MALE AND FEMALE CHROMOSOMES IN GENETIC ALGORITHMS

Ghodrat Moghadampour
VAMK, University of Applied Sciences, Technology and Communication, Wolffintie 30, 65200 Vaasa, Finland

Keywords: Evolutionary algorithm, Genetic algorithm, Distributional genetic algorithm, Function optimization, Male and female chromosomes, Adaptive operators.

Abstract: Evolutionary algorithms work on randomly generated populations, which are converged over runs toward the desired optima. Randomly generated populations are of different qualities based on their average fitness values. In many cases switching all bits of a randomly generated binary individual to their opposite values might quickly produce a better individual. This technique increases diversity among individuals in the population and allows exploring the search space in a more rigorous way. In this research the effect of such operation during the initialization of the population and crossover operator has been investigated. Experimentation with 44 test problems in 2200 runs showed that this technique can facilitate producing better individuals on average in around 32% of cases.

1 INTRODUCTION

Evolutionary algorithms are heuristic algorithms, which imitate the natural evolutionary process and are mainly used to solve problems which are hard to solve in conventional ways. In an evolutionary algorithm 1) problems are described by a set of parameters, 2) parameters are interpreted as a set of artificial genes, 3) genes are considered as blueprints of individuals and 4) evolution is applied to individuals (Fogel, Owens & Walsh 1966; Holland 1975; Krink 2005).

Evolutionary algorithms have typically five basic components: 1) a genetic representation of a number of solutions to the problem, 2) a way to create an initial population of solutions, 3) an evaluation function for rating solutions in terms of their "fitness", 4) "genetic" operators that alter the genetic composition of offspring during reproduction, 5) values for the parameters, e.g. population size, probabilities of applying genetic operators (Michalewicz 1996).

2 GENETIC ALGORITHMS

Most often genetic algorithms (GAs) have at least the following elements in common: one or more populations of chromosomes (solution candidates, individuals), selection according to fitness, crossover to produce new offspring, and random mutation of offspring. A simple GA works as follows: 1) A population of n l-bit strings (chromosomes) is randomly generated, 2) the fitness $f(x)$ of each chromosome x in the population is calculated, 3) chromosomes are selected to go through crossover and mutation operators with p_c and p_m probabilities respectively, 4) the old population is replace by the new one, 5) the process is continued until the termination conditions are met.

The basic concepts of GA have evolved over time and many heuristic techniques have been utilized to customize GA for specific types of problems. The studies on the usage of genetic components have been grouped into the following categories in (Talaslioglu 2009): 1) genetic operators with adjustable parameters and representation of design variables, which covers a.o. representation techniques and different operators used in GAs, 2) constraint handling for evaluation of fitness values, which for instance covers the design of penalty functions and adaptive approaches for handling the constraints.

Moghadampour G. (2010).
MALE AND FEMALE CHROMOSOMES IN GENETIC ALGORITHMS.
In *Proceedings of the 12th International Conference on Enterprise Information Systems - Artificial Intelligence and Decision Support Systems*, pages 220-225
DOI: 10.5220/0002897702200225

2.1 Problem Encoding

Genetic algorithms work on the genotype space (coding space) and the phenotype space (solution space) alternatively. The genetic operators work on the genotype space and the evaluation and selection operators work on the phenotype space. The mapping from the genotype space to the phenotype space influence the performance of genetic algorithms significantly. Natural selection is the link between chromosomes and the performance of decoded solutions (Gen & Cheng 2000).

The encoding methods can be divided to the following classes: 1) binary encoding for binary, integer and real numbers, 2) real number encoding for real numbers only, 3) integer or literal permutation encoding, 4) general data structure encoding. Binary encoding requires a decoding function and it might cause decoding anomalies (Krink 2005). The real number encoding is considered the best one for solving optimizations and constrained optimizations problems (Gen et al 2000).

2.2 Genetic Operators

For any evolutionary computation technique a representation of object variables must be chosen along with the appropriate evolutionary computation operators. For each representation, several operators might be employed (Michalewicz 2000).

The most commonly used genetic operators are crossover and mutation. However, the basic ideas of these operators have been adjusted and implemented in many different problem specific manners by many researchers. Several genetic operators have been presented by Moghadampour (Moghadampour 2006); random building block operator, integer and decimal mutation operators, variable crossover operator and variable replacement operator. In (Cervantes & Stephens 2008) it is argued that applying genetic operators with probabilities dependant on the fitness rank of a genotype or phenotype offers a robust alternative to the simple GA and avoids some questions of parameter tuning without having to introduce an explicit encoded self-adaptation mechanism.

2.2.1 Crossover

Crossover is the main distinguishing feature of a GA. In single-point crossover a single crossover position is chosen randomly and the parts of the two parents divided by the crossover position are exchanged to form two new individuals (offspring). It recombines building blocks (schemas) on different strings, but, it is "positional biased": the location of the bits in the chromosome determines the schemas that can be created or destroyed by crossover (Eshelman, Caruana & Schaffer 1989; Mitchell 1998).

In two-point crossover, two positions are chosen at random and the segments between them are exchanged. Two-point crossover reduces positional bias and endpoint effect, it is less likely to disrupt schemas with large defining lengths, and it can combine more schemas than single-point crossover (Mitchell 1998). Two-point crossover cannot combine all schemas.

Multipoint-crossover has also been implemented, e.g. in one method, the number of crossover points for each parent is chosen from a Poisson distribution whose mean is a function of the length of the chromosome.

Parameterized uniform crossover (Spears & De Jong 1991; Mitchell 1998) is also a multipoint-crossover in which each bit is exchanged with probability p ($0.5 \le p \le 0.8$) and any schemas contained at different positions in the parents can potentially be recombined in the offspring. Hence, there is no positional bias. This implies that uniform crossover can be highly disruptive of any schema and may prevent co-adapted alleles from ever forming in the population. (Mitchell 1998).

2.2.2 Mutation

The common mutation operator used in canonical genetic algorithms to manipulate binary strings $a = (a_1,...a_\ell) \in I = \{0,1\}^\ell$ of fixed length ℓ was originally introduced by Holland (1975) for general finite individual spaces $I = A_1 \times ...A_\ell$, where $A_i = \{\alpha_{i_1},...,\alpha_{i_{k_i}}\}$. By this definition, the mutation operator proceeds by:

i. determining the position $i_1,...,i_h (i_j \in \{1,...,l\})$ to undergo mutation by a uniform random choice, where each position has the same small probability p_m of undergoing mutation, independently of what happens at other positions.
ii. forming the new vector $a'_i = (a_1,...,a_{i_1-1},a'_{i_1},a_{i_1+1},...,a_{i_h-1},a'_{i_h},a_{i_h+1},...a_\ell)$, where $a'_i \in A_i$ is drawn uniformly at random from the set of admissible values at position i.

The original value a_i at a position undergoing mutation is not excluded from the random choice of $a_i' \in A_i$. This implies that although the position is chosen for mutation, the corresponding value might not change at all (Bäck, Fogel, Whitley & Angeline 2000).

Crossover is commonly viewed as the major instrument of variation and innovation in GAs, with mutation, playing a background role, insuring the population against permanent fixation at any particular locus (Mitchell 1998; Bäck, Fogel, Whitley & Angeline 2000). Mutation and crossover have the same ability for "disruption" of existing schemas, but crossover is a more robust constructor of new schemas (Spears 1993; Mitchell 1998).

While recombination involves more than one parent, mutation generally refers to the creation of a new solution form one and only one parent. Most mutation operators for permutations are related to operators, which have also been used in neighbourhood local search strategies. (Whitley 2000).

2.2.3 Other Operators and Mating Strategies

Examples of other operators used in Gas are: inversion, gene doubling and several operators for preserving diversity in the population. For instance, a "crowding" operator has been used in (De Jong 1975; Mitchell 1998) to prevent too many similar individuals ("crowds") from being in the population at the same time. This operator replaces an existing individual by a newly formed and most similar offspring. In (Mengshoel & Goldberg 2008) a probabilistic crowding niching algorithm, in which subpopulations are maintained reliably, is presented. It is argued that like the closely related deterministic crowding approach, probabilistic crowding is fast, simple, and requires no parameters beyond those of classical genetic algorithms.

The same result can be accomplished by using an explicit "fitness sharing" function (Goldberg & Smith 1987; Mitchell 1998) whose idea is to decrease each individual's fitness by an explicit increasing function of the presence of other similar population members. In some cases, this operator induces appropriate "speciation", allowing the population members to converge on several peaks in the fitness landscape (Goldberg et al. 1987; Mitchell 1998). However, the same effect could be obtained without the presence of an explicit sharing function (Smith, Forrest & Perelson 1993; Mitchell 1998).

Diversity in the population can also be promoted by putting restrictions on mating. For instance, distinct "species" tend to be formed if only sufficiently similar individuals are allowed to mate (Deb & Goldberg 1989; Mitchell 1998). Another attempt to keep the entire population as diverse as possible is disallowing mating between too similar individuals, "incest" (Eshelman 1991; Eshelman & Schaffer 1991; Mitchell 1998). Another solution is to use a "sexual selection" procedure; allowing mating only between individuals having the same "mating tags" (parts of the chromosome that identify prospective mates to one another). These tags, in principle, would also evolve to implement appropriate restrictions on new prospective mates (Holland 1975; Mitchell 1998).

Another solution is to restrict mating spatially. The population evolves on a spatial lattice, and individuals are likely to mate only with individuals in their spatial neighbourhoods. Such a scheme would help preserve diversity by maintaining spatially isolated species, with innovations largely occurring at the boundaries between species (Hillis 1992; Mitchell 1998).

3 APPLYING MALE-FEMALE PATTERN

In this research bit string c_2 is considered to be the female of bit string c_1 if the value of each locus in c_2 has the opposite value of the equivalent locus in c_1. The same idea can be expressed in the following way:

$$\forall l_1 \in c_1 \wedge \forall l_2 \in \ c_2 : l_1 = 1 \hat{} \ l_2 \qquad (1)$$

The male-female concepts were applied during population initialization at the beginning of the algorithm and during each crossover operator. In the following we go through each procedure in detail.

3.1 Initialization of the Population

Binary initialization of individuals was performed to assure the real random initialization of the population. This was simply done by randomly initializing each locus of gene in each chromosome with 0 or 1. During initialization, for each individual a mate of the opposite sex was created. This was done simply by inverting each gene in the individual to its opposite value.

The motivation for this operation was the observation that flipping all bits in a string could

lead to rapid fitness improvement. Moreover, crossover with two bit strings of opposite values increases the chance of producing better offspring. For each individual, both male and female chromosomes were evaluated. Of these chromosomes, the fitter one was set to be the male chromosome and the other one was set to be the female chromosome. It's clear that the order of male and female chromosomes could have been well vice versa.

This process was repeated until all members of the population were created. Therefore, each individual was actually presented by two chromosomes: a male chromosome and a female chromosome. However, operations were aimed at the male chromosomes by default.

After each evaluation of the population, individuals were sorted in ascending order according to their fitness values. This helped dividing the population to three separate parts: 25th percentile (lower quartile), 75th percentile (higher quartile) and interquartile range (above the lower quartile and below the higher quartile). This division was necessary in order to recognize the most critical areas in the distribution of the population and focus genetic operators on most promising individuals. This also helped implementing genetic operators more precisely and avoiding precious processing time.

It is assumed that as a result, genetic operators will be more efficient and improve the population more rapidly. This division will also help maintaining diversity in the population while for instance individuals in the higher quartile will go through continuous evolution process and improve more. Furthermore, the division of the population helps focusing evolutionary operators intentionally on certain individuals instead of hoping that a random process would take care of the process and select fitter individuals for different operators.

3.2 The Higher Quartile Crossover Operator

The higher quartile crossover (HQC) operator implements the idea of the well-known one-point crossover. However, parents are selected for breeding in a new way. For this operator, two different parents, p_1 and p_2 and a crossover point cp were randomly selected. Then, the male chromosome of parent p_1 was crossed over with the male and female chromosomes of parent p_2. As a result, four new offspring were created. These new offspring then went through the survivor selection procedure for possible replacement.

The new idea here is that the crossover operator is repeated with the same parent indexes and on the same crossover point as long as a better offspring is created. It is important to notice that the same parent indexes do not necessarily mean the same parents since a better offspring might have replaced the parent during the previous survivor selection. Crossover points were selected so that they were at least two loci far from the end points of the binary representations of the chromosomes. This was to make sure that the operator was really crossover and not mutation.

4 EXPERIMENTATION

The male and female chromosome pattern was applied as part of a genetic algorithm to solve the following minimization problems: 1) the Ackley function, 2) the Colville function, 3) the De Jong function F1, 4) the De Jong function F2, 5) the De Jong function F3, 6) the De Jong function F4, 7) the De Jong function F5, 8) the Griewank function F1, 9) the Rastrigin function, 10) the Rosenbrock function, 11) the Schaffer function F6, and 12) the Schaffer function F7.

During test runs the population size was set to 12, but no other fixed parameter value was used. During each phase operators were repeated as long as they managed to produce better offspring.

For multidimensional problems with optional number of dimensions (n), the algorithm was tested for $n = 1,\ 2,\ 3,\ 4,\ 5,\ 10,\ 20,\ 50$ and 100. These problems formed 44 cases and each one was tested in 50 runs. The efficiency of utilizing so called female chromosome in generating better individuals and improving the population fitness values was studied during the test runs. Table 1 summarizes the survival statistics of male and female chromosomes for the test functions.

The statistics indicate that using the female chromosome has resulted in generating better offspring in many cases. On average the female chromosomes resulted in generating better offspring in around 32% of test cases. Therefore it can easily be concluded that utilizing the female chromosome could help genetic algorithm in producing better individuals and accelerate the process of finding the optimal solution.

Table 1: Comparison of survival of male and female chromosomes in test runs. To save space, for multidimensional problems only cases with the number of variables equals to 2 and 100 are reported.

Function	Variables	Survival Rate in Selection (%)		Survival Rate in Crossover (%)			
				Worst Run		Best Run	
		Male	Female	Male	Female	Male	Female
Ackley	2	50	50	60	39	72	27
	100	50	50	52	47	100	0
Griewank	2	50	49	92	7	100	0
	100	0	100	81	18	0	100
Rosenbrock F1	2	50	49	100	0	100	0
	100	50	50	100	0	100	0
Rastrigin F1	2	77	22	80	20	92	8
	100	50	50	57	42	66	33
Colville	4	50	49	75	25	33	66
De Jong F1	3	50	50	-	-	-	-
De Jong F2	2	52	47	-	-	-	-
De Jong F3	5	50	49	-	-	-	-
De Jong F4	30	68	31	33	66	46	53
De Jong F5	2	49	50	78	21	33	66
Schaffer F6	2	50	50	77	22	66	33
Schaffer F7	2	50	50	44	55	33	66

5 CONCLUSIONS

In this paper male and female chromosome concepts were presented and techniques for generating and applying these patterns in practice were proposed. In addition, individuals were organized and selected for further processing in a new way. The population was ordered based on individuals' fitness values and the ones in the higher quartile of the population distribution were selected to go through the crossover operator.

Experimentation showed that the male female pattern can be useful in many cases and result in generating better individuals during the evolutionary process of genetic algorithm. However, experimentation suggests that the output of applying this technique to some extent depends on the nature of the search space and the function to be optimized. For symmetrical search spaces, i.e. the ones, which divide equally on the both sides of zero (like $-5.4 \le x \le 5.4$) creating so called female chromosome out of a male chromosome will result in creating the additive inverse of the equivalent floating-point value of the male chromosome.

This obviously will then provide an easy and fast way to explore the opposite areas of the search space. For asymmetrical search spaces creating female chromosomes will in many cases result in creating floating-point values of different magnitude, which also help exploring the search space more efficiently. Another observation with using female chromosomes was that in many cases even though the female chromosome itself didn't yield a better fitness value than its male mate, they produced better offspring than their male mates after going through genetic operators.

5.1 Future Research

The proposed pattern can be applied to new problems and its efficiency in helping the search process can be further evaluated. The pattern can also be refined and adjusted to match other types of problems.

ACKNOWLEDGEMENTS

This research is part of PhD dissertation (Moghadampour 2006), which was publicly defended on the 12th of May, 2006 at the University of Vaasa, Vaasa, Finland.

REFERENCES

Bäck, Thomas, David B. Fogel, Darrell Whitely & Peter J. Angeline (2000). *Mutation operators*. In: *Evolutionary*

Computation 1, Basic Algorithms and Operators. Eds T. Bäck, D. B. Fogel & Z. Michalewicz. United Kingdom: Institute of Physics Publishing Ltd, Bristol and Philadelphia. ISBN 0750306645.

Cervantes, Jorge & Stephens, Christopher Rhodes (2008). *Rank based variation operators for genetic algorithms*. In: *Proceedings of the 10th annual conference on Genetic and evolutionary computation*. ACM New York, NY, USA. ISBN: 978-1-60558-130-9

Deb, K. & D. E. Goldberg (1989). *An investigation of niche and species formation in genetic function optimization*. In: *Proceedings of the Third International Conference on Genetic Algorithms*. Ed. J. D. Schaffer. Morgan Kaufmann.

De Jong, K. A. (1975). *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. Ph.D. thesis, University of Michigan. Michigan: Ann Arbor.

Eshelman, L. J. (1991). *The CHC adaptive search algorithm: how to have safe search when engaging in nontraditional genetic recombination*. In: *Foundations of Genetic Algorithms*. Ed. G. Rawlins. Morgan Kaufmann.

Eshelman, L. J. & J. D. Schaffer (1991). *Preventing premature convergence in genetic algorithms by preventing incest*. In: *Proceedings of the Fourth International Conference on Genetic Algorithms*. Eds R. K. Belew & L. B. Booker. San Mateo, CA: Morgan Kaufmann Publishers.

Eshelman, L. J., R. A. Caruana & J. D. Schaffer (1989). *Biases in the crossover landscape*. In: *Proceedings of the Third International Conference on Genetic Algorithms*. Ed. J. D. Schaffer. Morgan Kaufmann.

Fogel, L. J., A. J. Owens & M. J. Walsh (1966). *Artificial Intelligence through Simulated Evolution*. Chichester, UK: John Wiley.

Goldberg, D. E. & R. E. Smith (1987). *Nonstationary function optimization using genetic algorithms with dominance and diploidy*. In: *Proceedings of The 2nd International Conference on Genetic Algorithms*, 59-68. Ed. J. J. Grefenstette. Lawrence Erlbaum Associates.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. Ann Arbor: MI: University of Michigan Press.

Gen, Mitsuo & RunWei Cheng (2000). *Genetic Algorithms and Engineering Optimization*. A Wiley-Interscience Publication. John Wiley & Sons, Inc. ISBN 0-471-31531-1.

Hillis, W. D. (1992). Co-evolving parasites improve simulated evolution as an optimization procedure. In: *Artificial Life II*. Eds C. G. Langton, C. Taylor, J. D. Farmer & S. Rasmussen. Addison-Wesley.

Krink, Thiemo (2005). *Foundations of Evolutionary Computation*, *Lecture Notes*. Available at: http:// www.daimi.au.dk/~krink/fec05/index.html. Checked in June 2005.

Lis, J. & M. Lis (1996). Self-adapting parallel genetic algorithm with the dynamic mutation probability, crossover rate and population size. In: *Proceedings of the 1st Polish National Conference on Evolutionary Computation*, 324-329. Ed. J. Arabas. Oficina Wydawnica Politechniki Warszawskiej.

Mengshoel, Ole J. & Goldberg, David E. (2008). *The crowding approach to niching in genetic algorithms*. Evolutionary Computation, Volume 16, Issue 3 (Fall 2008). ISSN:1063-6560.

Michalewicz, Zbigniew (1996). *Genetic Algorithms + Data Structures = Evolution Programs*. Third, Revised and Extended Edition. USA: Springer. ISBN 3-540-60676-9.

Michalewicz, Zbigniew (2000). *Introduction to search operators*. In: *Evolutionary Computation 1, Basic Algorithms and Operators*. Eds T. Bäck, D. B. Fogel & Z. Michalewicz. United Kingdom: Institute of Physics Publishing Ltd, Bristol and Philadelphia. ISBN 0750306645.

Michalewicz, Zbigniew & David B. Fogel (2004). *How to Solve It: Modern Heuristics.* Second, Revised and Extended Edition. Germany: Springer-Verlag Berlin Heidelberg. ISBN 3-540-22494-7.

Michalewicz, Zbigniew (1996). *Genetic Algorithms + Data Structures = Evolution Programs*. Third, Revised and Extended Edition. USA: Springer. ISBN 3-540-60676-9.

Mitchell, Melanie (1998). *An Introducton to Genetic Algorithms*. United States of America: A Bradford Book. First MIT Press Paperback Edition.

Moghadampour, Ghodrat (2006). *Genetic Algorithms, Parameter Control and Function Optimization: A New Approach*. PhD dissertation. ACTA WASAENSIA 160, Vaasa, Finland. ISBN 952-476-140-8.

Smith, R. E., S. Forrest & A. S. Perelson (1993). *Population diversity in an immune system model: implications for genetic search*. In: *Foundations of Genetic Algorithms 2*. Ed. L.D. Whitely. Morgan Kaufmann.

Spears, W. M. & K. A. De Jong (1991). *On the virtues of parametrized uniform crossover*. In: *Proceedings of the Fourth International Conference on Genetic Algorithms*. Eds R. K. Belew & L. B. Booker. Morgan Kaufmann.

Spears, W. M. (1993). Crossover or mutation? In: *Foundations of Genetic Algorithms 2*. Ed. L. D. Whitely. Morgan Kaufmann.

Talaslioglu, Tugrul (2009). A *New Genetic Algorithm Methodology for Design Optimization of Truss Structures: Bipopulation-Based Genetic Algorithm with Enhanced Interval Search*. Modelling and Simulation in Engineering archive. Volume 2009 (January 2009). ISSN: 1687-5591.

Whitley, Darrell (2000). *Permutations*. In: *Evolutionary Computation 1, Basic Algorithms and Operators*. Eds T. Bäck, D. B. Fogel & Z. Michalewicz. United Kingdom: Institute of Physics Publishing Ltd, Bristol and Philadelphia. ISBN 0750306645.