# Accelerating Genetic Schema Processing Through Local Search

Tarek A. El-Mihoub
Computer Engineering Department
University of Tripoli
Tripoli, Libya
tmihoub@tripoliuniv.edu.ly

Adrian Hopgood
Sheffield Business School
Sheffield Hallam University
Sheffield, UK
a.hopgood@shu.ac.uk

Ibrahim A. Aref
Computer Engineering Department
University of Tripoli
Tripoli, Libya
i.aref@ec.uot.edu.ly

***Abstract*— Achieving a balance between the exploration and exploitation capabilities of genetic algorithms is a key factor for their success in solving complicated search problems. Incorporating a local search method within a genetic algorithm can enhance the exploitation of local knowledge but it risks decelerating the schema building process.**

**This paper defines some features of a local search method that might improve the balance between exploration and exploitation of genetic algorithms. Based on these features a probabilistic local search method is proposed. The proposed search method has been tested as a secondary method within a staged hybrid genetic algorithm and as a standalone method. The experiments conducted showed that the proposed method can speed up the search without affecting the schema processing of genetic algorithms. The experiments also showed that the proposed algorithm as a standalone algorithm can, in some cases, outperform a pure genetic algorithm.**

***Keywords—hybrid genetic algorithm; Lamarckian search; Lamarckian learning; memetic search; local search; schema processing.***

## I. INTRODUCTION

A genetic algorithm is a population-based search and optimization method that mimics the process of natural evolution [1]. Genetic algorithms have received significant interest in recent years and are being increasingly used to solve real-world problems. The power of genetic algorithms comes from their ability to combine both exploration and exploitation in an optimal way [2]. The exploration and the exploitation abilities of a genetic algorithm can be enhanced by incorporating a local search method [3]. The combination can accelerate the search towards the global optimum [4]. This constructive form of cooperation between a genetic algorithm and a local search can produce an effective and efficient search algorithm [5].

However, the interference between the two methods can also be destructive. This destructive interference can be in the form of a premature convergence in the Lamarckian search, which may force the fast convergence speed of this strategy to be sacrificed for high quality solutions of other learning strategies [6]. It can also be in the form of destroying good local solutions. The staged hybrid genetic algorithm [7] and the use of carefully chosen control parameters [8] have been suggested to minimize such a destructive interference.

Genetic populations contain a huge amount of search information which can be utilized in different ways. For example, genetic information was used in the PMBGA algorithms [9] and quantum-inspired genetic algorithms [10] that were proposed as alternatives to the pure genetic algorithm to overcome some of the difficulties in solving real-world problems. Genetic information has been utilized to adapt the control parameters of genetic algorithms to improve the search performance [11]. Search information has also been used to decide on performing a genetic search or a local search in some hybrids [12] and on the optimal fraction of individuals that should perform a local search [13].

However, this valuable genetic search information is rarely used by the local search method. Due to the lack of positional information in the genetic information, advanced local methods are unable to improve its speed. The use of clustering techniques can be of high cost and dependent on the problem since these forms of local search usually work on the phenotype space. In addition to this, advanced local search methods usually consume a considerable number of function evaluations, which can aggravate the hybrid's loss caused by any destructive interference.

To sum up, some of the good features of a search method that may enable it to be incorporated in a genetic algorithm in an effective and efficient way are:

- The ability to avoid any disruption to the genetic algorithm schema processing.
- The ability to reuse the available genetic search information in efficient way.
- The cost of the search should be low to reduce the loss caused by any destructive interference.

In this paper, a simple probabilistic algorithm based on the features described above is proposed and tested as a secondary search method and as a standalone method.

## II. THE PROPOSED SEARCH ALGORITHM

The proposed algorithm is a probabilistic method that works on the genotype space. It modifies the initial solution based on a group of solutions from the genetic population. This can improve the initial solution in accordance with the global view captured by the genetic search in order to minimize any conflict between the two search methods. The partial global aspect of the search method can be controlled by the group size and the mechanism of selecting the group members. This method is also characterized by its low cost, which helps to minimize the loss of the hybrid's time in the case of any undesirable interference.

The algorithm assumes that each gene contributes uniformly to the fitness of the solution. Based on this assumption, the search method compares the genetic structure and the fitness of the initial solution with the structures and the fitness of a group of solutions. Depending on the differences in both the structure and the fitness between this solution and the group members, the solution structure is modified in the direction of improving its fitness score. The new solution is evaluated and then inserted back into the population regardless of its new fitness.

### *A. The search mechanism*

The algorithm starts with an initial solution and a randomly selected group of solutions from the current genetic population.

The algorithm assumes that the value of each gene in the initial solution represents the probability of that gene having the value of one. It also assumes that the produced set of probabilities represents the initial probabilities of having the value of one in each gene of the optimal solution's structure.

This set of initial probabilities is modified according to the differences in the genetic structure and the fitness between the initial solution and the group members in order to estimate the optimal solution structure. An increase in the fitness score of a group member compared to the initial solution accompanied by a change in gene value from '0' in the initial solution to '1' in the group member means increasing the probability associated with that gene. The probability is increased by a value that is proportional to the increase in fitness score in order to bias the initial solution toward a better structure. However, if that increase in the fitness is accompanied by a change from '1' to '0', the associated probability is decreased by the same value. A decrease in the fitness score in the previous cases will result in decreasing the probability in the first case and increasing it in the second by a value that is proportional to the absolute value of the difference in fitness score between the group member and the initial solution.

The algorithm compares every member of the group with the initial solution, in turn, and adjusts the genes' probabilities in the way described above. The resulting set of genes' probabilities is compared against a set of randomly generated numbers over the range [0, 1]. If the gene probability is less than or equal to the random number generated, the value of that gene is set to one otherwise it is set to zero. Then, the new structure is evaluated and returned as the new improved solution.

## III. EMPIRICAL METHODOLOGY

In order to evaluate the performance of the proposed search method within a global genetic algorithm, a set of experiments has been conducted. In these experiments, the performance of a hybrid genetic algorithm that utilizes the proposed search method is compared with the performance of the pure genetic algorithm. To maximize the interference between the two search methods, the hybrid genetic algorithm performs a local search iteration after each global genetic iteration. In order to assess the amount of disruption that this algorithm can cause on the schema processing, the algorithm is applied to every individual of the genetic population and the pure Lamarckian learning strategy was used. The decrease in the number of experiments that converge to the global optimum together with the convergence speed compared to the pure genetic algorithm are used as measures of the disruption to the schema processing.

The optimization problems were chosen to evaluate the basic assumption of the proposed method on the hybrid performance. Since the proposed method assumes that each gene of the solution contributes uniformly to the solution fitness, problems with different marginal fitness contribution of their genes were used.

In these experiments, two empirical methodologies were followed. The classical methodology, which uses a set of known test functions to evaluate the performance of an algorithm, was used. The other methodology, which employs a problem generator [14] for studying the behavior of evolutionary algorithms, has also been used to evaluate the proposed algorithm.

Following the classical methodology, three test functions were used to assess the hybrid's and the proposed search algorithm's performance. The first one is a uniformly scaled fitness function, which is the MaxOne problem, and the second is the BinInt problem [15], which has an exponentially scaled fitness structure. The third test function is Schwefel's function [16], which is a non-linear multimodal function.

In addition to the classical empirical methodology, the problem generator methodology has been followed. A problem generator is an abstract model capable of producing randomly generated problems on demand. The use of problem generators allows experimentation over a randomly generated set of problems rather than on a few hand-chosen examples. This can increase the predictive power of the results for a problem class as a whole.

The multimodal problem generator has been slightly modified and used. Instead of generating the locations of equal peaks, it generates the locations of a number of local optima.

A multimodal exponential problem generator has been proposed and used. The multimodal exponential problem generator is similar to the multimodal generator. The multimodal exponential generator also generates O local optima. The evaluation of a solution is carried out, first, by locating the nearest local optimum in Hamming space. Then, the nearest local optimum is used to generate a Hamming distance string of the current solution. The produced Hamming

string is inverted and evaluated using the following fitness function:

$$f(x) = A_{fo} \sum_{i=1}^{l} x_i 2^{l-i} \qquad x_i \in \{1,0\} \qquad (1)$$

where $A_{fo}$ is an amplitude factor associated with each local optimum and xi represents the value in the inverted Hamming distance string. The value of the amplitude factor is from the range (0, 1]. The value of this factor for the global optimum is 1.0.

## IV. Simulations and Analysis

The experiments that have been conducted aimed to evaluate the performance of the proposed algorithm within a hybrid. The performance is measured by investigating the search method's effect on the population size and the population convergence speed. The algorithm is also evaluated by studying its effect on the schema processing of the global genetic algorithm. In the last set of experiments the search algorithm is evaluated as a stand-alone algorithm by comparing its performance to the pure genetic algorithm and a hybrid combining them.

### *A. Minimum population size*

The first set of experiments was conducted to investigate the effect on the population size requirements by hybridizing the proposed algorithm. The experiments used the bisection method [17] to find the minimum population size required.

The hybrid used the simple elitist genetic algorithm with binary tournament selection, uniform crossover, and no mutation as the global search method. The crossover rate was set to 1.0. The proposed algorithm was used as an embedded search method that is performed by each individual of the population after each global genetic iteration. The experiments have been conducted using different group sizes. The group sizes tested were {0, 2, 4, 8, 16, 32}. A hybrid with a group size of 0 is identical to the pure genetic algorithm. In these experiments, two values of probability factors (0.5 and 1.0) were tested.

The experiments were conducted to determine the minimum population sizes required for solving the MaxOne problem with a string length of 120 bit and the BinInt problem with a string length of 30. The results of both problems show that using a probability factor of 0.5 significantly reduces the minimum population size required for group sizes of more than 2, Fig. 1. This reduction in is accompanied by a decrease in the convergence speed. However, using the statistical t-test shows that the decrease in the convergence speed is insignificant and the decrease in the population size is significant.

For a probability factor of 1.0, the experiments of the MaxOne problem show that there is a significant increase in the convergence speed with insignificant increase in the population size. However, the experiments of the BinInt problem show a significant increase in the convergence speed with a considerable decrease in the population size.

The poor performance of a hybrid when utilizing a group size of 2 can be explained in the terms of the partial view provided by the group size. The partial global view provided by that group size is very narrow and not enough to avoid a destructive interference between the two search methods especially when using a probability factor of 1.0, where the improved solution is more dependent on the partial view gained than the initial solution structure.

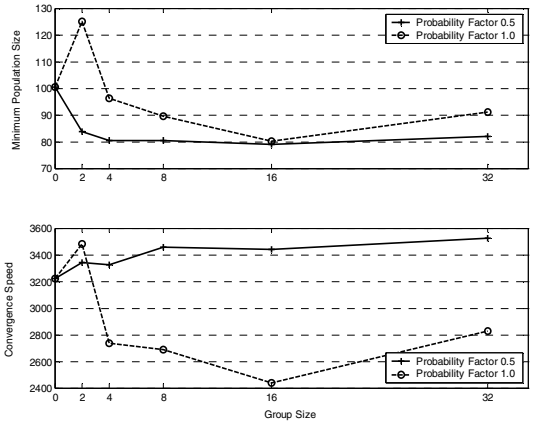


Fig. 1. The Effect of Group Size and the Probability Factor on the Hybrid's Minimum Population Size and Convergence Speed of the BinInt Problem.

The results show that utilizing the proposed search algorithm within a genetic algorithm using a suitable group size can improve the genetic performance in terms of the population size, the convergence speed or both of them..

### *B. Effect on Schema processing*

The aim of this set of experiments was to assess the disruption to the schema processing caused by the new algorithm. This can be accomplished by comparing the number of times each algorithm converges to the global optimum with that of the pure genetic algorithm.

The first set of experiments was conducted using the multimodal exponential problem generator. The experiments were carried out using five and ten randomly generated local optima with amplitude factors of {0.2,0.4,0.6,0.8,1.0} and {0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0} respectively. The chromosome length was set to 30. An elitist generational genetic algorithm with binary tournament selection and uniform crossover was used as the global search algorithm. The population size was set to 100. The crossover rate was 1.0 and the mutation probability was 0.000333 ($p_m$=1.0/(lxN)). Experiments were run for a complete convergence of the population or a maximum of 10,000 function evaluations.

The results of the two problems were similar. They show that all the tested group sizes and probability factors, except the combination of a group size of two and a probability factor of 1.0, show no decrease in the number of experiments that converged to the global optimum (Fig. 2). This means no disruption for the schema processing was induced by incorporating the proposed search method with the specified group sizes and probability factors. The graph also shows that using a probability factor of 1.0 increased the convergence speed of the population to the global optimum for all group

sizes tested. The diagram also shows that a probability factor of 1.0 is more suitable for large group sizes, whereas a value of 0.5 seems more suitable for small sizes. This ability of a large group size to capture a wide view of the search space when combined with a strong effect on the initial solution can direct it in the right direction. However, using the same strong effect with a small group size which provides a very narrow view of the search space can misguide the search.

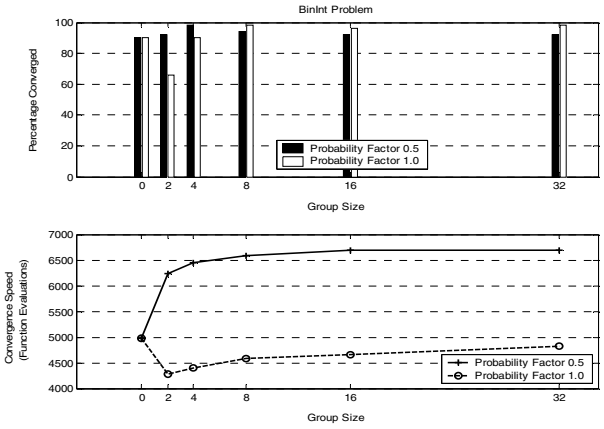


Fig. 2. Effect on the Schema Processing when Solving the BinInt Problem.

In another experiment, the algorithms were used to optimize Schwefel's function with ten variables. The chromosome length of each variable was 16 bits. The algorithm used a global simple elitist genetic algorithm with binary tournament selection and 2-point crossover. The crossover rate is 0.6 and the mutation rate is 0.000001. The population size was 300. The stopping criterion was a maximum number of function evaluations, which was set to 300,000.

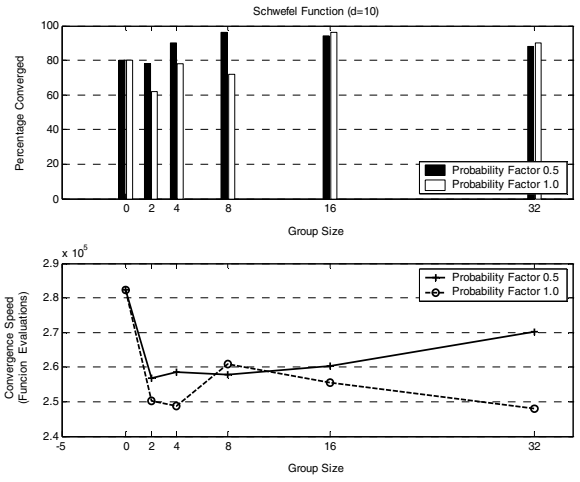


Fig. 3. Solving Schwefel's Function.

The results of these experiments, as depicted in Fig. 3, show that with a probability factor of 0.5 and for a group size greater than or equal to four there is always an increase in the number of times of finding the global optimum. However, for a probability factor of 1.0, the increase occurred with group sizes of 16 and 32. The results also show that this improvement in the number of times of reaching the global optimum is always accompanied by an improvement in the convergence speed. A larger group size is needed to capture a good partial view of the search space compared to the sizes needed in the previous experiments due to the nature of the fitness landscape which is more complicated than the previous problems.

### *C. The search method as a stand alone algorithm*

The proposed algorithm was tested as a standalone algorithm to optimize three multimodal functions. It has been used to optimize the Schwefel function with ten variables, the multimodal problem and the multimodal exponential problem. The algorithm used a population size equal to that used by the pure genetic algorithm and the hybrid. The performance of the algorithm was compared with that of the pure genetic algorithm and a hybrid combined them.

The results of employing the proposed search algorithm on Schwefel's fitness function demonstrated that the pure genetic algorithm outperformed the proposed algorithm using different group sizes and probability factors. Fig. 4 shows the average fitness of the population as a function of the number of function evaluations for the algorithm as a standalone and a secondary algorithm compared to that of the pure genetic algorithm. The algorithm shown in this figure used a group size of 8 and a probability factor of 0.5. The graphs demonstrate that despite the poor performance of the proposed algorithm as a standalone algorithm compared to the pure genetic algorithm, their combination outperformed either of them. The graphs depict the convergence of 50 experiments of each algorithm.

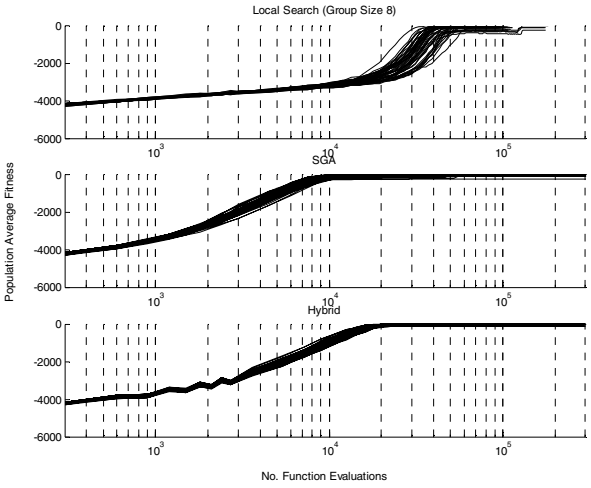


Fig. 4. The Convergence Details of the Schwefel's Function.

A multimodal exponential problem generator with five local optima was used to evaluate the performance of the algorithm. The amplitude factors were set to {0.2, 0.4, 0.6, 0.8, 1.0}. The experiments demonstrate that the search algorithm, in most cases, performed worse than the pure genetic algorithm. In a few cases however, the proposed algorithm outperformed the pure genetic algorithm in terms of the number of experiments that converged to the global optimum. It also outperformed the hybrid in terms of the convergence speed. The convergence of the population for the three algorithms as a

function of the number of function evaluations is depicted in Fig. 5. The group size used was 16 and the probability factor was set to 1.0.

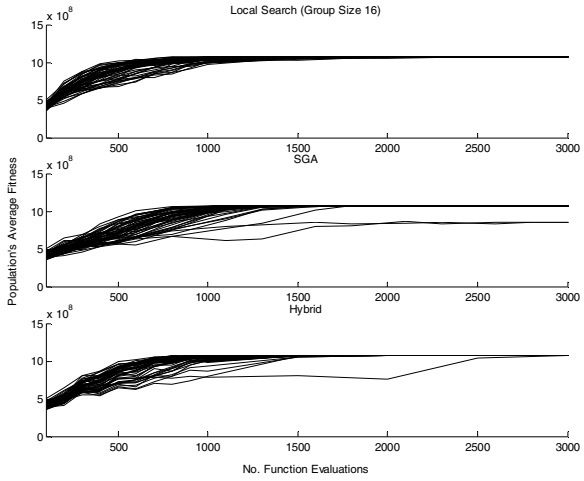


Fig. 5. Comparing the Convergence of the Proposed Algorithm with the Pure Genetic Algorithm and the Hybrid on 5-modal Exponential Problem.

In a third set of experiments, the multimodal problem generator with five local optima has also been used to evaluate the search method performance. The string length of solutions and the population size were set to 100. The amplitude factors were set to {5.0, 4.0, 3.0, 2.0, 1.0}.

An elitist generational genetic algorithm with binary tournament selection and two-point crossover was used as the global search algorithm. The crossover rate was 0.6 and the mutation probability was 0.0001. Experiments were run for a complete convergence of the population or a maximum of 100,000 function evaluations.

The results of this set of experiments were encouraging. They show that the proposed algorithm outperformed the pure genetic algorithm using different population sizes when combined with a probability factor of 0.5. The algorithm converged faster to the global optimum than the pure genetic algorithm. It also outperformed a hybrid that combined it with the genetic algorithms using a group size of 2 and 4 (Fig. 6). However, the standalone algorithm showed poor performance when using a probability factor of 1.0. The algorithm with a probability factor of 1.0 can guide the search to non-optimal solutions. The probability of guiding the search towards a non-optimal solution increases as the group size decreases. However, a hybrid with a probability factor of 1.0 outperformed the pure genetic algorithm, and both the standalone algorithm and the hybrid with a probability factor of 0.5 for group sizes of 8, 16 and 32.

Fig. 7 compares the performance of the algorithm using different group sizes and probability factors. The graph demonstrates the fast convergence speed associated with a probability factor of 1.0. The graph also shows that this convergence can be towards non-optimal solutions. There is a decrease in the number of experiments that converged to non-optimal solution accompanied with an improvement in the convergence speed as the group size increases. In contrast to the probability factor of 1.0, the probability factor of 0.5 shows a decrease in convergence speed as the group size changes from 2 to 32 with the ability to find the exact global optimum in all cases.

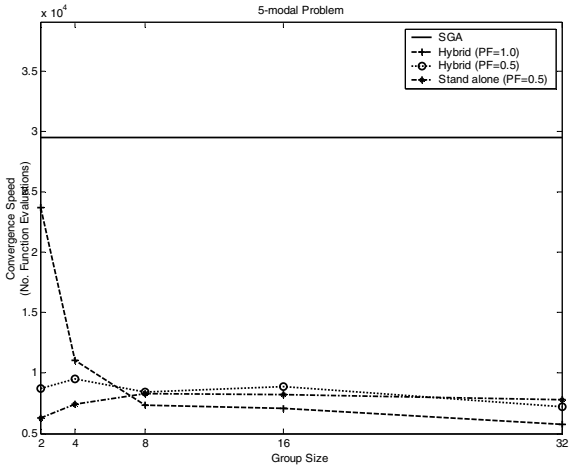


Fig. 6. Convergence Speed of the Proposed Algorithm as a Stand Alone Algorithm.

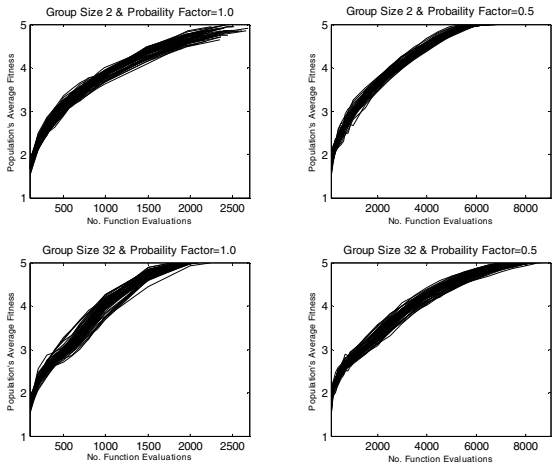


Fig. 7. Comparing the Effect of the Probability Factor and the Group Size on Algorithm Performance.

Fig. 8 compares the population convergence speed of the algorithm as a standalone optimization technique with that of the pure genetic algorithm and a hybridization of them. This graph compares an algorithm with a probability factor of 1.0 and a group size of 32. The graphs show that a hybridization can get the best out of the two search methods. It produced an algorithm that was able to find the global optimum in all the experiments, in contrast to the standalone algorithm which can miss that optimum some times. The hybrid was able to employ the ability of the pure genetic algorithm to reach the global optimum in all experiments and utilize the fast convergence speed of the secondary method to produce an effective and efficient algorithm.

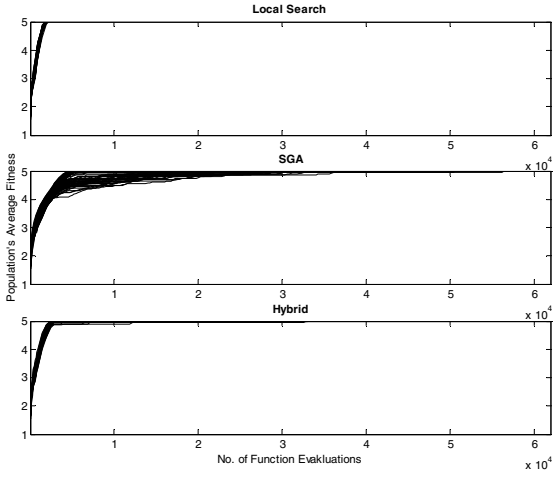


Fig. 8. Comparing the Convergence of the Proposed Algorithm with the Pure Genetic Algorithm and the Hybrid on 5-modal Problem.

## V. CONCLUSION AND FUTURE WORK

The proposed algorithm shows good performance as a standalone search method on problems with a uniformly scaled fitness function. However, the standalone algorithm and the pure genetic algorithm are outperformed by their hybrid on problems with non-uniformly scaled fitness function.

The basic assumption of the proposed algorithm, which states that each gene contributes uniformly to the fitness of the solution, can explain the good performance of the algorithm, as a standalone optimization technique, on the multimodal generator problem compared to the poor performance on the other two problems.

However, the encouraging performance of the algorithm as a secondary search method, even when applied to non-uniformly scaled fitness functions, can be explained as follows. The genes of non-uniformly scaled fitness functions converge at different rates [15]. The most important genes converge towards their optimal value before the less important genes. The proposed algorithm concentrates on the differences in the population structure and fitness to modify the non-identical genes. The algorithm does not modify the identical genes. These non-identical genes in the non-uniformly scaled problems are the genes that converge at a slower rate than the identical genes that have been converged to their optimal value as a result of the genetic search. The algorithm uses a sample of the genetic population to determine the genes that have not been converged yet. This sample involves the initial solution and a selected group of solutions. The accuracy of the algorithm in determining the converged genes increases as the sample size increases. This ability of determining the already converged genes in the population reduces the possibility of disrupting the genetic schema processing. This, in turn, can reduce the probability of facing premature convergence problems and can accelerate the search towards the global optimum. The good performance of the hybrid that uses large group sizes can thereby be explained.

One of the possible ways of improving the performance of the proposed algorithm is to use a variable group size for each iteration. It is also possible to set the values of the probability factor depending on the group size used. This can be done in accordance with the findings of the experiments of this paper. These experiments show that high probability factors are suitable for large groups and low factors are more suitable to small groups. The probability factor can be made adaptable to the group size using this approach.

## REFERENCES

[1] D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, 1989.

[2] D. Beasley, D. R. Bull and R. R. Martin, "An Overview Of Genetic Algorithms : Part 1, Fundamentals," University Computing, vol. 2, no. 15, pp. 58-69, 1993.

[3] A. Hopgood, Intelligent Systems for Engineers and Scientists, 3rd ed., CRC Press, 2012.

[4] T. A. El-Mihoub, A. Hopgood, L. Nolle and A. Battersby, "Hybrid Algorithms : A review," Engineering Letters, vol. 3, no. 2, pp. 12-45, 2006.

[5] A. Boriskin, M. Balaban, O. Y. Galan and R. Sauleau, "Efficient approach for fast synthesis of phased arrays with the aid of a hybrid genetic algorithm and a smart feed representation," in Phased Array Systems and Technology (ARRAY), 2010 IEEE International Symposium on, 2010.

[6] D. Whitley, V. S. Gordon and K. Mathias, "Lamarckian evolution, the Baldwin effect and function optimization," in Parallel Problem Solving from Nature—PPSN III, Springer Berlin Heidelberg, 1994, pp. 5-15.

[7] L. D. Whitley, K. Mathias, C. Stock and T. Kusuma, "Staged Hybrid Genetic Search For Seismic Data Imaging," in Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on, 1994.

[8] C. R. Houck, J. A. Joines, M. G. Kay and J. R. Wilson, "Empirical Investigation Of The Benefits Of Partial Lamarckianism," Evolutionary Computation, vol. 5, no. 1, pp. 31-60, 1997.

[9] M. D. Pelikan and D. E. Goldberg, "A Survey of Optimization by Building and Using Probabilistic Models," IlliGA, 1999.

[10] K. Han and J. H. Kim, "Quantum-Inspired Evolutionary Algorithm For A Class Of Combinatorial Optimization," IEEE Transactions On Evolutionary Computation, vol. 6, no. 6, pp. 580- 593, 2002.

[11] A. Eiben, R. Hinterding and Z. Michalewicz, "Parameter Control In Evolutionary Algorithms," Evolutionary Computation, IEEE Transactions on, vol. 3, no. 2, pp. 124-141, 1999.

[12] F. G. Lobo and D. E. Goldberg, "Decision Making in a Hybrid Genetic Algorithm," in IEEE International Conference on evolutionary Computation, Piscataway, 1997.

[13] T. A. El-Mihoub, A. Hopgood, L. Nolle and A. Battersby, "Self-adaptive Baldwinian search in hybrid genetic algorithms," in 9th Fuzzy Days International Conference on Computational Intelligence, Dortmond, 2006..

[14] K. A. De Jong, M. A. Potter and W. M. Spears, "Using Problem Generators to Explore the Effects of Epistasis," in the Seventh International Conference on Genetic Algorithms, East Lansing, 1997.

[15] D. Thierens, D. E. Goldberg and A. G. Pereira, "Domino Convergence, Drift, And The Temporal-Salience Structure Of Problems," in IEEE International Conference on Evolutionary Computation, Anchorage, 1998.

[16] H. Mühlenbein, M. Schomisch and J. Born, "The Parallel Genetic Algorithm as Function Optimizer," Parallel Computing, vol. 17, pp. 619-632, 1991.

[17] T. A. El-Mihoub, A. Hopgood, l. Nolle and A. Battasbry, "Performance of hybrid genetic algorithms incorporating local search," in 18th European Simulation Multiconference, Magdeburg, 2004.