Chromosome Reuse in Genetic Algorithms

Adnan Acan and Yüce Tekol

Computer Engineering Dept., Eastern Mediterranean University, Gazimagusa, T.R.N.C. Mersin 10, TURKEY
adnan.acan@emu.edu.tr, ytekol@ieee.org

Abstract. This paper introduces a novel genetic algorithm strategy based on the reuse of chromosomes from previous generations in the creation of offspring individuals. A number of chromosomes of above-average quality, that are not utilized for recombination in the current generation, are inserted into a library called the chromosome library. The main motivation behind the chromosome reuse strategy is to trace some of the untested search directions in the recombination of potentially promising solutions. In the recombination process, chromosomes of current population are combined with the ones in the chromosome library to form a population from which offspring individuals are to be created. Chromosome library is partially updated at the end of each generation and its size is limited by a maximum value. The proposed algorithm is applied to the solution of hard numerical and combinatorial optimization problems. It outperforms the conventional genetic algorithms in all trials.

1 Introduction

Genetic algorithms (GA's) are biologically inspired search procedures that have been successfully used for the solution of hard numerical and combinatorial optimization problems. Since their introduction by John Holland in 1975, there has been a great deal on the derivation of various algorithmic alternatives of the standard implementation toward a faster and better localization of optimal solutions. In all these efforts, mechanisms of natural evolution developed over millions of years have became the main source of inspiration. The power and success of GA's is mainly achieved by the diversity of individuals of a population which evolve following the Darwinian principle of "survival of the fittest". In the standard implementation of GA's, the diversity of individuals is achieved using the genetic operators mutation and crossover which facilitate the search for high quality solutions without being trapped into local optimal points [1], [2], [3], [4].

In order to determine the most efficient ways of using GA's, many researchers have carried out extensive studies to understand several aspects such as the role and types of selection mechanism, types of chromosome representations, types and application strategies of the genetic operators, memory-based approaches, parallel implementations, and hybrid algorithms. In particular, several studies

E. Cantú-Paz et al. (Eds.): GECCO 2003, LNCS 2723, pp. 695–705, 2003.

were made concerning the development of problem-specific hybrids combining genetic algorithms with other intelligent search methods, and it has been demonstrated by thousands of applications that these approaches provide better results than the conventional genetic algorithms on very difficult problems [5], [6], [7], [8].

Among many different improvement efforts, memory-based approaches have also been studied and successfully applied for the solution difficult problems. Memory-based approaches aim to improve the learning performance of GAs by reintroducing chromosomes of previous generations into the current population, and their fundamental inspiration comes from redundancy within genetic material in natural biology and intelligent search methods which use the experience-based knowledge developed during the search to decide on the new search directions. In memory-based implementations, information stored within a memory is used to adapt the GAs behavior either in problematic cases where the solution quality is not improved over a number of iterations or to provide further directions of exploration and exploitation. Memory in GAs can be provided internally (within the population) or externally (outside the population) [9].

The most common approaches using internal memory are polyploidy structures and polygenic inheritance. Polyploidy structures in combination with dominance mechanisms use redundancy in genetic material by having more than one copy of each gene. When a chromosome is decoded to determine the corresponding phenotype, the dominant copy is chosen. By switching between copies of genes, the GA can adapt faster to changing environments and recessive genes are used to provide information about fitness values from previous generations [10], [11], [12]. Polygenic inheritance is based on the idea that a trait can depend on more than one gene or gene pair. In this case, the more gene pairs involved in the calculation of a trait, the more difficult it is to distinguish between various phenotypes. This is certainly a situation which smooths the evolution in a variable environment [13], [14].

External memory implementations store specific information and reintroduce that information into the population at a later moment. In most cases, this means that individuals from memory are put into the initial population of a new or restarted GA [15]. Case-based memory approaches, which is actually a long term elitism, is the most typical form of external memory implemented in practice. In general, there are two kinds of case-based memory implementations: in one kind, case-based memory is used to re-seed the population with the best individuals from previous generations when a change in the variable problem domain takes place [15]. A different kind of case-based memory stores both problems and solutions [16], [17]. When GA has to solve a problem similar to problems in its case-based memory, it uses the stored solutions to seed the initial population. Case-based memory aims to increase the diversity by reintroducing individuals from previous generations and achieves exploitation by reintroducing individuals from case-based memory when a restart from a good initial solution is required.

This paper introduces a novel external memory-based genetic algorithms strategy based on the reuse of chromosomes from previous generations in the creation of offspring individuals. At the end of each generation, a number of potentially promising chromosomes, based on their fitness values, are inserted into a library, called the chromosome library. Basically, starting from any point in the solution space, it is possible to form a path to an optimal solution over many different alternatives. Consequently, chromosome reuse aims to trace untested possibilities in the recombination of potentially promising solutions. Those individuals having a fitness value above a threshold, that are not used in the current recombination process, are selected for insertion into the chromosome library. During the recombination process, chromosomes of current population are combined with the ones in the chromosome library to form a population from which offspring individuals are to be created. The size of the chromosome library is limited by a maximum value and in case of excessive insertions, only the best individuals within the limits are accepted. The proposed algorithm is applied to the solution of hard numerical and combinatorial optimization problems. The obtained results demonstrate the superiority of the proposed approach over the conventional genetic algorithms.

The idea of reusing some chromosomes of previous generations, in the formation of offspring individuals, arises from a well-known fact in intelligent search algorithms: a search process has to make frequent backtracks or restarts to find a path to an optimal solution [18], [19]. This is because, an alternative search direction that may not be seen attractive at some point, due to more promising alternatives or due to many alternatives, may provide a link to an optimal solution with smaller number of computational steps. This idea is illustrated with a simple example as follows:

Assume that we want to maximize the objective function $f(x) = x^2$, $x \in [0, 1]$, using 8-bit binary encoding. Certainly, $f(x)$ takes its maximum value for an individual $p* = 11111111$. Now, consider the following individuals $p1 = 00011111$, $p2 = 11100000$, and $p3 = 10000001$. Due to fitness-based selection procedures, it is obvious that $p2$ and $p3$ will produce much more offspring than $p1$ for the next generation. In addition to that, the number of recombinations between $p2$ and $p3$ will be greater than the ones between $p1$ and $p2$, and between $p1$ and $p3$. However, as can be seen from the structures of $p1$ and $p2$, a one-point crossover between the two from the position $j = 4$ will produce the optimal solution. Hence, it is worth to store chromosomes like $p1$ for a while to give them a chance for recombination with high quality individuals to provide a shorter path to an optimal solution. It is also important to note that, the individuals like $p1$ which can be accessed from the chromosome library are computationally free because their structure and fitness values are known from previous generations.

As explained with the above particular examples, in the recombination of two potential solutions, there are lots of possibilities and only a few of them are randomly tried due to restrictions of fitness-based selection procedures and the population size. In fact, for a binary encoding of length l, two individuals can be recombined in $(l - 1)$ different ways using the 1-point crossover. The number

of offspring that can be produced with 2-point crossover is $l(l-1)$, whereas this number using the uniform crossover is 2^k, $k \leq l$, where k is the number of positions where the two parents are different. Obviously, since the individuals of the current generation are completely replaced by their offspring, there is no way to retry another recombination operation with these individuals unless they are reproduced in future generations. In theoretical models of genetic algorithms, the branching process in genetic evolutionary search is explained by the schema theorem which is based on hyperplane sampling where the convergence process is modelled by increasingly more frequent sampling from high fitness individuals by crossover and mutation acting as a background operator to prevent premature convergence. In this respect, the use of the chromosome library will help the search process by providing additional intensification and diversification alternatives, through potentially promising untried candidates, at all stages of the search process. To clarify these points by experimental analysis, some statistical results for fitness-based selection behavior are given in section 2.

This paper is organized as follows. The statistical bases of chromosome reuse idea are illustrated in section 2. Algorithmic description of GAs with chromosome reuse strategy is given in section 3. Section 4 covers the case studies for numerical and combinatorial optimization problems. Finally, conclusions and future research directions are specified in section 5.

2 Statistical Reasoning on the Chromosome Reuse Strategy

The roulette-wheel and the tournament selection methods are the two most commonly used selection mechanisms in genetic algorithms. Both of these selection methods are fitness-based and they aim to produce more offspring from those high-fitness individuals. However, these selection operators leave a significant number of individuals having close to average fitness value unused in the sense that these individuals don't take part in any recombination operation. The idea of chromosome reuse is based on the fact that a significant percentage of these unused individuals have above average fitness values and they should not be just wasted. On the one hand, their reuse will provide additional intensification and diversification capabilities to the evolutionary search process. On the other hand, the use of the individuals in the chromosome library brings no extra computational cost. This is because, the structure and fitness values of these individuals are already known. When these individuals are reused, it is possible to localize an optimal solution over a shorter computational path as exemplified in Section 1 and as demonstrated by experimental evaluations in Section 4.

In order to understand the above reasoning more clearly, let's take the minimization problem for the Ackley's function of 20 variables [20]. A genetic algorithm with 200 individuals, uniform crossover with a crossover rate 0.7 and a mutation rate 0.01 is considered. Since it is more commonly used, the tournament selection operator is selected for illustration. Statistical data are collected over 1000 generations. First, the ratio of the unused individuals to population

size is shown in Figure 1. Obviously, on the average, 74% of the individuals in every generation remain unused, they are simply discarded and replaced by the newly produced offspring individuals. This ratio of unused individuals is independent of the encoding method used. That is, almost the same ratio is obtained with binary-valued and real-valued encodings.

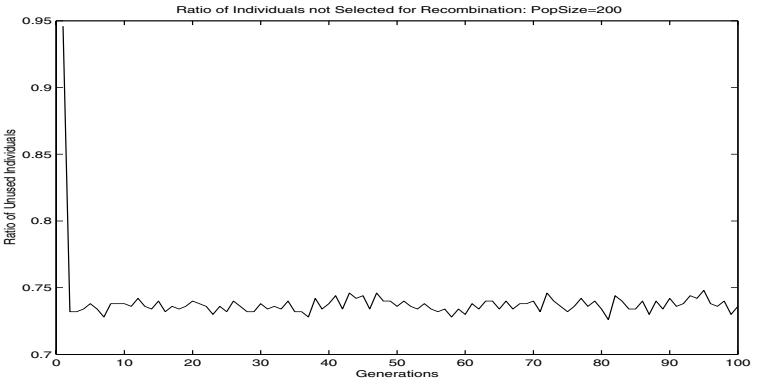

Fig. 1. The ratio of individuals which are not selected in any recombination operation for a population of 200 individuals.

The average ratio of individuals not selected for recombination changes with the population size. For example, this average is 52% for 100 individuals and 85% for 1000 individuals. In addition to this, these average ratios are approximately the same for the roulette-wheel selection method also.

A more clear insight can be obtained from the ratio of unused individuals having a fitness value greater than the population's average fitness. As illustrated in Figure 2, on the average, 32% of the individuals having a fitness value above the population average are not used at all in any recombination operation. The main motivation behind the chromosome reuse strategy is to put these close to average quality individuals into a chromosome library and make use of them for a number of future generations. This way, possible alternative paths to optimal solutions over these potentially promising solutions may be traced. In these experimental evaluations, it is also seen that 24% of the individuals having a fitness value above $0.75 * Average\_Fitness$ are not selected for recombination in all generations. Instead of totally wasting these potentially promising solutions, we can reuse them for a while to speedup the convergence process and to reduce the computational cost of constructing new individuals because chromosomes and fitness values of the individuals in the chromosome library are already determined.

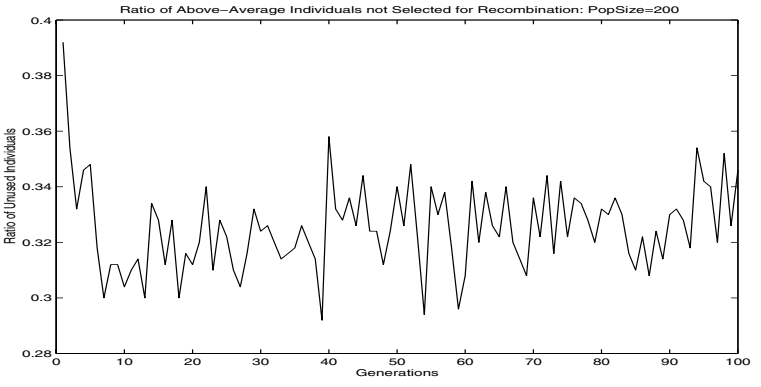

Fig. 2. The ratio of individuals having above average fitness and not selected in any recombination operation for a population of 200 individuals.

3 GAs with Chromosome Reuse Strategy

GAs with chromosome reuse strategy differs from the conventional GAs in the formation and maintenance of a chromosome library and the union of its individuals with the current population during the recombination procedure. The algorithmic description of the proposed approach is given in Figure 3.

In the proposed approach, the total memory space used to store individuals does not increase compared to the memory space needed by conventional GAs, because GAs with chromosome reuse strategy achieves better performance with smaller size populations. In experimental studies, the total number of individuals in the population and in the chromosome library is set equal to the number of individuals in the population of conventional GAs implementation, with which the proposed approach achieved better performance.

4 Two Case Studies

To study the performance of the described chromosome reuse strategy, it is compared with the conventional GAs for the solution of some benchmark problems from numerical and combinatorial optimization fields. Those benchmark numerical optimization problems handled in evaluations are listed in Table 1. They are taken from [20] and [21], which are claimed to provide reasonable test cases for the necessary combination of path-oriented and volume-oriented characteristics of a search strategy. For the combinatorial optimization problems, the 100-city symmetric traveling salesman problem, kroA100, taken from the website http://www.iwr.uni-heidelberg.de/ groups/ comopt/ software/ TSPLIB95/ tsp/ is taken as a representative problem instance.

In all experiments, real-valued chromosomes are used for problem representations. The selection method used is the tournament selection with elitism.

1. $Max\_Library\_Size = \alpha * Population\_Size, 0 < \alpha < 1.0$;
2. $Fitness\_Treshold = \beta, 0 < \beta < 1.0$;
3. $Life\_Time = K$, where K is a predefined integer constant;
4. Generate chromosome library with randomly generated individuals;
5. Set the life time of individuals in the chromosome library to $Life\_Time$;
6. Evaluate chromosome library;
7. Generate initial population;
8. Evaluate initial population;
9. While (NOT DONE)
10. Combine the individuals in the current population and the chromosome library;
11. Reproduction;
12. Crossover;
13. Mutation;
14. Evaluate new population;
15. Decrease the life time of individuals in the chromosome library by 1;
16. Update chromosome library with individuals having $Fitness\_Values > \beta * Average\_Fitness$ and not used in any recombination operation.
17. end

Fig. 3. Genetic algorithms with chromosome reuse strategy.

The elite size is 10% of the population size. The uniform crossover operator is employed with a crossover rate equal to 0.7 and the mutation rate is 0.01. Experiments are carried out using a population size of 200 individuals for conventional GAs, also the total number of individuals in the population and the chromosome library for the proposed approach is 200, i.e. 100 individuals in each. This way, total number of individuals in conventional GAs and GAs with chromosome reuse strategy are kept the same. Individuals in the chromosome library have a predefined life duration, taken as 5 iterations in the experiments, and the removal of an individual from the chromosome library occurs either at the end its life time or the chromosome library is full and an individual with a better fitness replaces it. Each experiment is performed 10 times. All the tests were run over 1000 generations.

In the following worked examples, the results obtained with conventional GA and GAs with the chromosome reuse strategy are compared for relative performance evaluations.

4.1 Performance of Chromosome Reuse Strategy in Numerical Optimization

Conventional GAs are compared with GAs with chromosome reuse strategy for the minimization of functions listed in Table 1. Each function has 20 variables. The best solution found using the conventional GAs and GAS with chromosome reuse strategy are given in Table 2. Chromosome reuse strategy provided very

Table 1. Bechmark functions considered for numerical optimization.

Function Name	Expression
Michalewicz	$f(x) = -\sum_{i=1}^{n-1} sin(x_i) sin(\frac{sin(ix_i^2)}{\pi})^{(2m)}$ $-\sum_{i=1}^{n-1} sin(x_{i+1}) sin(\frac{2x_{i+1}^2}{\pi})^{(2m)}$ $0 \leq x_i \leq \pi$
Griewangk	$f(x) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{x_i}})$ $-100 \leq x_i \leq 100$
Rastrigin	$f(x) = 10n + \sum_{i=1}^{n} (x_i^2 - 10cos(2\pi x_i))$ $-5.12 \leq x_i \leq 5.12$
Schwefel	$f(x) = \sum_{i=1}^{n} (-x_i sin(\sqrt{\lvert x_i \rvert}))$ $-512 \leq x_i \leq 512$
Ackley's	$f(x) = -ae^{-b\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n} cos(cx_i)} + a + e$ $a = 20, b = 0.2, c = 2\pi$ $-32.768 \leq x_i \leq 32.768$
De Jong (Step)	$f(x) = 6n + \sum_{i=1}^{n} \lfloor x_i \rfloor$ $-5.12 \leq x_i \leq 5.12$

close to optimal results in all trials. These results demonstrate the success of the implemented GAs strategy for the numerical optimization problems.

Table 2. Performance evalution of conventional GAs and GAs with chromosome reuse for numerical optimization.

Function	Global Opt., n=Num. Vars.	Best Found: Conv. GA		Best Found: Proposed	
		Global Min.	ITER	Global Min.	ITER
Michalewicz	-9.66 ,$n = m = 10$	-8.55	100	-9.36	100
Griewangk	0, $n = 20$	0.0001	85	$1.0e^{-}8$	35
Rastrigin	0, $n = 20$	0.1	100	0.001	100
Schwefel	$-n * 418.9829, n = 20$	-8159	100	-8374	100
Ackley's	0, $n = 20$	0.03	100	0.001	100
De Jong (Step)	0, n=20	3	100	0	77

4.2 Performance of Chromosome Reuse Strategy in Combinatorial Optimization

To test the performance of the chromosome reuse strategy over a difficult problem of combinatorial type, the 100-city TSP kroA100 is selected. The best found solution for this problem is 21282 obtained using a branch-and-bound algorithm. In the ten experiments performed, the best solution found for this problem using the conventional GAs is 21340 which is obtained in 1000 generations with

population size equal to 200. The best solution obtained with the chromosome reuse strategy is 21282 which is obtained after 620 generations. Figure 4 shows the relative performance of chromosome reuse approach compared to the conventional GAs implementation, the straight line plot shows the results for the chromosome reuse strategy.

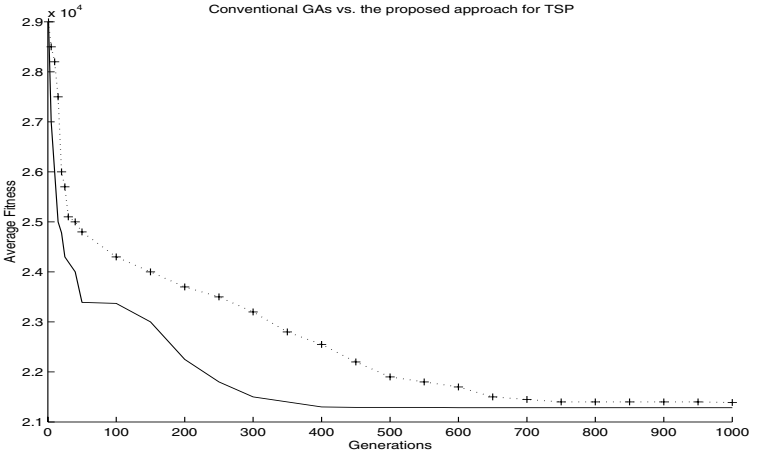

Fig. 4. Performance comparison of conventional genetic algorithms and chromosome reuse strategy in combinatorial

5 Conclusions and Future Work

In this paper a novel external memory-based genetic algorithms strategy based on the reuse of some potentially promising solutions from previous generations for the production of current offspring individuals is introduced as an alternative to the conventional implementation of GAs. The implemented strategy is used to solve difficult problems from numerical and combinatorial optimization areas and its performance is compared with the conventional GAs for representative problem instances. Each problem is solved exactly the same number of times with the employed strategies and the best and the average fitness results are analyzed for performance comparisons. All GA parameters are kept the same in the comparison of the two approaches.

From the results of case studies, for the same population size, it is concluded that the chromosome reuse strategy outperforms the conventional implementation in all trials. The performance of the chromosome reuse approach is the same for both numerical and combinatorial optimization problems. In fact, problems from these classes are purposely chosen to examine this side of the proposed strategy.

This work requires further investigation from following point of views: performance comparisons with other memory-based methods, performance evaluations for other problem classes, such as neural network design, speech processing, and face recognition; problem representations involving variable size chromosomes, particularly genetic programming; and mathematical analysis of chromosome reuse strategy.

References

1. Holland, J.H.: Adaptation in Natural and Artificial Systems: An introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, MIT Press, (1992).
2. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley Publishing Company, (1989).
3. Eshelman, L., Schaffer, J.: Foundations of Genetic Algorithms 2. In: L. Whitley (editor): pp. 187–202, Morgan Kaufmann Publishers, San Mateo, CA, (1993).
4. Back, T.: Evolutionary Algorithms in Theory and Practice, Oxford University Press, (1996).
5. Gen, M., Runwei, C.: Genetic Algorithms in Engineering Design, John Wiley & Sons. Inc., (1997).
6. Miettinen, K., Neitaanmaki, P., Makela, M.M., Periaux, J.: Evolutionary Algorithms in Engineering and Computer Science, John Wiley & Sons Ltd., (1999).
7. Cantu-Paz, E., Mejia-Olvera, M.: Designing efficient master-slave parallel genetic algorithms, IlliGAL Report No. 97004, Illinois Genetic Algorithm Laboratory, Urbana, IL, (1997).
8. Whitley, D., Starkweather, T.: GenitorII: A distributed genetic algorithm, Journal of Experimental and Theoretical Artificial Intelligence, (1990).
9. Eggermont, J., Lenaerts, T.: Non-stationary function optimization using evolutionary algorithms with a case-based memory, url:http://citeseer.nj.nec.com/484021.html.
10. Goldberg, D. E., Smith, R. E.: Non-stationary function optimization using genetic algorithms and with dominance and diploidy, Genetic Algorithms and their Applcations: Proceedings of the Second International Conference on Genetic Algorithms, p. 217–223, (1987).
11. Goldberg, D. E., Deb, K., Korb, B.: Messy Genetic Algorithms: Motivation, analysis, and the first results, Complex Systems, Vol. 3, No. 5, p. 493–530, (1989).
12. Lewis, J., Hart, E., Ritchie, G.: A comparison of dominance mechanisms and simple mutation on non-stationary problems, in Eiben, A. E., Back, T., Schoenauer, M., Schwefel, H. (Editors): Parallel Problem Solving from Nature- PPSN V, p. 139–148, Berlin, (1998).
13. Ryan, C., Collins, J. J.: Polygenic inheritance- a haploid scheme that can outperform diploidy, in Eiben, A. E., Back, T., Schoenauer, M., Schwefel, H. (Editors): Parallel Problem Solving from Nature- PPSN V, p. 178–187, Berlin, (1998)
14. Ryan, C.: The degree of oneness, Firts Online Workshop on Soft Computing, Aug. 19–30, (1996).
15. Ramsey, C.L., Grefenstette, J. J.: Case-based initialization of GAs, in Forest, S., (Editor): Proceedings of the Fifth International Conference on Genetic Algorithms, p. 84–91, San Mateo, CA, (1993).

16. Louis, S., Li, G.: Augmenting genetic algorithms with memory to solve travelling salesman problem, (1997).
17. Louis, S. J., Johnson, J.: Solving similar problems using genetic algorithms and case-based memory, in Back, T., (Editor):Proceedings of the Seventh International Conference on Genetic Algorithms, p. 84–91, San Fransisco, CA, (1997).
18. Luger, G.F.: Artificial Intelligence, 4th edition, Addison-Wesley, (2002).
19. S. Russel and P. Norvig, Artificial Intelligence: A Modern Approach, Prentice-Hall, (1995).
20. http://www.f.utb.cz/people/zelinka/soma/func.html.
21. Kim, H.S., Cho, S.B: An Efficient genetic algorithm with less fitness valuations by clustering, Proc. of the 2001 IEEE Congress on Evolutionary Computation, p.887–894, Seoul, Korea, May 27-30, (2001).