Multi-objective Co-operative Co-evolutionary Genetic Algorithm

Nattavut Keerativuttitumrong[2], Nachol Chaiyaratana[1], and Vara Varavithya[2]

[1] Research and Development Center for Intelligent Systems
[2] Department of Electrical Engineering
King Mongkut's Institute of Technology North Bangkok
1518 Piboolsongkram Rd., Bangkok 10800, Thailand
E-mail: {nchl,vara}@kmitnb.ac.th

Abstract. This paper presents the integration between two types of genetic algorithm: a multi-objective genetic algorithm (MOGA) and a co-operative co-evolutionary genetic algorithm (CCGA). The resulting algorithm is referred to as a multi-objective co-operative co-evolutionary genetic algorithm or MOCCGA. The integration between the two algorithms is carried out in order to improve the performance of the MOGA by adding the co-operative co-evolutionary effect to the search mechanisms employed by the MOGA. The MOCCGA is benchmarked against the MOGA in six different test cases. The test problems cover six different characteristics that can be found within multi-objective optimisation problems: convex Pareto front, non-convex Pareto front, discrete Pareto front, multi-modality, deceptive Pareto front and non-uniformity in the solution distribution. The simulation results indicate that overall the MOCCGA is superior to the MOGA in terms of the variety in solutions generated and the closeness of solutions to the true Pareto-optimal solutions. A simple parallel implementation of MOCCGA is described. With an 8-node cluster, the speed up of 2.69 to 4.8 can be achieved for the test problems.

1 Introduction

A *genetic algorithm* has been established as one of the most widely used technique for multi-objective optimisation. This is because the parallel search nature of genetic algorithm makes the task of approximating Pareto front of optimal solutions in one optimisation run becomes possible. From early developments in the eighties (Schaffer, 1984; Fourman, 1985) to the introduction of a direct relationship between Pareto-optimality and a fitness function (Horn and Nafpliotis, 1993; Fonseca and Fleming, 1993), research interests in this branch of evolutionary computation remain strong. Regardless of the methodology employed, the ultimate aim of multi-objective optimisation using a genetic algorithm remains the same: to identify the solutions that approximate the true Pareto-optimal solutions.

Various types of genetic algorithm are currently available for use in multi-objective optimisation (Hajela and Lin, 1992; Horn and Nafpliotis, 1993; Fonseca and Fleming, 1993; Zitzler and Thiele, 1999). Although a number of applications have been benefited from the use of such algorithms, as the search space or problem size increases, the performance of the algorithm always degrades. As a result, the non-dominated solution set identified by the algorithm may highly deviate from the true Pareto front. In addition, the coverage of the Pareto front by the solutions generated may also be affected. This is because the issue regarding the relationship between the problem representation and the size of search space has rarely been discussed in the context of algorithm development for use in *multi-objective optimisation.* Nonetheless, various techniques for single-objective optimisation using a genetic algorithm are available for solving the problem induced by an increase in problem size. One possible technique involves an insertion of a *co-operative co-evolutionary* effect into the search algorithm. A genetic algorithm that exploited such strategy is known as a co-operative co-evolutionary genetic algorithm or CCGA (Potter and De Jong, 1994). In contrast to other co-evolutionary genetic algorithms where the co-evolutionary effect found among sub-populations is the result of a competition for survival by the individuals, the co-evolutionary effect in the CCGA is produced by a co-operation among all species. In brief, a species member in the CCGA represents a part of the decision variable set where all species will co-operatively produce complete solutions to the problem. Each species member will then independently evolve using a standard genetic algorithm mechanism. By partitioning the problem in this manner, the search space that each sub-population has to cover would significantly reduce. Although the CCGA is originally developed for use in single-objective optimisation, the co-operative co-evolutionary effect can also be embedded into a genetic algorithm which is designed for multi-objective optimisation.

In this paper, the co-operative co-evolutionary effect as described by Potter and De Jong (1994) will be integrated into a genetic algorithm called a multi-objective genetic algorithm or MOGA (Fonseca and Fleming, 1993). The MOGA is chosen for this case because of its simplicity and the clear definition regarding the relationship between the Pareto-optimality level of a solution and its corresponding fitness value. The modified MOGA will be referred to as a multi-objective co-operative co-evolutionary genetic algorithm or MOCCGA throughout the paper. Note that the co-operative co-evolutionary effect can also be integrated into other types of genetic algorithm that are designed for use in multi-objective optimisation. In addition to the proposed integration between two genetic algorithms, possible approach for parallel implementation of the developed algorithm will also be discussed in this paper.

The organisation of this paper is as follows. In section 2, the background on the multi-objective genetic algorithm (MOGA) and the co-operative co-evolutionary genetic algorithm (CCGA) will be discussed. The integration between the MOGA and the CCGA will be explained in section 3. In addition, the test problems that will be used to assess the performance of the MOGA will also be given in this section. In section 4, the benchmarking results and discussions are given. Section 5 describes the parallel implementation of MOCCGA and its test results. Finally, conclusions are drawn in section 6.

2 Multi-objective Genetic Algorithm and Co-operative Co-evolutionary Genetic Algorithm

Two types of genetic algorithm that will be integrated together are the multi-objective genetic algorithm (MOGA) and the co-operative co-evolutionary genetic algorithm (CCGA). A brief description of the algorithms follows.

2.1 Multi-objective Genetic Algorithm

The multi-objective genetic algorithm (MOGA) was first introduced by Fonseca and Fleming (1993). The MOGA functions by seeking to optimise the components of a vector-valued objective function. Unlike single-objective optimisation, the solution to a multi-objective optimisation problem is a family of points known as the Pareto-optimal set. Each point in the set is optimal in the sense that no improvement can be achieved in one component of the objective vector that does not lead to degradation in at least one of the remaining components. Given a set of possible solutions, a candidate solution is said to be Pareto-optimal if there are no other solutions in the solution set that can dominate the candidate solution. In other words, the candidate solution would be a non-dominated solution. Assuming, without loss of generality, a minimisation problem, an n-dimensional cost vector $\mathbf{a}$ is said to be dominating another n-dimensional cost vector $\mathbf{b}$ if, and only if, $\mathbf{a}$ is partially less than $\mathbf{b}$ ($\mathbf{a}\ \ p < \mathbf{b}$), i.e.

$$\mathbf{a}\ \ p < \mathbf{b} \leftrightarrow \forall i = 1, \ldots, n : a_i \leq b_i \wedge \exists i = 1, \ldots, n : a_i < b_i \tag{1}$$

By identifying the number of solutions in the solution set that dominate the solution of interest, a rank value can be assigned to the solution. In other words, the rank of a candidate solution is given by the number of solutions in the solution set that dominate the candidate solution. After a rank has been assigned to each solution, a fitness value can then be interpolated onto the solution where a genetic algorithm can subsequently be applied in the optimisation procedure. Note that since the aim of a search by the MOGA is to locate Pareto-optimal solutions, in essence the multi-objective optimisation problem has also been treated as a multi-modal problem. Hence, the use of additional genetic operators including the fitness sharing and mating restriction procedures is also required. However, in addition to the usual application of the fitness sharing and mating restriction procedures in the decision variable space (Fonseca and Fleming, 1995), they can also be carried out in the objective value space (Fonseca and Fleming, 1993). A comprehensive description of the MOGA which covers other advanced topics including goal attainment and priority assignment strategies can be found in Fonseca and Fleming (1998).

2.2 Co-operative Co-evolutionary Genetic Algorithm

The co-operative co-evolutionary genetic algorithm (CCGA) was first introduced by Potter and De Jong (1994). The CCGA functions by introducing an explicit notion of modularity to the optimisation process. This is done in order to provide reasonable opportunity for complex solutions to evolve in the form of interacting co-adapted sub-components. In brief, the CCGA explores the search space by utilising a population which contains a number of species or sub-populations. In contrast to other types of genetic algorithm, each species in the CCGA represents a variable or a part of the problem which is needed to be optimised. A combination of an individual from each species will lead to a complete solution to the problem where the fitness value of the complete solution can be found in the usual way. This value of fitness will be assigned to the individual of interest that participates in the formation of the solution. After the fitness values have been assigned to all individuals, the evolution of each species is then commenced using a standard genetic algorithm. Although the CCGA has been successfully used in various applications, its performance can reduce in the circumstance where there are high interdependencies between the optimisation function variables. In order to solve this problem, Potter and De Jong (1994) have suggested that the fitness of a species member should be obtained after combining it with the current best individuals or the randomly selected individuals from the remaining species depending upon whether which combination yields a higher fitness value. This helps to increase a chance for each individual to achieve a fitness value which is appropriate to its contribution to the solution produced. A comprehensive description of the CCGA and the summary of its applications can be found in Potter and De Jong (2000).

3 MOCCGA and Test Problems

By combining the MOGA and the CCGA together, the resulting algorithm can be referred to as a multi-objective co-operative co-evolutionary genetic algorithm or MOCCGA. Similar to the CCGA, each species in the MOCCGA represents a decision variable or a part of the problem which is needed to be optimised. However, instead of directly assigning a fitness value to the individual of interest which participates in the construction of the complete solution, a rank value will be determined first. Similar to the MOGA, the rank of each individual will be obtained after comparing it with the remaining individuals from the same species. Then a fitness value can be interpolated onto the individual where a standard genetic algorithm can be applied within each sub-population. Note that in this investigation, the fitness sharing strategy utilised in the MOCCGA is similar to the one described in Fonseca and Fleming (1993) where the fitness sharing is carried out in the objective space.

In order to assess the performance of the MOCCGA, the MOCCGA will be benchmarked against the MOGA in six optimisation test cases. These six test problems are developed by Zitzler et al. (2000) for use in multi-objective optimisation benchmarking. The problems are minimisation problems with m decision variables and two objectives. Brief descriptions of the test problems are summarised in Table 1. Each test problem represents different aspects of multi-objective optimisation problems. The benchmarking results will be displayed and discussed in the following section.

Table 1. Descriptions of the test problems.

Test Problem	Characteristic
T_1	Convex Pareto front
T_2	Non-convex Pareto front
T_3	Discrete Pareto front containing several non-contiguout convex parts
T_4	Multi-modality
T_5	Deceptive Pareto front
T_6	Non-uniformity in the solution distribution in the search space

4 Benchmarking Results and Discussions

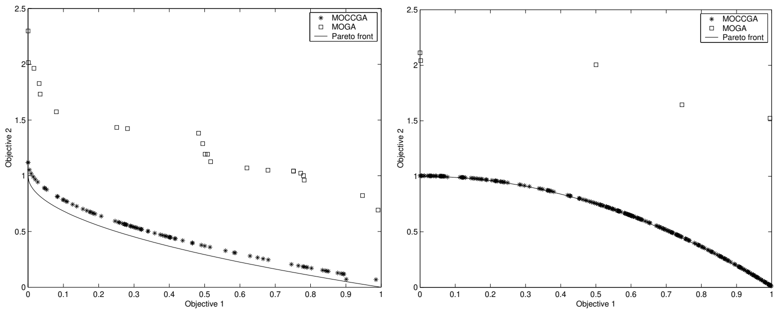
The MOCCGA will be benchmarked against the MOGA in six test cases described in section 3. The common parameter settings for both the MOGA and the MOCCGA are displayed in Table 2.

Table 2. Common parameter settings for both the MOGA and the MOCCGA.

Parameter	Value
Selection method	Stochastic universal sampling (Baker, 1989)
Crossover probability	0.7
Mutation probability	0.01

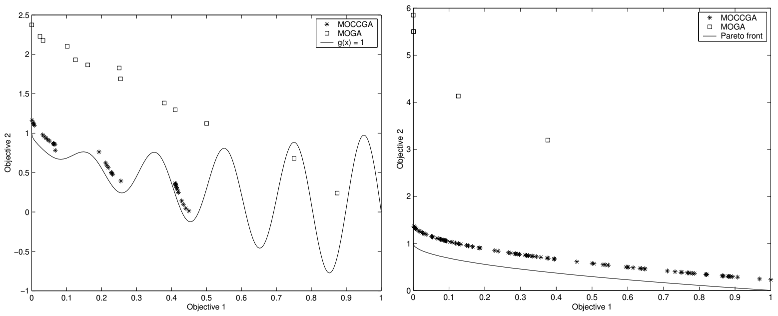
The parameter settings which vary from one test problem to the others are the number of generations, the number of species (for the case of the MOCCGA), the number of individuals and the length of binary chromosome. These parameters are set to accommodate the size of search space in each problem. Although the settings are different for each problem, the parameter values are chosen such that the total numbers of objective evaluations are the same for both the MOGA and the MOCCGA. Also recall that the objective values are evaluated twice for each individual in the MOCCGA using the strategy based on the one described in Potter and De Jong (1994).

In this investigation, each algorithm is run five times using different initial populations. Then the search results from all runs are combined where the non-dominated solutions are subsequently extracted from the overall results. The non-dominated solutions located by the MOGA and the MOCCGA from all test cases are displayed in Figures 1-3.



(a) (b)

Fig. 1. (a) Non-dominated solutions and the true Pareto front of the test problem T_1, (b) Non-dominated solutions and the true Pareto front of the test problem T_2.

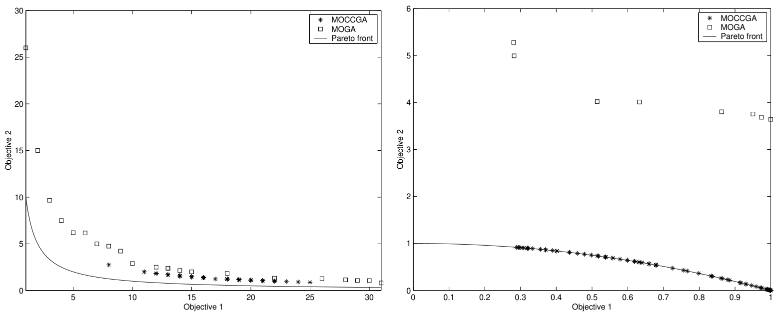


(a) (b)

Fig. 2. (a) Non-dominated solutions and the boundary for $g(\mathbf{x}) = 1$ in the test problem T_3, (b) Non-dominated solutions and the true Pareto front of the test problem T_4.

Firstly, the range of variety in solutions found is considered. The MOCCGA can identify the solutions which cover the whole Pareto front in the cases of T_1, T_2 and T_4. In contrast, the MOGA can locate the solutions which indicate the boundary of Pareto front only in the case of T_5. From these observations, the performances of the MOGA and MOCCGA are lowest with the problem that has a discrete Pareto front (T_3) or a non-uniform distribution of solutions in the search space and along the Pareto front (T_6). This means that although the co-operative co-evolutionary effect can help improving the performance of the search algorithm in the context of the Pareto front coverage, this additional effect is insufficient for the algorithm to cope with the discrete and non-uniformity features of the optimisation problems.

Moving onto the consideration on the closeness of non-dominated solutions to the true Pareto-optimal solutions. The MOCCGA has proven to be highly efficient in all test problems. In particular, most of solutions identified by the MOCCGA in the test problems T_1, T_2, T_3, T_4 and T_6 dominate the corresponding solutions identified by the MOGA. The only test problem at which the



(a) (b)

Fig. 3. (a) Non-dominated solutions and the true Pareto front of the test problem T_5, (b) Non-dominated solutions and the true Pareto front of the test problem T_6.

MOGA can identify non-dominated solutions that are reasonably close to the true Pareto-optimal solutions is the problem T_5. Moreover, the non-dominated solutions identified by the MOCCGA in the cases of T_2 and T_6 are also the true Pareto-optimal solutions. In overall, the introduction of co-operative co-evolutionary effect can improve the genetic algorithm performance in terms of the Pareto front coverage and the number of solutions found which are close to the true Pareto-optimal solutions.

5 Parallel Implementation of MOCCGA

Genetic algorithms inherit a high degree of parallelism. A large portion of tasks that operated on a population of individuals can be performed without dependency. Examples of parallel tasks in GAs include objective function evaluations and genetic operators. Parallel computing can improve the performance of GAs in two ways, increasing population size to solve complex problem or reducing computation time. Many parallel implementations of GAs have been purposed. Some of the parallel GAs can be found in Spiessens and Manderick (1991) and Gorges-Scheleuter (1990).

In parallel GAs, the entire population is partitioned into sub-populations. The sub-population can consist of complete strings representing the whole decision variable sets or portion of strings. Each sub-population is processed by an assigned processor. At each iteration, sub-populations interact with each other, according to GAs, by exchanging information between processors. In multiprocessor systems, the cost of exchanging information is considered as an overhead. The performance of parallel GAs is therefore limited by the amounts of information that need to be exchanged between sub-populations. For this reason, reducing information exchanges between processors is one approach of improving the performance of parallel GAs. In simple GAs, the algorithm has access to all individuals of the population. However, the information of all individuals is not required to maintain the performance of GAs. The parallel GAs can take advantage of this fact and exchange only a subset of available information. For example, only the best individuals are exchanged between sub-populations. The performance improvement is obtained not only by less communication overhead but also high diversity of chromosome to avoid premature convergent. Hart, et al. (1996) studied the effects of relaxed synchronization on parallel GAs in which an improvement on execution time was shown. The effects of chromosome migration was investigated by Matsumura, et. al. (1997). Each processor executes the genetic operations on a set of chromosome and exchange information to only its neighbours based on different network topologies.

As the price to performance ratio of computers has dropped in a rapid rate, high performance computing platforms can be built by interconnecting a group of PCs, so called a cluster. The clusters have been implemented widely and their application domains have been extended into many new areas. Patrick, et. al. (1997) has proposed a distributed genetic algorithm for cluster environment in which a set of library functions for performing parallel genetic operations is provided. However, the parallel implementation of MOCCGA requires different details. In MOCCGA, operations in ranking calculation, selection, and crossover require information accesses within the same species. Information from different species is required only in the objective value calculation. Therefore partitioning the MOCCGA based on species can reduce amounts of information exchange and simplify the program coding. Figure 4 shows the problem partitioning based on Single Program Multiple Processors paradigm. The parallel process starts from a single node and creates parallel process on a set of computers. Each process responsible for one species or a group of species. The information exchange as well as synchronisation is carried out at the beginning of the objective calculation stage.

In original MOCCGA, the fitness calculation strategy prevents communication reduction since fitness calculation for each species require the whole species information in order to have a complete gene pool. The objective values are calculated from two combinations; firstly,the decision variable (after decode the string) of individual in the current species is combined with the best decision variable from other species, and secondly, the decision variable of the current species is combined with the decision variable of individual from other species in random. The objective values of both cases are then compared and the better one is selected. The communication at the beginning of objective calculation involves the broadcast of the best decision variable of individuals and the whole decision variable of the species to all other processes. The communication performance can be improved using divide and conquer approach in which $\log_2 P$ communication steps are adequate for P processes. The parallel version of MOCCGA was developed using C language and MPI library. The performance results were measured from a 8-node cluster. The number of test species is fixed to 32. An individual is encoded as a binary of 10 bits. The stochastic universal sampling was used in the selection process. The test problem T_1 is selected in the test. We have considered three problem sizes, small, medium, and large problem. The test results are shown in Table 3

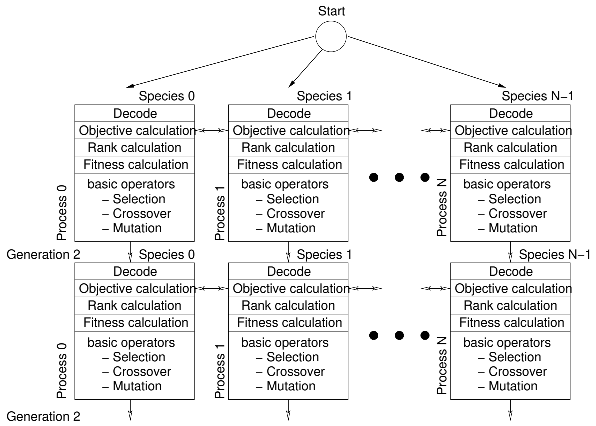


Fig. 4. Implementation of parallel MOCCGA. The communication occurs before the objective function calculation.

Table 3. Performance results of the parallel MOCCGA: The execution time is shown in second. The number of individuals in small, medium, and large problems is equal to 100, 500, and 1000 individuals, respectively. The small, medium, and large problems are processed, respectively, for 100, 20, and 10 generations. In Bcast, the standard MPI_bcast() library is used in the program. The divide and conquer broadcast using send() and receive() is used in Log_P results. The speedup results are calculated using Bcast results.

	Small			**Medium**			**Large**		
	Bcast	Log_P	SpeedUp	Bcast	Log_P	SpeedUp	Bcast	Log_P	SpeedUp
1	53.83	59.05	1	299.55	331.95	1	721.48	754.92	1
2	32.16	34.14	1.6	194.53	205.49	1.5	490.72	529.50	1.4
4	19.17	19.15	2.7	129.12	128.31	2.3	355.32	352.28	2.0
8	11.90	10.41	4.8	89.99	86.55	3.3	268.61	260.54	2.69

The speed up of parallel MOCCGA is quite satisfactory. We considered two approaches of broadcasting objectives value, using MPI_Bcast function library and customize decision value broadcast based on the divide and conquer technique. The results show slightly different in performance. The custom broadcast has better performance results for the 4 and 8 processors. The performance improvement results from the parallelism in communication.

6 Conclusions

In this paper, the integration between two types of genetic algorithms are presented: a MOGA and a CCGA algorithm. The MOCCGA has been benchmarked against the MOGA in six test cases. The search performance of the MOGA can be improved by adding the co-operative co-evolutionary effect to the algorithm. A parallel implementation of MOCCGA was described.

References

Baker, J. E. (1989). An analysis of the effects of selection in genetic algorithms, Ph.D. Thesis, Computer Science Department, Vanderbilt University, Nashville, TN.

Fonseca, C. M. and Fleming, P. J. (1993). Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. Genetic Algorithms: Proceedings of the Fifth International Conference, Urbana-Champaign, IL, 416-423.

Fonseca, C. M. and Fleming, P. J. (1995). Multiobjective genetic algorithms made easy: Selection, sharing and mating restriction. The Second International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA'95), Sheffield, UK, 45-52.

Fonseca, C. M. and Fleming, P. J. (1998). Multiobjective optimization and multiple constraint handling with evolutionary algorithms - Part 1: A unified formulation. IEEE Transactions on Systems, Man, and Cybernetics, 28(1), 26-37.

Fourman, M. P. (1985). Compaction of symbolic layout using genetic algorithms. Proceedings of the First International Conference on Genetic Algorithms and Their Applications, 141-153.

Gorges-Scheleuter, M. (1990). Explicit parallelism of genetic algorithms through population structures. Parallel Problem Solving from Nature, 150-159.

Hajela, P. and Lin, C. -Y. (1992). Genetic search strategies in multicriterion optimal design. Structural Optimization, 4, 99-107.

Hart, W. E.,Baden, S., Bekew, R. K. and Kohn, S. (1996). Analysis of the numerical effects of parallelism on a parallel genetic algorithm, Proceeding of the 10th international parallel processing symposium, 606–612.

Horn, J. and Nafpliotis, N. (1993). Multiobjective optimization using the niched pareto genetic algorithm, IlliGAL Report No. 93005, Department of Computer Science, Department of General Engineering, University of Illinois at Urbana-Champaign, Urbana-Champaign, IL.

Matsumura, T., Nakamura, M. and Okech, J. (1997). Effect of chromosome migration on a parallel and distributed genetic algorithm, International Symposium on Parallel Architecture, Algorithm and Networks (ISPAN'97), 357–612.

Patrick D., Green, P. and York, T. (1997). A distributed genetic algorithm environment for unix workstation clusters. Genetic Algorithms in Engineering Systems: Innovations and Applications, 69–74.

Potter, M. A. and De Jong, K. A. (1994). A cooperative coevolutionary approach to function optimization. Proceedings of the Third International Conference on Parallel Problem Solving from Nature (PPSNIII), Jerusalem, Israel, 249-257.

Potter, M. A. and De Jong, K. A. (2000). Cooperative coevolution: An architecture for evolving coadapted subcomponents. Evolutionary Computation, 8(1), 1-29.

Schaffer, J. D. (1984). Some experiments in machine learning using vector evaluated genetic algorithms, Ph.D. Thesis, Vanderbilt University, Nashville, TN.

Spiessens, P., and Manderick, B. (1991). A massively parallel genetic algorithm: Implementation and first analysis, the Fourth International Conference on Genetic Algorithms, 279-285.

Zitzler, E. and Thiele, L. (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. IEEE Transactions on Evolutionary Computation, 3(4), 257-271.

Zitzler, E., Deb, K. and Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computation, 8(2), 173-195.