Publication P4

J. Martikainen and S. J. Ovaska
"Hierarchical two-population genetic algorithm"
in
International Journal of Computational Intelligence Research
vol. 2, no. 4, 2006, in press.

Hierarchical Two-Population Genetic Algorithm

Jarno Martikainen[1] and Seppo J. Ovaska[2]

Helsinki University of Technology, Institute of Intelligent Power Electronics
P. O. Box 3000, 02015 HUT, FINLAND
[1] E-mail: jkmartik@cc.hut.fi
[2] E-mail: ovaska@ieee.org
URL: http://powerelectronics.hut.fi

Abstract*: This paper proposes a new hierarchical two-population genetic algorithm (2PGA). The 2PGA scheme constitutes of two differently sized populations containing individuals of similar fitness or cost function values. The smaller population, the elite population, consists of the best individuals, whereas the larger population contains less fit individuals. These populations have different characteristics, such as size and mutation probability, based on the fitness of the candidate solutions in these populations. The performance of our 2PGA is compared to that of a single population genetic algorithm (SPGA). Because the 2PGA has multiple parameters, the significance and the effect of the parameters is also studied. Experimental results show that the 2PGA outperforms the SPGA reliably without increasing the amount of fitness function evaluations. Although genetic algorithms are used as a platform for the 2PGA scheme, the principles presented here are applicable also to other population based evolutionary optimization methods.

Keywords: Genetic algorithms, multipopulation genetic algorithm, hierarchical populations, evolutionary algorithms, coevolution.

I. Introduction

Genetic algorithms (GA) [1]-[3] is a branch of evolutionary computation modeling the nature's way of finding competitive solutions based on the demands of the environment. GAs, like other evolutionary computation methods, evolve solutions in an iterative manner by applying variation and selection operators to a pool of candidate solutions. To enhance GAs performance, genetic algorithms can take advantage of parallel computing environments due to their well parallelizable nature. Parallel implementations usually constitute of multiple populations running on several separate computing units. Such parallel GAs have been studied before, and [4] offers a good introduction to the basics of the field. Parallelism in evolutionary algorithms in general is discussed in [5]. Parallel multipopulation GAs and their parameters are discussed in [6] and [7]. These parallel approaches mainly rely on implementing multipopulation GAs so that individual populations have their own computing units and, thus, these schemes can usually be considered as parallel single population genetic algorithms.

However, when considering our surrounding environment, it is remarkably distinct that these different populations are internally divided into smaller groups. These groups, whether animal or human, share the property of being roughly homogeneous when considering their fitness value based on some appropriate fitness function. Genetic algorithms using subpopulations have been studied up to some extent before. Parallel genetic algorithm scheme is studied in [8] in which different crossover operators are used in different subpopulations for exploration and exploitation. Fourier expansion based approach of dividing the fitness landscapes to sub-landscapes and searching them using subpopulations is presented in [9].

A notion that the mere implementation of parallel GA without actual parallel hardware adds to the performance of a serial GA is suggested in [10]. Niching [11], [12] also known as speciation, is a popular multipopulation scheme in evolutionary computation. In niching, subpopulations are used to search a sub-region of the fitness landscape.

Coevolution, the use of multiple evolutionary algorithms or populations in parallel to achieve a common goal can also be seen as a multipopulation approach to evolutionary computation. In [13] two separate populations are used: a solution population and a population of test cases. The fitness function can be a combination of different test cases. In that scheme, the desired property of the test case population is to gradually evolve into more challenging but appropriate test cases thus enabling the solution population to be able to solve even harder problems as the algorithm proceeds. An optimization scheme in which two genetic algorithms are working with the same population is discussed in [14]. The coevolution scheme can be either co-operative, in which the multiple components work on the same side, or a *predator-prey* scheme, in which the components fight each other as in [15].

In this paper, we present a hierarchical two-population genetic algorithm (2PGA) that implements two populations on a single processing unit. In our approach, the population is divided into a small elite population and large, hierarchically lower population based on the fitness values of the chromosomes. The method can be embedded to parallel implementations of GAs running single population GAs, or any other population based evolutionary computation scheme, side by side. Nowadays practicing engineers do not have the time to repeat calculations a number of times, and, thus, our goal is to create an algorithm that would reliably produce competitive quality solutions using only a few runs.

Many methods exist that outperform the standard GA. For example, in [16] a powerful branch and bound method with local sampling is introduced, [17] introduces an advanced scatter search mechanism and [18] discusses a hybrid algorithm fusing genetic algorithms with a local search method. However, our proposed method is intended only to be an improvement to the basic algorithm that can be later enhanced using different methodologies, e.g., local search mechanisms.

The performance of the proposed 2PGA is compared to that of a single population GA (SPGA) and it is evaluated using the well-known minimization problems of Ackley and Rastrigin, as well as a demanding filter design maximization problem and a traveling salesman problem (TSP).

This paper is organized as follows. Section II discusses the proposed 2PGA scheme and Section III describes the reference SPGA scheme. Section IV explains the problems used in testing the performance of the algorithms. Section V summarizes the experimental results. Section VI discusses a three-population hierarchical GA, and Section VII concludes the article.

II. Hierarchical 2PGA

The idea of our multipopulation genetic algorithm originates from the notion of nature dividing various populations into subpopulations, e.g., a small elite and a large plain, based on their fitness similarities. In our 2PGA scheme the two populations evolve separately in parallel, but they are exchanging chromosomes under certain conditions, i.e., the best chromosome from the plain population is allowed to enter the elite population if its fitness value is high enough. Then again, the worst chromosome from the elite population is transferred to the plain population to keep the population sizes constant.

The evolution in both of the subpopulations is as in the SPGA. The principal difference is that in the plain population the mutation probability is higher. The analogy supporting this assumption can be derived from nature, where weaker individuals, in terms of fitness, have to change their behavior more in order to succeed in competition.

GAs typically have a tendency of finding a good neighborhood fast, but it may take a long time to reach the optimum in that area [11]. This is the reason for using specialized hybrid methods, in which a GA is used for global search, and local search is carried out by some more traditional technique, such as the hill-climbing method. Our 2PGA is an effort to improve the basic GA with little additional computation and no separate algorithms.

The operation of our 2PGA can be divided into seven stages as follows:

1. Generate an initial random population of solutions.

2. Evaluate the fitness (or cost) of the chromosomes in the initial population and divide the population into a small elite population and large plain population.

3. Evaluate the fitnesses of plain and elite populations.

4. Implement reproduction separately in both of the populations.

5. Compose populations for the next generation combining parents, offspring, and possibly migrated chromosomes. If the fitness value of the best chromosome in the plain population supersedes a certain limit value, exchange this chromosome with the worst chromosome in the elite population. The parents not chosen to reproduce in the previous round in the elite population migrate to the plain population.

6. Mutate chromosomes using different mutation probabilities for both of the populations. Elitist mutation that keeps the best solutions in both the populations intact is used.

7. Go to 3 or exit if convergence or run time constraints have been met.

The basic idea of 2PGA algorithm is to conduct global and local search in parallel using two populations. The elite population, having a small mutation probability, searches among the best solutions to find even better solutions, whereas the large plain population, with large mutation probability, searches the whole search space in hope of finding new promising areas of high fitness.

The proposed method is illustrated in Figs 1-3. Figure 1 describes the division of the initial population into two separate subpopulations in the 2PGA scheme. In this illustrative example, the size of the initial population is only 14 and the elite and the plain populations to be formed contain 4 and 10 chromosomes, respectively. So, the elite population size, *es*, is 4. This division into subpopulations can be carried out directly after initialization, or alternatively, the initial population can be allowed to converge as a single population GA for some time. We call the point of division as the *population division point*, n_d.

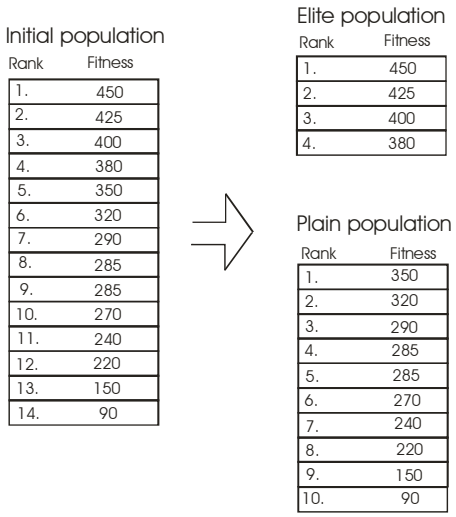

Figure 1. Example of how the initial population is divided into two subpopulations.

In addition to the difference in subpopulation sizes, the characteristics of the populations also differ in terms of mutation probabilities; m_p, the mutation probability of the plain population is higher than the mutation probability of the elite population, m_e. If the initial population is not divided directly after initialization, mutation probability for the initial population is described by m. In both plain and elite populations an elitist mutation scheme is applied so that the best chromosome is never mutated. Thus, two solutions per generation are kept intact in terms of mutation, one for each population.

Figure 2 describes how the populations for the next generation are composed in 2PGA. Within both the elite and plain populations the best half of the chromosomes are selected as parents for the next generation. In Fig. 2 this means that the two best chromosomes from the elite population produce two offspring. The worst two chromosomes in the elite population are transferred to the plain population. Accordingly, in the plain population the four best chromosomes act as parents for four offspring.

Therefore, the elite population in the next generation is composed of two new offspring and two old parents from the previous generation. The plain population, then again, is composed of the two chromosomes left out from the elite population in the previous generation. In addition, there is also the new offspring of the previous plain population parent chromosomes. To fill the plain population up to the fixed number of chromosomes we add as many chromosomes from the previous plain population as there is space for. In numbers, two chromosomes from the previous elite population are accompanied by four new plain population offspring. Therefore, we can accommodate four chromosomes from the previous plain population. The rest of the chromosomes are discarded.

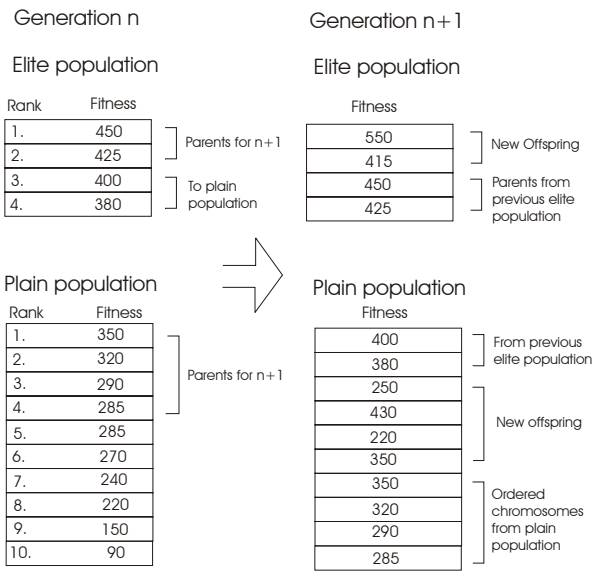

Figure 2. Example of how the subpopulations of generation n+1 are formed from the subpopulations of generation n.

Figure 3 describes the process of how plain chromosomes can enter the elite population. Plain chromosomes need not have access to the elite population during every generation. i_m, the migration interval describes how often the chromosomes are allowed to enter from plain population to the elite population. The *migration condition*, c_m, describes the condition based on which the migration either does or does not take place. In Fig. 3 c_m equals one, meaning that the best chromosome in the plain population has to be better in terms of fitness value than the best chromosome in the elite population in order migration to take place. Indeed, 1150 excels 1000 and thus migration takes place. To keep the population sizes constant, the worst chromosome from the elite population, valued 850, is transferred to the plain population.

The proposed 2PGA scheme can be embedded on every population based evolutionary optimization scheme. It is not designed to compete with highly sophisticated and application specific optimization schemes, such as hybrid evolutionary algorithm methods equipped with gradient-based local search methods. Instead, the proposed scheme is intended to improve the performance of the basic evolutionary algorithm with minimal computational overhead, and thus any modification benefiting a standard evolutionary algorithm will also benefit the proposed scheme.

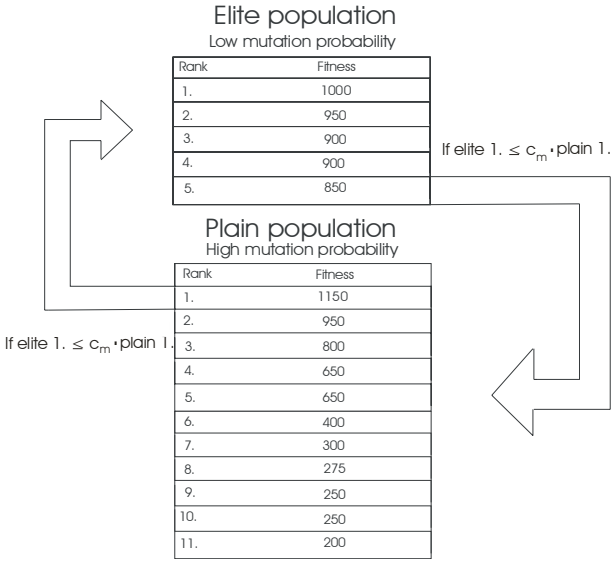

Figure 3. Example of exchanging chromosomes between the two populations in the case the migration condition is fulfilled.

The implementation of the 2PGA scheme does not require parallel hardware, rather it is suitable for all platforms, both single processor as well as parallel environments. In the 2PGA scheme the two populations are different in size as well as in the mutation probability. The proposed scheme is similar to coevolution schemes, in which separate populations evolve towards a common goal. Also, the 2PGA scheme resembles niching since the individuals can only reproduce with individuals within the same population. However, these individuals can migrate between the populations and this does not require complex computations, as could be the case in niching methods implemented using crowding or fitness sharing.

III. The Reference Genetic Algorithm

To evaluate the performance of the proposed 2PGA method, a reference SPGA was constructed and applied to same problem as our 2PGA. This reference SPGA operated in six stages [1]:

1. Create an initial population randomly.

2. Evaluate the fitness of the chromosomes.

3. Mate the chromosomes to produce offspring using crossover. Select the best half of the population to become parents for the next generation.

4. Select parents and offspring to survive to the next generation.

5. Apply elitist mutation to chromosomes so that the best chromosome is not mutated.

6. Go to 2 or exit if convergence or run time constraints have been met

The initial populations in the reference SPGAs were created randomly. Reproduction was implemented so that parents were selected based on their rank and they mated with the probability of 1. The best parent mated with the second best, third best with fourth best, and so on. Reproduction was implemented using blending crossover [3] in optimizing Ackley's and Rastrigin's functions and a single point crossover in the filter design problem. The traveling salesman problem was implemented using a modification of Grefenstette's greedy crossover [19]. In all the problems, the population for the next generation was built up using the best half of the previous generation and the offspring of these parents, thus keeping the population size constant.

Mutation was implemented using an elitist scheme, in which the best chromosome was never mutated. The mutation probabilities, *m*, and initial population sizes for the reference SPGAs are shown in Table 1. Mutation probability here means the probability of a single solution being selected for mutation.

Problem	*m*	Population size
Ackley	10%	100
MGP	40%	80
Rastrigin	20%	100
TSP	10%	40

Table 1. Reference SPGA parameters.

Values presented in Table 1 were found suitable based on several test runs. To ensure statistical reliability, each algorithm was run 50 times before averages of the results were taken. The number of generations run in the Ackley's and Rastrigin's functions case was 100 000. The filter design problem was run for 300 generations and the traveling salesman problem was run for 2000 generations. The number of generations in each problem was determined by the computational requirements of the problem.

Naturally, there are algorithms outperforming the standard GA, but in this paper comparison is made between the standard GA and the proposed method, because this paper studies only the effect of the two-population scheme.

IV. Performance Test Description

A collection of four different kinds of test cases is used in this paper and these problems test the capability of the proposed scheme in different types of problems. The functions of Ackley and Rastrigin represent continuous multimodal problems with single optimum. The dimension of 30 and 50 is used for these functions, respectively. The optimization of the multiplicative general parameter finite impulse response filter is a demanding discrete optimization problem, with no known global optimum. Finally, a demanding 100-city traveling salesman optimization problem was used to study combinatorial optimization problems.

A. Ackley's Function

The well-known Ackley's function [1] is a continuous minimization problem presenting exhaustive search space, in which random walk or other brute force methods hardly give satisfactory results in a reasonable time. Ackley's function is defined as

$$f(x) = -c_1 \cdot \exp\left(-c_2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\cdot\sum_{i=1}^{n}\cos\left(c_3\cdot x_1\right)\right) + c_1 + \exp(1) \quad (1)$$

In our tests, we used the parameter values suggested in [1]. The values are

$$c_1 = 20,\ c_2 = 0.2, c_3 = 2\pi, n = 30, -20 \le x_i \le 30.$$

The global minimum of Ackley's function is zero, and this is achieved with parameter vector $x = [0, \ldots, 0]^T$.

B. Multiplicative General Parameter Filter

Predictive lowpass and bandpass filters play an important role in numerous delay-constrained signal processing applications, especially in the area of 50/60 Hz power systems instrumentation. To cope with this demanding problem, Vainio et al. introduced the multiplicative general parameter (MGP) finite impulse response (FIR) filtering scheme in [20]. Since the line frequency tends to vary within a constrained interval, typically ±2%, adaptive filters should be used. In MGP-FIR the adaptation is achieved through adjusting the two MGPs. The coefficient values of the FIR basis filter do not change during the adaptation process. The purpose of the MGP-FIR is to extract the fundamental 50/60 Hz sinusoid signal among disturbances without causing any delay to this primary signal.

In a typical MGP-FIR, the filter output is computed as

$$y(n) = g_1(n)\sum_{k=0}^{N-1} h_1(k)x(n-k) + g_2(n)\sum_{k=0}^{N-1} h_2(k)x(n-k) \quad (2)$$

Where $g_1(n)$ and $g_2(n)$ present the adaptive MGP parameters, and $h_1(k)$ and $h_2(k)$ are the fixed coefficients of an FIR basis filter. Thus, the coefficients of the composite filter are $\theta_1(k) = g_1(n)\ h_1(k)$, $k \in [0, 1, \ldots, N-1]$, for the first MGP, and, $\theta_2(k) = g_2(n)\ h_2(k)$, $k \in [0, 1, \ldots, N-1]$, for the second MGP. An example of MGP-FIR with N=4 is shown in Fig. 4. Here N denotes the filter length. The adaptive coefficients, $g_1(n)$ and $g_2(n)$, are updated as follows

$$g_1(n+1) = g_1(n) + \mu e(n)\sum_{k=0}^{N-1} h_1(k)x(n-k) \quad (3)$$

$$g_2(n+1) = g_2(n) + \mu e(n) \sum_{k=0}^{N-1} h_2(k) x(n-k) \quad (4)$$

where μ is the adaptation gain factor and $e(n)$ is the prediction error between the filter output and the training signal, i.e., $x(n)$–y(n–p), p being the prediction step. The MGP-FIR has two adaptive parameters to adapt only to the phase and amplitude of the principal frequency. More degrees of freedom would allow the filter to adapt also to undesired properties, such as the harmonic frequencies. Our training signal $s(n)$ is defined as

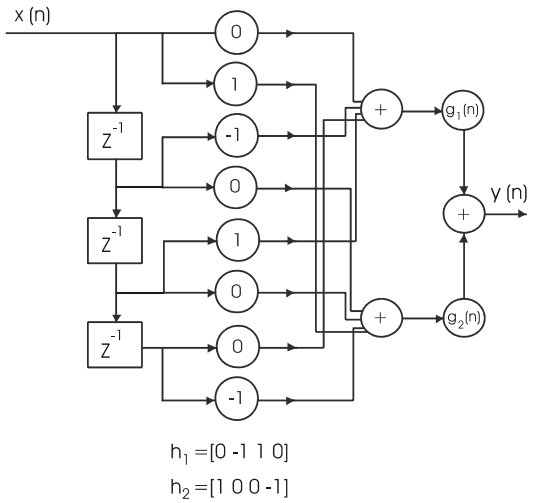

Figure 4. An example of MGP implementation, where N=4. Signal values (n-1) and (n-2) are connected to the first MGP and values (n) and (n-3) are connected to the second MGP with filter coefficients -1, 1, 1, and -1 respectively

$$s(n) = \sin(2 \cdot \pi \cdot 49 \cdot n) + \sum_{i=3,5,7,\ldots}^{15} 0.1 \cdot \sin(2 \cdot \pi \cdot i \cdot 49 \cdot n) + 0.004 \cdot r(n),$$
$$0 < n \le 300 \text{ samples} \quad (5)$$
$$s(n) = \sin(2 \cdot \pi \cdot 50 \cdot n) + \sum_{i=3,5,7,\ldots}^{15} 0.1 \cdot \sin(2 \cdot \pi \cdot i \cdot 50 \cdot n) + 0.004 \cdot r(n),$$
$$300 < n \le 600 \text{ samples}$$
$$s(n) = \sin(2 \cdot \pi \cdot 51 \cdot n) + \sum_{i=3,5,7,\ldots}^{15} 0.1 \cdot \sin(2 \cdot \pi \cdot i \cdot 51 \cdot n) + 0.004 \cdot r(n),$$
$$600 < n \le 900 \text{ samples.}$$

$r(n)$ denotes a uniformly distributed random value between –1 and 1. The duration of the training signal is 900 samples. This signal is divided into three parts, each of which contains 300 samples. These training signal blocks correspond to frequencies of 49 Hz, 50 Hz, and 51 Hz. The training signal constitutes thus of the nominal frequency sinusoid, odd harmonics up to the 15th with amplitudes 0.1 each, and white noise. The training signal is similar to that used in [20], and it mimics the line voltage/current with varying fundamental frequency, harmonics, and noise.

The basic idea of MGP-FIR filters is that all the samples of input delay line should be connected either to the first or to the second MGP, and no value should be left unused. Computational efficiency of these particular MGP filters arises from the fact that the filter coefficients are either -1, 0, or 1. Thus the number of multiplications in filtering operations is drastically reduced compared to a normal filtering operation using real-valued coefficients.

MGP-FIR is designed in the following way. The basis filter is optimized first either by applying traditional hard computing methods, or, as this paper presents, genetic algorithms. This optimization problem is discrete and no derivative information is available, so basically only some form of exhaustive search is an option to evolutionary computation methods. Next, the MGP-FIR is used in the actual application, and fine-tuning is left to the multiplicative general parameters.

The fitness of the chromosomes is evaluated using the fitness function:

$$fitness = \frac{K}{\max(NG49, NG50, NG51) \cdot (ITAE49 + ITAE50 + ITAE51)}. \quad (6)$$

For convenience, K is assigned a value of 10^7 to scale the output of the fitness function to be expressed in thousands. Terms $NG49$, $NG50$, and $NG51$ represent the white noise gain at a specific stage of the test input signal. This noise gain is calculated as

$$NG(n) = \sum_{k=0}^{N-1} [g_1(n) \cdot h_1(k)]^2 + \sum_{k=0}^{N-1} [g_2(n) \cdot h_2(k)]^2 . \quad (7)$$

$g_1(n)$ and $g_2(n)$ represent the first and second MGP at the end of a certain frequency period, respectively, whereas $h_1(n)$ and $h_2(n)$ denote the filter coefficients associated to the corresponding MGPs. In other words, $NG49$ is calculated using the MGP values after 300 samples, $NG50$ and $NG51$ are calculated after 600 and 900 samples, respectively, while the frequency of the training signal changing every 300 samples.

$ITAE49$, $ITAE50$, and $ITAE51$ stand for the Integral of Time-weighted Absolute Error (ITAE) for each of the three signal parts, respectively. These terms were added to the fitness function to smoothen the adaptation of the filter to the varying input signal characteristics. The ITAE is calculated as follows.

$$ITAE = \sum_{n=1}^{M} n \cdot e(n). \quad (8)$$

n is the sample index, and M stands for the sample number at the end of a specific frequency period, in this case 300, 600 and 900. $e(n)$ is the error between the output of the system and the pure primary sinusoid without any harmonics or noise. While evaluating the performance of the genetic algorithms the filter length was 40. Mutation in the case of MGP basis filter optimization was conducted so that a single gene was changed from 0 to 1/-1, 1 to 0/-1, or -1 to 0/1, in order to comply with the MGP theory.

C. Rastrigin's Function

Rastrigin's function is another well-known minimization problem. Rastrigins function has a single global minimum at zero.

$$f(x) = A \cdot n + \sum_{i=1}^{n} \left(x(i)^2 - A \cdot \cos\left(w \cdot x(i) \right) \right) \quad (9)$$

The following parameter values were used here

A=100, w=2π, n=50, $20 \leq x(i) \leq 30$.

D. Traveling Salesman Problem

Traveling salesman problem (TSP) is a well-known and much studied minimization problem. The problem describes a traveling salesman who has to visit a certain number of cities without having to visit the same city twice, in other words, the shortest closed path connecting all the cities has to be found. A number of common problems, among others, printed circuit board design [21] and robot path planning [22] can be converted to a TSP. In this paper we used a challenging 100-city TSP.

When solving a TSP using GA, special attention has to be paid to the crossover operator. A traditional single point crossover could easily create loops, which are unacceptable in proper solutions. In this case, we used a modification of the Grefenstette's greedy crossover [19]. The modified greedy crossover creates a solution by selecting a random city from either of the parents. After this, the next city is included by taking the closest city to the previous city in the parents. There are four candidate cities. If the closest city already exists in the solution, we take the second closest and so on. If all the four cities exist in the solution, we take a random city that does not yet exist in the solution.

V. Results

Increasing the complexity of an algorithm usually increases the possibility to modify the operation of the algorithm by tweaking a parameter here and there. However, as can be seen from the results, the 2PGA outperforms the reference SPGA reliably despite the reasonable initial parameter set assigned to it. In the following, we present the evaluation results of the various parameter configurations. Before presenting the actual results of our optimization runs, Table 2 sums up the various variables used in the text. To analyze the effect of the different parameters of the 2PGA only a single parameter was changed at a time, other values remaining at the default values presented in Table 3.

The Student's t-test [23] was conducted between the SPGA results and the 2PGA results to verify whether we really were dealing with different sample means. The performance of the reference SPGA in the optimization problems is shown in Table 4. Average fitness value over 50 runs was scaled to equal 1 for comparison purposes.

The population division point was considered first, and the results are shown in Tables 5-8. Also, the Student's t-test has been conducted to verify that the SPGA and the 2PGA sample means are actually different. The smaller the t-value the more likely it is that the two sample sets are from different distributions. The results show that if the division into separate populations occurs too late, the whole population may have converged beyond the limit after which the 2PGA is not capable of producing as good solutions as possible. In the light of these results, a proper point for dividing the population into separate entities could lie between 0 and 25% of the total number of generation to be run. But still, regardless of the division point, the 2PGA was able to outperform the reference SPGA. t-test results confirm the existence of differences resulting in values below 0.05, a value considered to clearly show difference between two sample set means.

Variable	Description
n_d	Population division point. Describes the point, after which the population is divided into separate subpopulations. Expressed as a percentage of the total number of generations evaluated before the division.
i_m	Migration interval. Describes the interval between the points in which the best chromosome from the plain population can be transferred to elite population. Expressed as a number of generation between two migration points.
es	Elite size. Describes the size of elite population in percentage of the total population size.
c_m	Migration condition. Describes the limit that has to be superseded in order for a plain population chromosome to enter the elite population. Described as a value relative to the best chromosome in the elite population.
m	Mutation probability. Describes the mutation probability before the division into separate subpopulations.
m_p	Mutation probability. Describes the mutation probability of the plain population after population division.
m_e	Mutation probability. Describes the mutation probability of the elite population after population division.
f_{ave}	Average fitness.
f_{std}	Fitness standard deviation.
f_{scaled}	Scaled fitness value. Scaled value 1 corresponds to the average fitness value of the reference SPGA.
c_{ave}	Average cost.
c_{std}	Cost standard deviation.
c_{scaled}	Scaled cost value. Scaled value 1 corresponds to the average cost value of the reference SPGA.

Table 2. Explanation of symbols used.

Problem	Ackley	MGP	Rastrigin	TSP
n_d	10%	10%	10%	10%
i_m	1	1	1	1
es	15%	15%	15%	15%
c_m	100%	100%	100%	100%
m	40%	10%	10%	20%
m_p	80%	20%	20%	40%
m_e	20%	5%	5%	10%

Table 3. Default parameters for the 2PGA.

Problem	f_{ave} / c_{ave}	f_{std} / c_{std}	f_{scaled} / c_{scaled}
Ackley	0.00145	0.00070	1
MGP	3657	310	1
Rastrigin	0.0669	0.0408	1
TSP	725.41	31.32	1

Table 4. Results for the reference SPGA.

n_d	c_{ave}	c_{std}	c_{scaled}	t-test
-	0.00145	0.00070	1	-
0	0.00080	0.00042	0.55	1.964E-07
10%	0.00082	0.00039	0.56	2.764E-07
25%	0.00099	0.00071	0.68	1.329E-03
50%	0.00110	0.00043	0.76	3.303E-03

Table 5. Results with different population division points (Ackley).

n_d	f_{ave}	f_{std}	f_{scaled}	t-test
-	3657	310	1	-
0	3770	229	1.03	0.0421
10 %	3772	231	1.03	0.0384
25 %	3810	175	1.04	0.0032
50 %	3741	256	1.02	0.1448

Table 6. Results with different population division points (MGP-FIR).

n_d	c_{ave}	c_{std}	c_{scaled}	t-test
-	0.0669	0.0408	1	-
0	0.0201	0.0152	0.30	1.819E-10
10%	0.0253	0.0207	0.38	1.176E-08
25%	0.0241	0.0168	0.36	3.003E-09
50%	0.0519	0.0531	0.77	1.148E-01

Table 7. Results with different population division points (Rastringin).

n_d	c_{ave}	c_{std}	c_{scaled}	t-test
-	725.41	31.32	1	-
0	711.38	34.66	0.98	0.0350
10 %	710.55	30.24	0.98	0.0169
25 %	719.21	30.52	0.99	0.3612
50 %	715.63	25.52	0.99	0.0878

Table 8. Results with different population division points (TSP).

Results concerning the elite population size are shown in Tables 9-12. Results here suggest that the elite population should be kept small, to assign more individuals to global optimization than to explore already good solutions. The t-test results clearly point out the differences in the sample means.

es	c_{ave}	c_{std}	c_{scaled}	t-test
-	0.00145	0.00070	1	-
5%	0.00100	0.00053	0.69	5.423E-05
25%	0.00096	0.00046	0.67	4.974E-05
45%	0.00115	0.00053	0.79	5.270E-04

Table 9. Results with different elite population sizes (Ackley).

es	f_{ave}	f_{std}	f_{scaled}	t-test
-	3657	310	1	-
5%	3846	201	1.05	0.1270
25%	3781	214	1.03	0.0071
45%	3780	230	1.03	0.0029

Table 10. Results with different elite population sizes (MGP-FIR).

es	c_{ave}	c_{std}	c_{scaled}	t-test
-	0.0669	0.0408	1	-
5%	0.0246	0.0171	0.37	4.422E-09
25%	0.0234	0.0163	0.35	1.835E-09
45%	0.0406	0.0341	0.61	7.071E-04

Table 11. Results with different elite population sizes (Rastrigin).

es	c_{ave}	c_{std}	c_{scaled}	t-test
-	725.41	31.32	1	-
5%	685.38	21.94	0.94	7.022E-11
25%	712.21	28.43	0.98	2.856E-02
45%	718.43	30.18	0.99	2.528E-01

Table 12. Results with different elite population sizes (TSP).

Result concerning migration interval are displayed in Tables 13-16. Results suggest that the populations should be given a little time to converge before individuals are exchanged between the populations. Too short convergence time between migration points may cause oscillation, and too long time, then again, may lead to premature convergence of the whole algorithm. As expected, the t-test values show that there exists difference in the means using different i_m parameter settings. However, for the TSP, using these migration intervals the 2PGA was in fact outperformed by the SPGA. This is likely due to the fact that the migration intervals for this kind of problem were too large. Migration intervals less than 2% produced better results.

i_m	c_{ave}	c_{std}	c_{scaled}	t-test
-	0.00145	0.00070	1	-
2%	0.00092	0.00049	0.64	0.0004
5%	0.00097	0.00038	0.67	0.0001
10%	0.00102	0.00047	0.70	0.0155

Table 13. Results with different migration intervals (Ackley).

i_m	f_{ave}	f_{std}	f_{scaled}	t-test
-	3657	310	1	-
2%	3742	261	1.02	0.0005
5%	3802	202	1.04	0.0221
10%	3810	184	1.04	0.0267

Table 14. Results with different migration intervals (MGP-FIR).

i_m	c_{ave}	c_{std}	c_{scaled}	t-test
-	0.0669	0.0408	1	-
2%	0.0299	0.0207	0.45	2.242E-07
5%	0.0255	0.0172	0.38	7.682E-09
10 %	0.0425	0.0472	0.64	6.777E-03

Table 15. Results with different migration intervals (Rastrigin).

i_m	c_{ave}	c_{std}	c_{scaled}	t-test
-	725.41	31.32	1	-
2%	755.77	39.07	1.04	4.645E-05
5%	760.42	32.95	1.05	4.109E-07
10%	767.85	34.94	1.06	6.054E-09

Table 16. Results with different migration intervals (TSP).

The conditions for migration between the populations were studied next. Results are shown in Tables 17-20. Results show that the migrating chromosome should not differ too much from the best chromosome in the elite population in order not to harm the local search of the elite population. t-test results confirm that there is difference in setting different values to the migration condition parameter. In minimization problems the condition the condition is larger than 1 and in maximization problems smaller than 1. When considering migration intervals, the SPGA outperformed 2PGA in the TSP case. Also in this case, the used values were likely out of reasonable range for this specific problem.

c_m	c_{ave}	c_{std}	c_{scaled}	t-test
-	0.00145	0.00070	1	-
1.01	0.00101	0.00049	0.69	0.0004
1.05	0.00117	0.00060	0.81	0.0203
1.10	0.00113	0.00039	0.77	0.0047

Table 17. Results with different migration conditions (Ackley).

c_m	f_{ave}	f_{std}	f_{scaled}	t-test
-	3657	310	1	-
0.99	3772	263	1.03	0.0487
0.95	3856	152	1.05	0.0001
0.90	3814	209	1.04	0.0039

Table 18. Results with different migration conditions (MGP-FIR).

c_m	c_{ave}	c_{std}	c_{scaled}	t-test
-	0.0669	0.0408	1	-
1.01	0.0330	0.0255	0.49	3.291E-06
1.05	0.0257	0.0174	0.38	9.001E-09
1.10	0.0262	0.0160	0.39	1.060E-08

Table 19. Results with different migration conditions (Rastrigin).

c_m	c_{ave}	c_{std}	c_{scaled}	t-test
-	725.41	31.32	1	-
1.01	744.24	34.00	1.03	5.117E-03
1.05	838.08	37.66	1.16	3.712E-29
1.10	832.38	37.42	1.15	9.820E-28

Table 20. Results with different migration conditions (TSP).

m	m_p	m_e	c_{ave}	c_{std}	c_{scaled}	t-test
10%	-	-	0.00145	0.00070	1	-
10%	10%	10%	0.00156	0.00072	1.08	4.459E-01
10%	20%	5%	0.00096	0.00055	0.66	1.814E-04
10%	5%	5%	0.00309	0.00145	2.13	5.364E-10

Table 21. Results with different mutation probability schemes (Ackley).

Finally, different mutation probability conditions were verified, and the results are shown in Tables 21-24. These results suggest that the elite population should undergo fewer mutations than the plain population in order to finally achieve good results. In the larger plain population, the number of mutations should be high. The low t-test value in the case of both population's mutation probabilities being lower than the reference SPGA mutation probability describes remarkable differences in the two sample sets. However, exceptionally in this case, the SPGA performs significantly better than the 2PGA due to the fact that the used mutation probability is insufficiently low.

m	m_p	m_e	f_{ave}	f_{std}	f_{scaled}	t-test
40%	-	-	3657	310	1	-
40%	40%	40%	3637	339	1.00	0.7608
40%	80%	20%	3857	165	1.06	0.0001
40%	20%	20%	3534	320	0.97	0.0534

Table 22. Results with different mutation probability schemes (MGP-FIR).

m	m_p	m_e	c_{ave}	c_{std}	c_{scaled}	t-test
10%	-	-	0.0669	0.0408	1	-
10%	10%	10%	0.1339	0.0986	2.00	3.602E-05
10%	20%	5%	0.0274	0.0190	0.41	3.501E-08
10%	5%	5%	0.4996	0.2428	7.46	3.714E-17

Table 23. Results with different mutation probability schemes (Rastrigin).

m	m_p	m_e	c_{ave}	c_{std}	c_{scaled}	t-test
20%	-	-	725.41	31.32	1	-
20%	20%	20%	747.21	29.21	1.03	5.320E-04
20%	40%	10%	708.49	26.63	0.98	4.266E-03
20%	10%	10%	773.61	44.31	1.07	1.312E-08

Table 24. Results with different mutation probability schemes (TSP).

Figures 5-8 show a comparison between the performance of the 2PGA and the reference SPGA solving the different test problems. The 2PGA results include both the best and the worst parameter setting in terms of the average fitness values. The results show that 2PGA usually outperforms SPGA when the parameter settings are reasonable. Figures 9-12 show the corresponding standard deviations. As can be seen, the standard deviation is usually lower using 2PGA than SPGA.

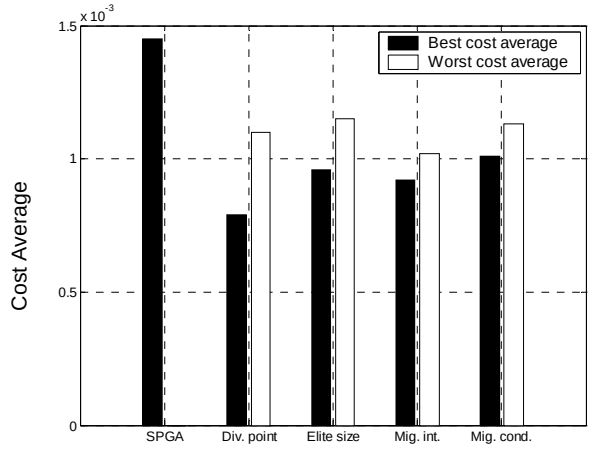

Figure 5. Comparison of the achieved cost value averages between the reference SPGA and 2PGAs with different parameters in the Ackley's function minimization problem.

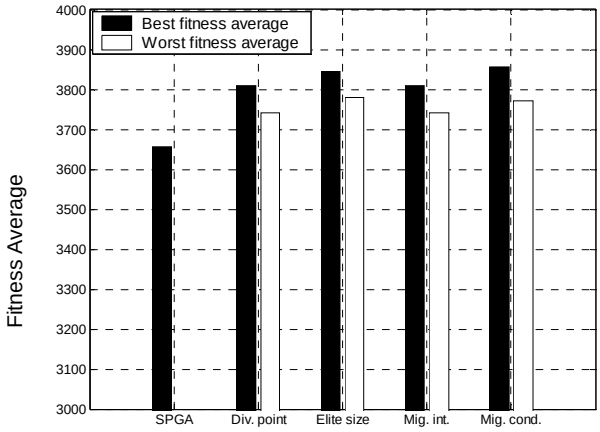

Figure 6. Comparison of the achieved fitness value averages between the reference SPGA and 2PGAs with different parameters in the MGP basis filter maximization problem.

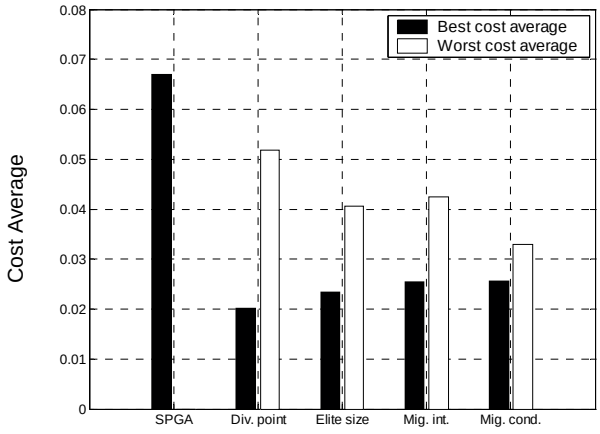

Figure 7. Comparison of the achieved cost value averages between the reference SPGA and 2PGAs with different parameters in the Rastrigin's function minimization problem.

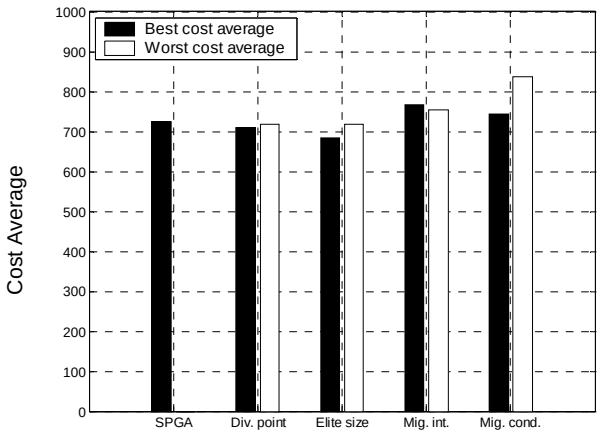

Figure 8. Comparison of the achieved cost value averages between the reference SPGA and 2PGAs with different parameters in the TSP minimization problem.

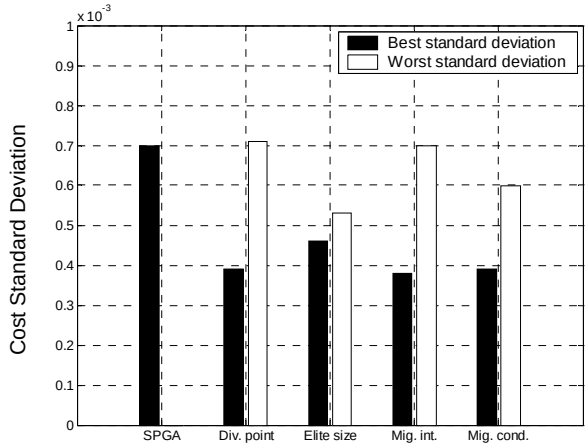

Figure 9. Comparison of the achieved standard deviations between the reference SPGA and 2PGAs with different parameters in the Ackley's function minimization problem.

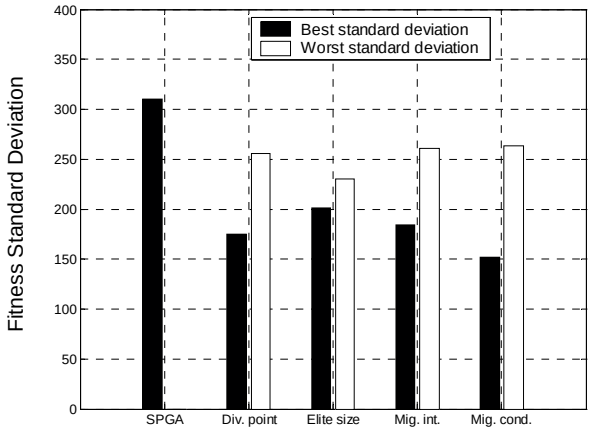

Figure 10. Comparison of the achieved standard deviations between the reference SPGA and 2PGAs with different parameters in the MGP basis filter maximization problem.

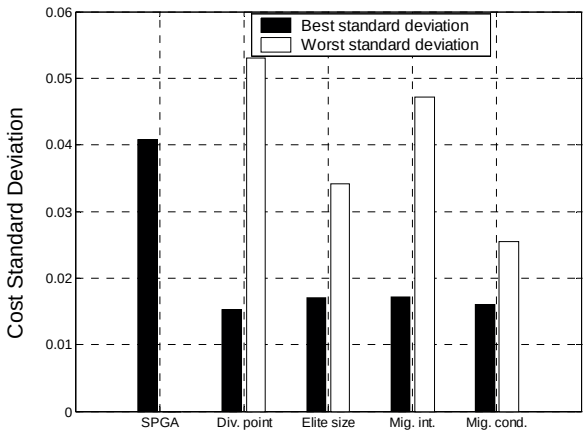

Figure 11. Comparison of the achieved standard deviations between the reference SPGA and 2PGAs with different parameters in the Rastrigin's function minimization problem.

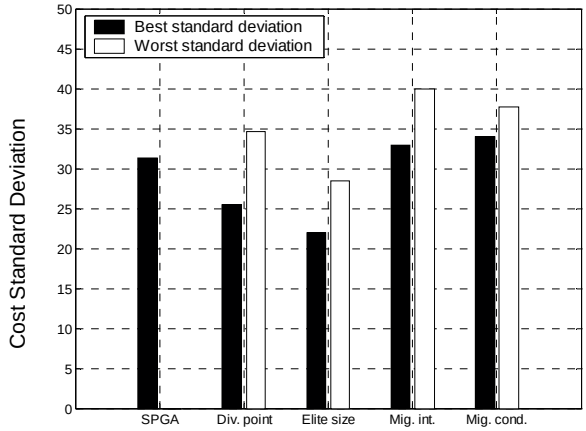

Figure 12. Comparison the achieved standard deviations between the reference SPGA and 2PGAs with different parameters in the TSP minimization problem.

Finally, Figs. 13-16 display the averaged convergence characteristics of the reference GA and the 2PGA schemes. The figures clearly show the better convergence characteristics of the 2PGA scheme compared to those of the reference GA. The reference parameter values, shown in Table 3, were used for the 2PGA calculations.

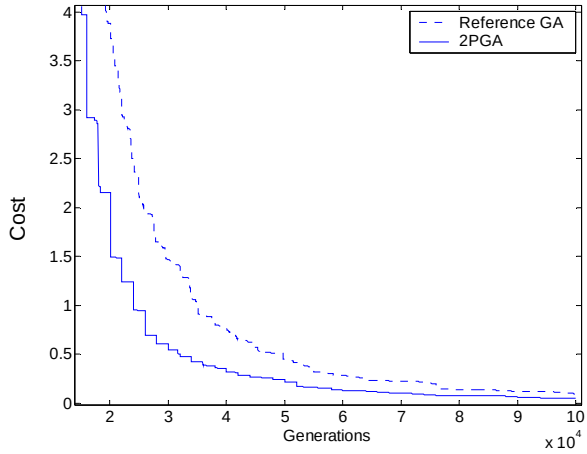

Figure 13. Convergence characteristic of the 2PGA and the reference GA in Ackley's function minimization.

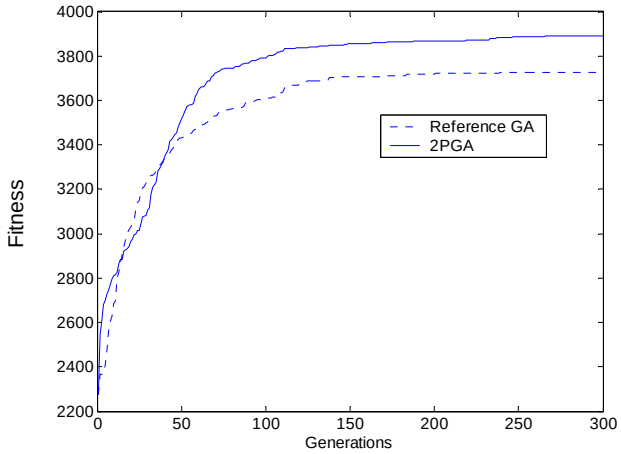

Figure 14. Convergence characteristic of the 2PGA and the reference GA in MGP-FIR maximization.

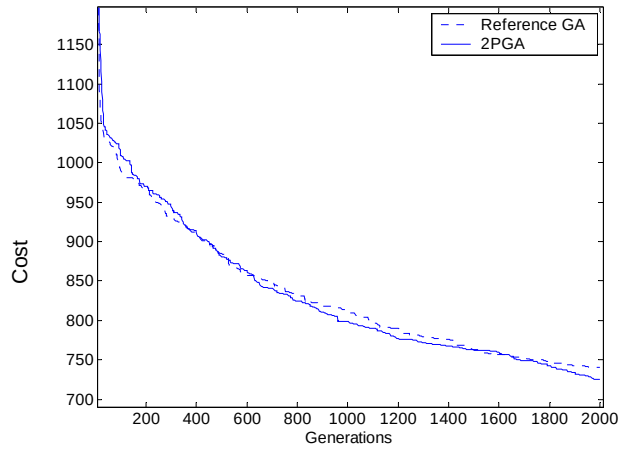

Figure 15. Convergence characteristic of the 2PGA and the reference GA in TSP minimization.

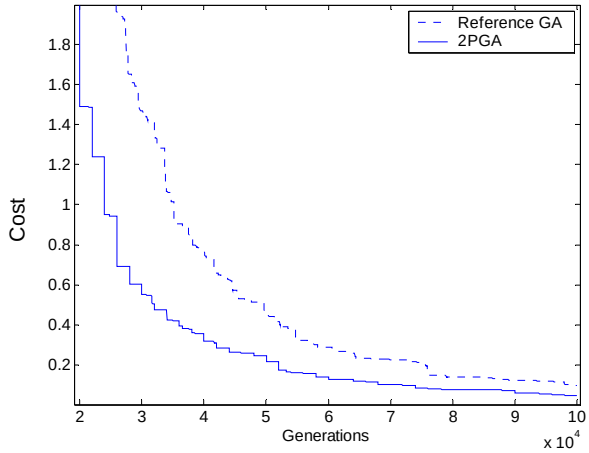

Figure 16. Convergence characteristic of the 2PGA and the reference GA in Rastrigin's function minimization.

Commonly, parameters within evolutionary algorithms are quite application specific and no strict guidelines can be given to achieve good results in a specific problem. We demonstrated above that almost regardless of the evaluated parameter settings our proposed 2PGA produces better results than the ordinary SPGA used as a reference. The performance of the reference SPGA can naturally be tuned for example using seeding, adaptive parameters, and such, but the same improvements would also benefit the 2PGA

scheme. Some guidelines on finding the proper parameter settings can, however, be given based on these results.

The total population should be divided into subpopulations before the algorithm has converged, thus eliminating the change of further improvement in fitness values. Experiments suggest that this division should take place before some 25% of the total number of generations is run. Intuitively, everybody cannot be elite, so the size of elite population should be limited to those individuals who posses the highest fitness or lowest cost. Elite size around 5% of the total population is favored by the simulation results.

Migration interval should be long enough for the populations to converge moderately, but not too much. Migration interval of 5-10% of the total number of generations has produced plausible solutions in our experiments. Migration condition sets the limits for the chromosomes from the plain population trying to enter the elite population. Results from the experiments point that the fitness of the plain population chromosome should roughly be at most 1.05 or 0.95 of the cost of the best chromosome in the elite population depending if we are dealing with minimization or maximization problem, respectively. This way, relatively moderate chromosomes cannot interfere the search of the local optimum.

Mutation probabilities are very application specific parameters, and only the ratio between the plain and elite population mutation probabilities can be given here. To stress the local search nature of the elite population the mutation probability should be kept rather small, whereas the individuals in the plain population should be more likely to undergo mutations while searching the whole solution space.

The comparison of SPGA and 2PGA results is somewhat problematic. Naturally, the number of function evaluations has to remain the same, as it does in this case. The number of mutations per run, however, rarely is exactly the same. This is likely the case when using for example adaptive mutation probabilities. It is generally not a problem, since the computational burden induced by a mutation operation is generally negligible to that of a solution evaluation. In terms of mutation, what our results show, is that when we find a mutation probability m giving us good results, we can get even better results using two populations, elite and plain, using mutation probabilities of $0.5m$ and $2m$, respectively.

It seems a feasible solution to pick the best parameters from the alternatives presented above, and see how such an algorithm performs. Unfortunately, the relations between the parameters are more complex than that, but picking up the best values for every parameter still produces competitive result.

The computational burden of 2PGA does not differ much from that of the reference SPGA, since both the algorithms use the same amount of crossovers and chromosome evaluations. The need for extra computation arises only during the initialization of the two populations and migration procedure later on. Tests show at maximum a 5% increase in computation time when using 2PGA scheme compared to that of a SPGA.

VI. Hierarchical 3PGA

The term two-population genetic algorithm implies that there could be even more populations than the previously considered plain and elite populations used in the 2GPA. The three-population GA, 3PGA, is a complex system and the performance comparison between the two-population and the single population GAs in not so straightforward. A careful consideration has to be made that the result are comparable, in other words, the amount of fitness function evaluations remains the same. The populations in the 3PGA are formed as in the two-population case explained in Section II. The migration from middle to elite and lower to middle population happens as described earlier in Section II related to 2PGA. Also, the chromosomes not selected as parents for next generation are transferred from elite to middle population, from middle to lower population or discarded. In Table 25 we present the population parameters for the 3PGAs.

Problem	Elite pop. size	Middle pop. size	Lower pop. size	Total pop. size
MGP	12	42	26	80
Ackley	16	52	32	100

Table 25. Population sizes for three-population 3PGAs.

To test the performance of the 3PGA, we implemented the algorithm to solve two test problems, the Ackley's function and the MGP filter design problem. The total population sizes in both the MGP basis filter and Ackley's function optimization are the same as in the respective SPGAs, as are also the number of reproduced individuals per generation. The parameters for the reference SPGAs can be seen in Table 26. The reproduction characteristics for both the MGP basis filter problem and the Ackley's function optimization problem are shown in Tables 27 and 28, respectively.

Problem	Pop. size	Parents	Offspr.	Discarded
MGP	80	16	16	16
Ackley	100	22	22	22

Table 26. The reproduction characteristics of the reference SPGAs.

Table 26 shows that when optimizing the MGP basis filter, for every generation 16 parents produce 16 offspring, which replace 16 worst chromosomes in the population. For Ackley's function optimization the corresponding figure is 22.

Population	Elite	Middle	Low
Population size	12	42	26
Parents	6	8	2
Offspring	6	8	2
Migrated lower	6	14	-
Discarded	-	-	16
Mutation probability	10%	20%	100%

Table 27. Reproduction and mutation characteristics of the three-population 3PGA for the MGP basis filter scheme for each generation.

Population	Elite	Middle	Low
Population size	16	52	32
Parents	8	10	4
Offspring	8	10	4
Migrated lower	8	18	-
Discarded	-	-	22
Mutation probability	20%	40%	100%

Table 28. Reproduction and mutation characteristics of the 3PGA for the Ackley's function scheme for each generation.

Similar migration scheme was used in the 3PGA as was in the 2PGA. Chromosomes could migrate to upper population if their fitness was higher than the fitness of the best chromosome in the upper population. The worst chromosome from the upper population was then migrated to lower population.

Results using 2PGA and 3PGA are not directly comparable, because different number of offspring was created on each generation. The results of the Reference SPGA with the same reproduction characteristics as the 3PGA in the case of MGP basis filter optimization are shown in table 29. The results are averages of 50 runs. The results for optimizing Ackley's function with the three-population MPGA are shown in Table 30.

Algorithm	f_{ave}	f_{std}
SPGA	3631	306
3PGA	3773	215

Table 29. Results of the 3PGA in the MGP basis filter maximization.

Algorithm	c_{ave}	c_{std}
SPGA	0.00172	0.00072
3PGA	0.00081	0.00041

Table 30. Results of the 3PGA in the Ackley's function minimization.

The results show that the 3PGA outperforms the corresponding SPGA. However, the 3PGA is a rather complex system to implement and it is obvious that the 2PGA outperforms the three-population counterpart in terms of programming convenience. Based on the experience gained programming 2PGAs and 3PGAs, and also considering the results achieved, we prefer using 2PGA instead of 3PGA. Hierarchical GAs with more than three populations have not been experimented with and are not convenient for problems with population sizes of the magnitudes dealt in this paper.

VII. Conclusion

In this paper, we have presented a hierarchical multipopulation genetic algorithm scheme. In our idea, the solution population is divided into multiple intermigrating entities based on their similar fitness characteristics. The proposed scheme aims to mimic the phenomena distinct to both human and animal populations, according to which individuals tend to gather in groups of homogeneous fitness characteristics. Evaluation results show that the suggested system is capable of producing competitive performance compared to the traditional single population genetic algorithm. As our scheme requires negligible amount of additional calculation, it could well add to the performance of every algorithm using some sort of single population genetic algorithm as a part of it.

Our multipopulation genetic algorithm is somewhat more complex than the regular single population genetic algorithm, and it also has more tunable parameters. Our results show, however, that almost regardless of the reasonable parameter settings, the proposed scheme is capable of producing competitive results in optimization problems in terms of fitness values and convergence reliability.

The proposed 2PGA scheme cannot replace hybrid algorithms equipped with hill-climbing methods or other fusion methods, but it offers a valuable addition to the performance of a standard genetic algorithm, thus making it attractive basic platform for any application involving a GA, or any other evolutionary computation method.

Acknowledgments

The authors whish to thank the referees for their insightful comments. This research work was funded by the Academy of Finland under Grant 214144.

References

[1] T. Bäck. *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, New York, NY, 1996.

[2] D. B. Fogel. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence,* IEEE Press, Piscataway, NJ, 2000.

[3] R. L. Haupt and S. E. Haupt. *Practical Genetic Algorithms*, John Wiley & Sons, New York, NY, 1998.

[4] M. Nowostawski and R. Poli. "Parallel genetic algorithm taxonomy", in *Proceedings of the 3rd International Conference on Knowledge-Based Intelligent Information Engineering Systems,* Adelaide, Australia, pp. 88-92, 1999.

[5] E. Alba and M. Tomassini. "Parallelism and evolutionary algorithms", *IEEE Transactions on Evolutionary Computation*, VII (5), pp. 443-462, 2002.

[6] E. Cantú-Paz. "Markov chain models of parallel genetic algorithms", *IEEE Transactions on Evolutionary Computation*, IV (3), pp. 216-226, 2000.

[7] J. Branke, A. Kamper, and H. Scheck. "Distribution of evolutionary algorithms in heterogeneous networks," *Genetic and Evolutionary Computation*

Conference, LNCS 3102, Springer, Berlin, Germany, pp. 923-934, 2004.

[8] F. Herrera and M. Lozano. "Gradual distributed real-coded genetic algorithms", *IEEE Transactions on Evolutionary Computation*, IV (1), pp. 43-63, 2000.

[9] V. Slavov and N. Nikolaev. "Genetic algorithms, sublandscapes and subpopulations," in *Foundations of genetic Algorithms 5*, W. Banzhaf and C. Reeves (Eds.), Morgan Kaufmann, San Francisco, CA, 1999.

[10] V. S. Gordon and D. Whitley. "Serial and parallel genetic algorithms as function optimizers", in *Proceedings of the 5th International Conference on Genetic Algorithms*, San Mateo, CA, pp. 177-183, 1993.

[11] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*, Boston, MA: Kluwer Academic Publishers, 1989.

[12] P. J. Darwen and X. Yao. "Speciation as automatic categorical modularization", *IEEE Transactions on Evolutionary Computation*, I (2), pp. 101-108, 1997.

[13] J. Werfel, M. Mitchell, and J. P. Crutchfield. "Resource sharing and coevolution in evolving cellular automata", *IEEE Transactions on Evolutionary Computation*, IV (4), pp. 388-393, 2000.

[14] H. Handa, N. Baba, O. Katai, T. Sawaragi, and T. Horiuchi. "Genetic algorithm involving coevolution mechanism to search for effective genetic information", in *Proceedings of the IEEE International Conference on Evolutionary Computation*, Indianapolis, IN, pp. 709-714, 1997.

[15] T. Arita and A. Ojika. "Generation of color patterns based on the interactions between predators and prey", in *Proceedings of IEEE International Conference on Evolutionary Computation*, Nagoya, Japan, pp. 291-294, 1996.

[16] M. Sun and A. W. Johnson. "Interval branch and bound with local sampling for constrained global optimization", *Journal of Global Optimization*, XXXIII, pp. 61-82, 2005.

[17] M. Laguna and M. Marti. "Experimental testing of advanced scatter search designs for global optimization of multimodal functions", *Journal of Global Optimization*, XXXIII, pp. 235-255, 2005.

[18] A. R. Hedar and M. Fukushima. "Simplex coding genetic algorithm for the global optimization of nonlinear functions," in *Multi-Objective Programming and Goal Programming*, T. Tanino, T. Tanaka and M. Inuiguchi (Eds.), Springer-Verlag, Berlin-Heidelberg, pp. 135-140, 2003.

[19] J. Grefenstette, R. Gopal, R. Rosmaita, and D. Gucht. "Genetic algorithms for the traveling salesman problem", in *Proceedings of the Second International Conference on Genetic Algorithms*, Lawrence Eribaum Associates, Mahwah, NJ, pp. 160-168, 1985.

[20] S. J. Ovaska and O. Vainio. "Evolutionary-programming-based optimization of reduced-rank adaptive filters for reference generation in active power filters", *IEEE Transactions on Industrial Electronics*, LI (4), pp. 910-916, 2004.

[21] K. Fujimura, O.-C. Kwaw, and H. Tokutaka. "Optimization of surface component mounting on the printed circuit board using SOM-TSP method", in *Proceedings of the 6th International Conference on Neural Information Processing,* Perth, Australia, pp. 131-136, 1999.

[22] W. Sheng, N. Xi, M. Song, and Y. Chen. "Optimization in Automated Surface Inspection of Stamped Automotive Parts", in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2, Lausanne, Switzerland, pp. 1850-1855, 2002.

[23] J. S. Milton and J. C. Arnold. *Introduction to Probability and Statistics: Principles and Applications for Engineering and the Computing Science*, McGraw-Hill, New York, NY, 1990.

Author Biographies

JARNO MARTIKAINEN received an M.Sc. and an Lic.Sc. degree in electrical engineering from the Helsinki University of Technology in 2003 and 2006, respectively. He is currently working towards the Ph.D. degree. His research interests include evolutionary algorithms in time-constrained optimization and the fusion of soft computing techniques.

SEPPO J. OVASKA (M'85–SM'91) received an M.Sc. degree in electrical engineering from Tampere University of Technology, Finland, an Lic.Sc. degree in computer science and engineering from Helsinki University of Technology, Finland, and a D.Sc. degree in electrical engineering from Tampere University of Technology in 1980, 1987, and 1989, respectively.

He is currently a Professor in the Department of Electrical and Communications Engineering, Helsinki University of Technology. Before joining Helsinki University of Technology in 1996, he was an Associate/Full Professor in the Department of Information Technology, Lappeenranta University of Technology, Finland. From 1980 to 1992, he held engineering, research, and R&D management positions with Kone Elevators and Nokia Research Center, both in Finland and in Kentucky. In the summer of 1999, he was a Visiting Scientist at Muroran Institute of Technology, Japan; in the summers of 2000 and 2001, at Virginia Polytechnic Institute and State University; in the summers of 2002–2004, at Utah State University; and in the summer of 2005, at University of Passau, Germany. His research interests are in soft computing, fault diagnosis, signal processing, and control. During his career, he has authored or coauthored over 200 papers in peer-reviewed journals and international conferences. He edited "Computationally Intelligent Hybrid Systems: The Fusion of Soft Computing and Hard Computing" (Wiley – IEEE Press, 2004), and holds nine patents in the area of systems and control.

Dr. Ovaska is an Associate Editor for the *IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews* and the *IEEE Transactions on Neural Networks*. He was the Founding General Chair of the 1999 IEEE Midnight-Sun Workshop on Soft Computing Methods in Industrial Applications. In addition, he was the General Chair of the 5th Online World Conference on Soft Computing in Industrial Applications (2000). Dr. Ovaska is a recipient of two Outstanding Contribution Awards of the IEEE Systems, Man, and Cybernetics Society.