq A^n$ contains a pair of polar chromosomes, then the whole space A^n is an ancestral space.*

4 An Optimal Algorithm for Exploring the Space of Binary Chromosomes

Before we prove a theorem allowing us to construct an efficient (in terms of number of operations used) algorithm for searching A^n space, we introduce some definitions and designations.

The space A^n consisting of binary chromosomes generated by 1ptc from two primary chromosomes (i.e. making primary population) can be presented by a binary tree, called CT - crossover tree. In Figure 1, the binary space A^5 in the CT form is shown. In the root node of CT there are two primary chromosomes: (00000), (11111) with their phenotypes 0 and 31, respectively.

We say that the chromosome h in CT is located on the level $k \geqslant 1$ when there is $k-1$ nodes between the root node of the tree and h. The primary population of chromosomes is located on the 0 level. In the Figure 1, chromosome (00111), with phenotype 7, is situated on the level 3. There are two intermediate nodes to the root: (00011) - phenotype 3 and (00001) - phenotype 1.
The child chromosome is obtained by using one-point crossover from parent chromosomes. In the Figure 1, chromosome (11101) - phenotype 29 from the level 2 - is the child of two chromosomes (00001) - 1 and (11111) - 31 from the level 1. Affiliate chromosome (parent) is the chromosome associated with another parent chromosome to perform a crossover operation. For example, a primary population in CT is a pair of affiliate chromosomes.

Now we will prove a simple but important for our discussion theorem.

Theorem 3. *Any algorithm established to restore (span) the entire space A^n from any two chromosomes with a one-point crossover operator needs at least $2^{n-1}-1$ operations.*

Proof (of Theorem 2). Note that 1ptc operator gives always two offspring chromosomes. Let us assume (as we show below this is a realistic assumption) that each of the fresh chromosome is new (i.e. not obtained earlier) and chromosomes in the child pair differ from each other.

Since in A^n we have 2^n chromosomes and the whole population starts from two primary chromosomes, then the minimal number of 1ptcs to be performed to explore the entire space is $(2^n-2)/2 = 2^{n-1}-1$. □

It is therefore not possible to create an algorithm exploring the whole binary space A^n performing less than $2^{n-1}-1$ one-point crossovers.

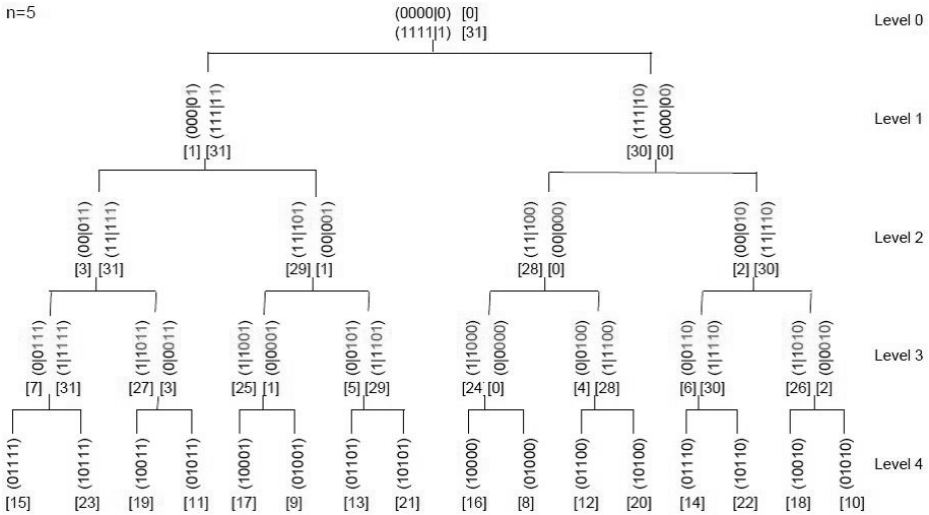

Fig. 1. Exemplary binary space A^5 with the primary population (00000) and (11111). Square brackets are used to denote a phenotype of chromosome.

Listing 1.1. Algorithm CT122

```
Input:
n                    // size of the space under study
hr0                  // input chromosome
Begin
 Tree(1,1,0) = hr0
 Tree(1,2,0) = 2^n-1-Tree(1,1,0)
 For c = 1 To n-1
  Begin //c
   For j = 1 To 2^(c-1)
    Begin //j
     Tree(2*j-1,1,c) = Tree(j,1,c-1) -
                       (Tree(j,1,c-1)Mod(2^c)) +
                       (Tree(j,2,c-1)Mod(2^c))
     Tree(2*j,1,c) = Tree(j,2,c-1) -
                     (Tree(j,2,c-1)Mod(2^c)) +
                     (Tree(j,1,c-1)Mod(2^c))
     If c < n-1 Then
      Begin //If
       Tree(2*j-1,2,c) = Tree(j,2,c-1)
       Tree(2*j,2,c) = Tree(j,1,c-1)
      End   //If
    End //j
  End //c
End.
```

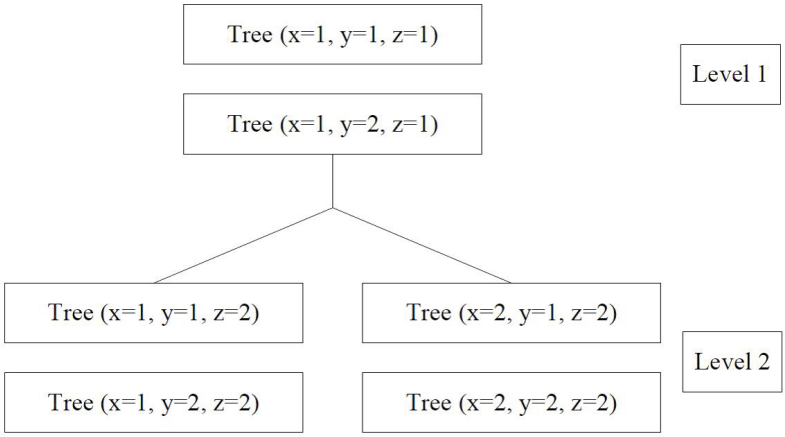

Fig. 2. *Tree* notation for a node in a crossover tree

We now present the algorithm, which reconstructs the space A^n in exactly $2^{n-1} - 1$ one-point crossovers. In the algorithm presented below (Listing 1.1) and called CT122 (Crossover Tree 1-point 2 children 2 parents algorithm), spanning binary space A^n over a binary tree, each chromosome (except the primary ones) is generated by 1ptc operation. The primary population consists of polar chromosomes. Cutting is always performed after $k+1$ allele for all chromosomes from k level, counting from the right side. The parent chromosomes in each node (except the root node) are created by affiliating the child with one of its parent chromosome, i.e. by using so called backcrossing [28].

Each node in CT122 is denoted by $Tree(x, y, z)$ (see Figure 2), where:

- x counts the nodes for a given level of CT, from the left to the right,
- y distinguishes partners for crossing operation: the value of 1 has the first parent obtained as a result of crossing on the previous level (generation), the second affiliating chromosome, taken from its parental couple, has the value of 2,
- z depicts the level of the node.

According to the above notation $Tree(x, y, 1)$ for the binary tree from Figure 1 gives the following set of chromosomes: (00001) - phenotype 1, (11111) - 31, (11110) - 30, and (00000) - 0. For the same tree, $Tree(3, 1, 4)$ denotes the chromosome (10011) with the phenotype 19.

The proof of the correctness of the Algorithm CT122 is relatively easy, and here omitted. Note that between chromosomes arranged in a tree node, occur the following properties (the reader easily sees them in Figure 1):

- a pair of the chromosomes at the same level c receives the remainders of the division by 2^c according to the rule: the first offspring and the second parent, as well as the second offspring and the first parent, have the same reminders,

- the first offspring from the level c and its first parent, as well as the second offspring from the level c and its second parent, have the same alleles from positions $c+1$ to n,
- the affiliated chromosomes coupled with a children at the level $c \leqslant n-1$ in the lines 11-16 of the CT122 Algorithm (see the restriction in the line 17) have the following properties in respect to their partners chosen in the lines 19-20:
  - equal remainders of the division by 2^c,
  - different alleles between affiliated chromosomes at the position $c+1$.

5 CT122 Algorithm in Palindrome Recognition

Now the question can be raised, what is the use of the minimal number of one-point crossovers performed to explore the entire search space, and introduced in Theorem 2? How and where can the CT122 Algorithm be used? For example, we can apply this number to compare two different algorithms searching a set.

Let us consider the apparently simple task of finding palindromes. This problem is intensively exploited in bioinformatics, e.g. to find palindromic sequences in proteins [25].

Assume, we want to find all palindromes in the space A^n. From the Theorem 2 we know without any computation, that the fastest algorithm can generate all 2^n chromosomes from A^n space in $2^{n-1}-1$ crossovers. But by knowing some properties of CT122 algorithm we are able to reduce the number of operations twice! In this case we explore two facts:

- the chromosome is a palindrome if and only if the polar-to-it chromosome is a palindrome,
- in the tree CT generated by CT122 algorithm, each polar pair of chromosomes (except by pair of polar chromosomes located in the root node at the level 0) has one each chromosome in two sub-trees, which root nodes contain chromosomes located at the level 1.

Taking into account above properties, we can determine all palindromes in A^n space, scanning two primary chromosomes and only one of the sub-tree, which in its root node has one of the child chromosome from the level 1 (in Figure 2 chromosome (00001) - phenotype 1 or (11110) - 30. It means, that we need to scan only $2+\frac{2^n-2}{2} = 2^{n-1}+1$ chromosomes, what requires $1+\frac{2^{n-1}-1-1}{2} = 2^{n-2}$ crossovers. Comparing these two algorithms, we can notice, that the second algorithm performs $\frac{2^{n-1}+1}{2^{n-2}} \cong 2$ times less operations.

6 Conclusions

The paper considered problem of exploring the binary space A^n using only one-point crossover genetic operator. The result obtained entitle us to claim, that 1ptc operator does not have to lead to a too rapid convergence. Its ability to penetrate the space is determined by two factors: starting pool and selection.

This conclusion is somewhat supported by the literature: [4,5] and especially [1] pay attention to the problem of selection, and even apparently extreme proposition of inserting a random chromosome [11], is in fact an attempt to maintain diversity of starting pool. Theorem 1 shows that if the replenishment of the current pool will be ensured by, for example, the presence of at least one pair of polar chromosomes, then the genetic algorithm can escape from the local trap. The results indicate the need to control the selection impact for reducing the possibility of exploration and exploitation properties of crossover operator.

References

1. Bäck, T.: Evolutionary Algorithms in Theory and Practice. Oxford University Press, New York (1996)
2. Choubey, N.S., Kharat, M.U.: Approaches for Handling Premature Convergence in CFG Induction Using GA. Advances in Intelligent and Soft Computing 96, 55–66 (2011)
3. Da Ronco, C.C., Benini, E.: GeDEA-II: A Simplex Crossover Based Evolutionary Algorithm Including the Genetic Diversity as Objective. Engineering Letters 21, 1 (2013)
4. Davis, L.: Adapting operator probabilities in genetic algorithms. In: International Conference on Genetic Algorithms 1989, pp. 61–69 (1989)
5. Davis, L.: Handbook of genetic algorithms. New York Van Nostrand Reinhold (1991)
6. De Jong, K.A., Spears, W.M.: A formal analysis of the role of multi-point crossover in genetic algorithms. Annals of Mathematics and Artificial Intelligence 5(1), 1–26 (1992)
7. Dietzfelbinger, M., Naudts, B., Van Hoyweghen, C., Wegener, I.: The analysis of a recombinative hill-climber on H-IFF. IEEE Transactions on Evolutionary Computation 7(5), 417–423 (2003)
8. Fischer, S., Wegener, I.: The one-dimensional Ising model: mutation versus recombination. Theoretical Computer Science 344(2-3), 208–225 (2005)
9. Fogel, D.B.: Evolving artificial intelligence. Doctoral dissertation University of California (1992)
10. Holland, J.H.: Adaptation in Natural and Artificial System. University of Michigan Press, Ann Arbor (1975)
11. Jones, T.: Crossover, macromutation and population-based search. In: Proceedings of the Sixth International Conference on Genetic Algorithms, pp. 73–80 (1995)
12. Kötzing, T., Sudholt, D., Theile, M.: How crossover helps in pseudo-boolean optimization. In: Proceedings of the 13th Annual Genetic and Evolutionary Computation Conference (GECCO 2011), Dublin, Ireland, pp. 989–996 (2011)
13. Ochoa, G., Verel, S., Daolio, F., Tomassini, M.: Local Optima Networks: A New Model of Combinatorial Fitness Landscapes. In: Recent Advances in the Theory and Application of Fitness Landscapes, pp. 233–262. Springer, Heidelberg (2014)
14. McGinley, B.: Maintaining Healthy Population Diversity Using Adaptive Crossover, Mutation, and Selection. IEEE Transactions on Evolutionary Computation 15, 692–714 (2011)
15. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer (1996)

16. Neumann, F., Theile, M.: How crossover speeds up evolutionary algorithms for the multicriteria all-pairs-shortest-path problem. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6238, pp. 667–676. Springer, Heidelberg (2010)
17. Neumann, F., Oliveto, P.S., Rudolph, G., Sudholt, D.: On the effectiveness of crossover for migration in parallel evolutionary algorithms. In: Proceedings of the 13th ACM Conference on Genetic and Evolutionary Computation (GECCO 2011), Dublin, Ireland, pp. 1587–1594 (2011)
18. Oliveto, P., He, J., Yao, X.: Analysis of population-based evolutionary algorithms for the vertex cover problem. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2008), Hong Kong, China, pp. 1563–1570 (2008)
19. Pandey, H.M., Chaudhary, A., Mehrotra, D.: A comparative review of approaches to prevent premature convergence in GA. Applied Soft Computing 24, 1047–1077 (2014)
20. Pliszka, Z., Unold, O.: Metric Properties of Populations in Artificial Immune Systems. In: Proceedings of the International Multiconference on Computer Science and Information Technology (AAIA 2010), Wisla, Poland, pp. 113–119 (2010)
21. Pliszka, Z., Unold, O.: How to predict future in a world of antibody-antigen chromosomes. In: Ganzha, M., Maciaszek, L., Paprzycki, M. (eds.) Proceedings of the Federated Conference on Computer Science and Information Systems, pp. 91–96. IEEE (2011)
22. Pliszka, Z., Unold, O.: Efficient crossover and mutation operator in genetic algorithm. Elektronika (LII), 166–170 (2011) (in Polish)
23. Pliszka, Z., Unold, O.: On some properties of binary chromosomes and states of artificial immune systems. Int. J. of Data Analysis Techniques and Strategies 4(3), 277–291 (2012)
24. Pliszka, Z., Unold, O.: On multi-individual crossing over in evolutionary algorithms. Elektronika (LV) (9/2014), 140–141 (in Polish)
25. Prasanth, N., Kirti Vaishnavi, M., Sekar, K.: An algorithm to find all palindromic sequences in proteins. Journal of Biosciences 38(1), 173–177 (2013)
26. Richter, J.N., Wright, A., Paxton, J.: Ignoble trails-where crossover is provably harmful. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 92–101. Springer, Heidelberg (2008)
27. Qian, C., Yu, Y., Zhou, Z.-H.: An analysis on recombination in multi-objective evolutionary optimization. In: Proceedings of the 13th ACM Conference on Genetic and Evolutionary Computation (GECCO 2011), Dublin, Ireland, pp. 2051–2058 (2011)
28. Schweitzer, J.A., Martinsen, G.D., Whitham, T.G.: Cottonwood hybrids gain fitness traits of both parents: a mechanism for their long-term persistence? American Journal of Botany 89(6), 981–990 (2002)
29. Spears, W.M.: Crossover or mutation? In: FOGA, pp. 221–237 (1992)
30. Syswerda, G.: Uniform crossover in genetic algorithms. In: Schaffer, J.D. (ed.) Proceedings of the International Conference on Genetic Algorithms, pp. 2–9. Morgan Kaufmann Publishers, San Mateo (1989)
31. Uy, N.Q., Hoai, N.X., O'Neill, M., McKay, R.I., Phong, D.N.: On the roles of semantic locality of crossover in genetic programming. Information Sciences 235, 195–213 (2013)
32. Verel, S., Ochoa, G., Tomassini, M.: Local optima networks of NK landscapes with neutrality. IEEE Transactions on Evolutionary Computation 15(6), 783–797 (2011)