2nd International Conference on Autonomous Robots and Agents
December 13-15, 2004 Palmerston North, New Zealand

Performance Study of a Multi-Deme Parallel Genetic Algorithm with Adaptive Mutation

B.T. Skinner, H.T. Nguyen, D.K. Liu
Faculty Of Engineering,
University of Technology, Sydney, Australia
Brad.Skinner@uts.edu.au, Hung.Nguyen@uts.edu.au, Dikai.Liu@uts.edu.au

Abstract

This paper presents a performance study of a parallel, coarse-grained, multiple-deme Genetic Algorithm (GA) with adaptive mutation. The effect of varying migration period and number of subpopulations upon the GA is evaluated. Using common unimodal and multimodal objective functions, this study measures the convergence velocity and solution quality for the proposed genetic algorithm. In this paper, we briefly survey previous work in static and adaptive control parameters and parallel genetic algorithms (PGAs). Experimental results show that migration period and the number of subpopulations significantly influence the performance of the genetic algorithm.

Keywords: Adaptive parameters, multi-population, parallel GA, distributed computation.

1 Introduction

Historically, genetic algorithms begin with a single population of candidate solutions of the problem solution space. The initial population is generated at random and then evolved towards better solutions by applying genetic operators such as selection, crossover, mutation, and inversion. Through the recursive application of genetic operators the GA stochastically searches for an optimal solution within a continuous or discrete domain. It is accepted that a GA with single population and static parameters performs this search strategy through *implicit parallelism* as suggested by the Schemata Theorem [1]. In this study, two, four and eight serial GAs evolve isolated populations in parallel. Once each isolated population reaches a defined generation the fittest individual within each population migrates to a master who calculates the fittest individual among the migrated candidates. The fittest overall individual then migrates back into each of the isolated populations.

In Section 2 we briefly survey previous research in static and dynamic adaptation of crossover (P_c) and mutation (P_m) probabilities, parallel and distributed GAs. Section 3 describes the proposed genetic algorithm. Section 4 describes the experiment and numerical objective functions used for performance evaluation. Section 5 presents experimental results and Section 6 concludes the findings.

2 Previous Research

2.1 Static P_c and P_m

It is well recognised [1]-[16] in genetic algorithms that crossover (P_c) and mutation (P_m) probabilities are important and commonly used operators. The likelihood of performing crossover of two or more parent chromosomes is governed by P_c and the likelihood of randomly mutating one or more genes within the chromosome is governed by P_m. In the static approach, P_c and P_m are predetermined and remain fixed throughout the GA evolution [2]. Since P_c and P_m are fundamental operators for GAs, considerable research has been done in tuning of crossover and mutation rates before the GA is used to solve the particular optimisation problem. Parameter tuning is usually performed by trial-and-error or some search algorithms are used to determine the more appropriate values [3]. Typically, this involves running numerous systematic tests and trying to find a cause (parameter value) and effect (GA performance) relationship.

In De Jong's pivotal 1975 dissertation [4] he examined, among other things, parameter values of a traditional GA with one point crossover and bit mutation. He obtained static values of $P_c = 0.6$ and $P_m = 0.001$ that guarantee good GA performance for a test bed of five objective functions. Grefenstette [5] proposed a Meta-GA, which considers many GA control parameters at one time. The Meta-GA is a two-level GA with the upper-level GA determining the optimal set of static control parameters for the lower-level GA, which performs the actual optimisation task of interest. Grefenstette determined static values of $P_c = 0.95$ and $P_m = 0.01$.

2.2 Adaptive P_c and P_m

In the domain of artificial genetic search, as in nature, the process of evolution is an inherently dynamic and adaptive process. The use of static control parameters, such as P_c and P_m in a GA contrasts this evolutionary idea. Much research has been conducted into adaptive control parameters of GAs with the aim to reduce the amount of time required to perform static parameter tuning, minimise mistakes and sub-optimal performance as a result of user selection error and to optimise parameter values at different stages of the evolutionary process. Maintaining a balance and precedence between exploration and exploitation is a major goal in developing strategies to adapt GA control parameters. In GA search, *exploration* is the ability to discover new regions of the solution space while trying to locate the optimum. Exploration is directly related to the population diversity. Conversely, *exploitation* is the ability to converge to the optimum, after discovering the region locating the optimum. Exploration should, during the early phase of GA evolution, precede exploitation to prevent premature convergence to a sub-optimal solution.

Hinterding *et al,* developed a comprehensive classification of adaptation techniques [6]. They discuss two main types of adaptation: static and dynamic. Static was discussed in Section 2.A. Dynamic can be partitioned into deterministic, adaptive and self-adaptive.

2.2.1 Dynamic Deterministic

Deterministic dynamic adaptation of GA control parameters modifies the parameter value according, usually, to some decreasing value as GA evolution progresses. Mutation probability is appealing because it allows for adaptively tuning the degree of exploration versus exploitation in the search.

Thierens [7] describes a deterministically decreasing function of mutation probability that Bäck and Schultz tested.

$$p_m(t) = \left(2 + \frac{n-2}{T-1} \cdot t\right)^{-1} \tag{1}$$

where, $p_m(t) = P_m(t)$ is the time-dependent mutation probability at generation step t, $t \in \{0,1,...,T-1\}$, T is the maximum number of generations in a single epoch and n is the problem dimensionality. Maximum mutation probability is $p_m(0)$=0.5.

2.2.2 Dynamic Adaptive

This type of adaptation relies on *feedback* information from the genetic algorithm to determine the direction and/or magnitude of modification to the GA control parameter values. Buczak [8], Wu [9], Lee [3], Ho [10] describe a powerful adaptive method developed by Srinivas and Patnaik. In this approach, P_c and P_m are adapted simultaneously; P_c and P_m are kept large for individuals whose fitness values are small; creating a strong exploration characteristic among individuals with low fitness. Alternatively, P_c and P_m values are kept small for individuals whose fitness values are large; favouring the exploitation characteristic among individuals with high fitness. High fitness solutions are protected, while solutions with sub-average fitness are totally disrupted. Ho *et al*, proposed a Probabilistic Rule-based Adaptive Model (PRAM) that can be used to adapt mutation probability, crossover probability or both in co-evolving population genetic algorithms [10].

2.2.3 Dynamic Self-Adaptive

This technique follows the evolutionary spirit in so much as modifying the control parameter values by evolving them. The parameters to be adapted are encoded into the genotype (resulting in a larger chromosome) of the individual and undergo mutation and recombination together with the chromosomes. The dynamic self-adaptation technique exploits the indirect link between 'favourable' control parameters and individual fitness values, as there is *no* direct feedback mechanism on the performance of parameter values. Due to limited success in early efforts of dynamic self-adaptation of crossover and mutation probabilities, Bäck and Schutz proposed a self-adaptive scheme for binary strings following the principles from the continuous domain [11].

2.3 Parallel Genetic Algorithms

Increasing demands have been placed upon GAs from engineering, science, and financial disciplines to search larger landscapes with multiple objective functions coupled with larger population sizes. This need creates an ongoing requirement for implementations that provide quick and flexible experimentation. Serial genetic algorithms are excellent candidates for parallelisation, given their inherent property of *implicit parallelism*, enabling them to evolve, in parallel, a population of individuals. Moreover, parallel implementations of GAs are easily scalable to larger populations, providing good potential to exploit massively parallel and distributed hardware. Cantu Paz [12] provides a detailed survey of three fundamental parallel genetic algorithms (PGAs): Global Single-Population Master-Slave, Single-Population Fine-Grained and Multiple-Population Coarse-Grained.

2.3.1 Global single population master-slave PGAs

Global single-population PGAs are based upon the master-slave architecture. The master (server) process hosts a single, panmictic population and the slave processes (clients) perform fitness evaluations and/or apply genetic operators to each individual in parallel. The most common operation parallelised, particularly with large populations, is gene decoding and calculation of the objective function for fitness evaluation. The master process assigns a predefined or dynamically determined subset of the population to each slave process. The slave processes decode the chromosome genotype and calculate the fitness of each individual within their assigned population subset and return the results to the master process. Studies of synchronous and asynchronous communication between master and slaves have been performed. The master-slave architecture does not require additional genetic operators; hence the approach does not modify the fundamental search behaviour of the genetic algorithm. This PGA model is suited to SIMD[1] shared-memory and distributed-memory computer systems [13].

2.3.2 Single population fine-grained PGAs

In this approach, parallelisation is performed on the level of individuals (fine-grained parallelism), resulting in a large number of interacting parallel processes of low complexity. In the fine-grained model (or neighbourhood model) a single population evolves, with each individual placed in a processing cell of a spatial structure, usually a planar grid that limits the interactions between individuals. Selection and crossover operations are applied only between neighbouring individuals. Typically massively parallel computers arrange their processing elements in a two dimensional grid topology, enabling individuals to occupy a single processing element of the grid topology. This PGA model is suited to massively parallel MIMD[2] computer systems with a large number of local processors and high-speed communications bus.

2.3.3 Multi-population coarse-grained PGAs

The set of partners which any individual in a population may mate with is termed *deme*. If the deme for all members of a population is equal to the entire population then the population is said to be *panmictic*. Any individual may mate with any other [13]. The simplest example of distributed population PGA is modelled on a network of isolated islands, each being panmictic. With isolated populations, evolution becomes more rapid and less likely to get stuck as isolation increases within any population [13]. In terms of genetic algorithms, premature convergence becomes less likely since the distributed, parallel, and independent nature of the search becomes better able to escape from unfit local minima. An important characteristic of multi-deme PGAs is the use of a number of relatively large subpopulations and the concept of migration between subpopulations. Migration is characterised by four factors: period of migration, number of migrants, the selection and replacement strategy of individuals.

The underlying network topology, which determines interconnection between the demes, is an important factor in the performance of a PGA because it determines the cost of migration and the rate at which good solutions disseminate to the other demes. The majority of multi-deme PGAs use synchronous migration, which means that migration occurs at periodic intervals. Asynchronous migration between demes removes the simplistic and artificial periodicity, although is more difficult to implement.

3 Multi-deme, PGA with adaptive mutation

The PGA is multiple-deme and coarse-grained with loosely-coupled, static subpopulation sizes of 200 individuals (*popsize*). All subpopulations evolve in isolation starting from an initial random population of candidate solutions. All subpopulations maintain two non-overlapping populations, with each new generation replacing the old to help maintain population diversity (generational model).

For all subpopulations, the *tournament selection* method is used to select a fixed number of *tour* individuals chosen randomly from the population. The fittest individual from the *tour* is then selected for mating. This process occurs twice with two, different parents chosen for mating. The tournament selection method allows selective intensity and population diversity to be adjusted through setting of the *tour* size, ($2 \le tour \le popsize$). Population diversity decreases and selective pressure increases with increasing tour size. Multi-Point crossover is performed with static probability of (P_c=0.95). Four crossover sites (*xsites*=4) are chosen at random with no duplicates. Segments of chromosome, lying between the crossover sites are then exchanged between the two parents to produce two new offspring. The section between the start of the chromosome and the first crossover site is not exchanged. The disruptive nature of multi-point crossover appears to encourage the exploration of the search space, rather than favouring the convergence to highly fit individuals early in the search, thus making the search more robust [11]. Bit-flip mutation is performed with the adaptive probability given by equation (2).

[1] Single Instruction, Multiple Data
[2] Multiple Instruction, Multiple Data

$$p_m(t) = \left(\frac{n \cdot l}{20} + \frac{2 \cdot n \cdot l}{T-1} \cdot t \right)^{-1} \quad (2)$$

where, $p_m(t)$ is the temporal mutation probability at generation step t, $t \in \{0,1,...,T-1\}$, T is the maximum number of generations in a single epoch, n is the problem dimensionality, and l is the gene bit length (chromosome length is nl). Initially, a large $p_m(0 \le t \le 0.45T)$, can be beneficial in the early stage of evolution for aiding the exploration of the search space. Alternatively, small $p_m(0.45T \le t \le T)$ can be beneficial in the later stage of evolution to protect highly fit individuals from random mutations, since they usually have 'more to lose' in variation of the chromosome, thus retaining good solutions [14]. The genes of each child chromosome are decoded into corresponding phenotype values, for direct use in objective function evaluation and determination of fitness. Each chromosome contains a number of genes corresponding to the number of function dimensions in each test function – (3)F_1(n=30), (4)F_2(n=20), and (5)F_3(n=30).

Migration occurs at a rate given by the migration period, (0|5|10|20|40). No migration occurs for a single population (migration period = 0). All subpopulations evolve until the number of generations is equal to the migration period. The fittest individual in each subpopulation is sent (migrated) to a master process for evaluation as illustrated in Figure 1. The master process, which also contains a local subpopulation and participates in migration, determines the fittest individual among those individuals that migrated. The overall fittest individual is then sent back to each population replacing the least-fittest individual in the local population as illustrated in Figure 1. All subpopulations continue to evolve fitter individuals until the maximum number of generations has been reached.

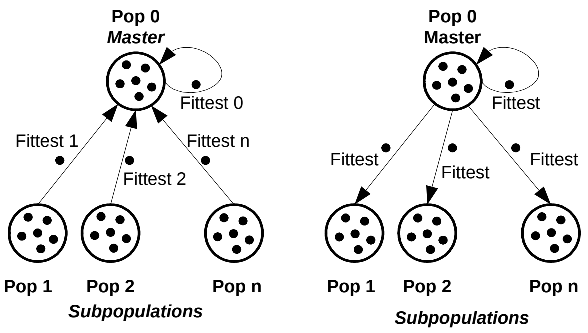

Figure 1: Synchronous Migration between loosely-coupled subpopulations

Each subpopulation is allocated to a single processing node of a computing cluster, described in 4.3. Processor and memory resources are not shared between subpopulations. Subpopulations execute in isolation with a *single* individual exchanged periodically and synchronously.

4 Experiment Description

To facilitate an empirical comparison of migration period and number of subpopulations on GA performance, a test environment is provided in the form of a small set of idealised objective functions, expressed in closed and analytical form.

4.1 Numerical Test Functions

The Sphere Function [4], [15], [10] is a continuous, strongly convex, unimodal function. Rastrigin's Function [10], [16] and Ackley's Path Function [4], [15] are continuous, highly multimodal functions. The global minimum for each objective function is 0.

The performance criteria for the genetic algorithm are *convergence velocity* and *solution quality*. Convergence velocity is a measure of the number of objective function evaluations required to obtain the optimum or quasi-optimum solution. Solution quality represents the value of the objective function, with higher quality solutions closer to the optimum or quasi-optimum value.

Sphere Function

$$F_1(x) = \sum_{i=1}^{n} \left(x_i^2 \right) \quad (3)$$

where, $-5.10 \le x_i \le 5.10$, n =30, $F_1(0)$=0, i=1:n

Rastrigin's Function

$$F_2(x) = n \cdot A + \sum_{i=1}^{n} \left(x_i^2 - A\cos\left(2\pi x_i\right) \right) \quad (4)$$

where, $-5.10 \le x_i \le 5.10$, A=10, n=20, $F_2(0)$ = 0, i=1:n

Ackley's Path Function

$$F_3(x) = 20 + e - 20 \cdot \exp\left(-0.2\sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n} x_i^2} \right) - \exp\left(\frac{1}{n} \cdot \sum_{i=1}^{n} \cos\left(2\pi x_i\right) \right) \quad (5)$$

where,$-32.8 \le x_i \le 32.8$,n=30, e=exp(1),$F_3(0)$=0, i=1:n

4.2 Genetic Algorithm Parameters

There is a complex, non-linear relationship between GA parameters which influences behaviour [2]. To minimise such effects, many GA parameters are kept constant for the duration of the experiment. Experiments were conducted using the set of parameters listed in Table 1.

Synchronous migration periods of 5, 10, 20, and 40 generations are used in conjunction with population values of 1, 4, and 8. No migration is used for the single population case. Gene lengths of 33-bits and

49-bits are used to provide high resolution and minimise quantisation error between possible solutions. Population diversity and selection intensity is controlled using the *Tour* size parameter.

Table 1: PGA Parameter Set

	Sphere Function	**Rastrigin's Function**	**Ackley's Function**
Generations	400	400	600
Subpopulation Size	200	200	200
# Subpopulation	1,4,8	1,4,8	1,4,8
Total Population Size	200,800, 1600	200,800, 1600	200,800, 1600
Subpopulation Type	Non-overlap	non-overlap	Non-overlap
Migration Period	0,5,10,20, 40	0,5,10,20,40	0,5,10,20,40
# Migrants	1	1	1
Migrant Selection	Fittest	Fittest	fittest
Migrant Reinsertion	least-fit	least-fit	least-fit
Gene Size (bits)	33	33	49
Dimensions	30	20	30
Chromosome Length (bits)	990	660	1470
Tour Size	5	3	6
P_m	Adaptive(2)	Adaptive(2)	Adaptive(2)
P_c	0.95	0.95	0.95
# Crossover Sites	4	4	4
Migration Packet Payload (bytes)	2256	1536	3236

Each experiment terminates after 80000, 80000 and 120000 objective function evaluations for the objective function (3), (4) and (5) respectively. Results are averaged over 50 independent experiments.

4.3 Cluster Computing Environment

All experiments were performed on a Linux computing cluster with specifications in Table 2. The PGA was coded in C++ and employed the Message Passing Interface Specification V2.0 for communication between GA subpopulations running on separate cluster nodes.

Table 2: Cluster Computing Hardware and Software Environment

Computing Environment Component	**Description**
Number of Nodes Used	1, 4, 8
Processor Type and core Speed	Intel Pentium 4 @ 3.0GHz
Front-side Bus Bandwidth	800MHz
DRAM capacity and bandwidth	2GB DDR @ 400MHz
Network Type and Bandwidth	Ethernet 1000Mbps
Network Switching Type	Gigabit Switching Fabric
Network Protocol	TCP/IP
OS Kernel Type and Version	Linux 8.0 (2.4.20-19.8)
MPI Type and Version	LAM 6.5.6 / MPI 2
Compiler Type and Version	mpiCC and gcc (3.2)
Coding Language Standard	ISO C++

5 Experimental Results

The GA performance is shown in Figure 2 to Figure 7 for a fixed number of subpopulations and varying migration periods. The (0-gen) curve, in each figure, represents a single population with no migration and is the baseline performance. The graphs use a log-linear scale, to display the behaviour of the GA for the entire run.

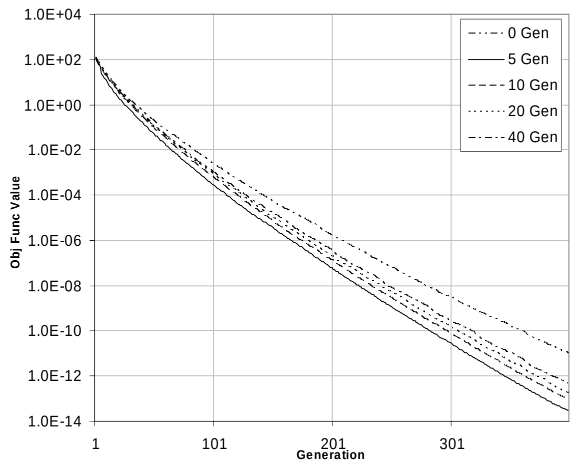

Figure 2: Sphere Function with 4-subpopulations and variable migration period.

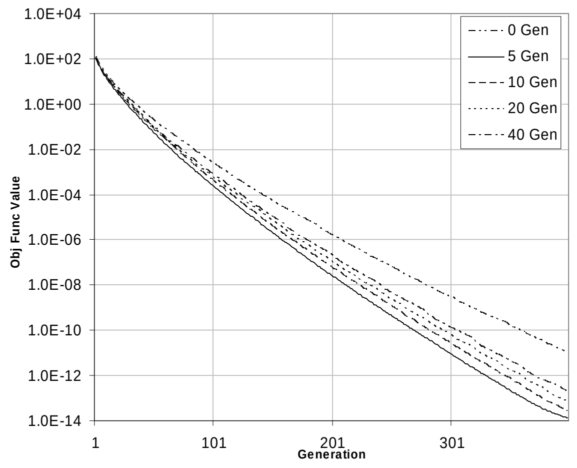

Figure 3: Sphere Function with 8-subpopulations and variable migration period.

A single population with no migration produces the lowest convergence velocity and solution quality. A migration period 5 generations produces the greatest convergence velocity and solution quality throughout the entire run for both 4- and 8-subpopulations of the Sphere Function (3) in Figure 2 and Figure 3.

Results for Rastrigin's Function (4) are shown in Figure 4 and Figure 5. A single population with no migration produces the lowest convergence velocity and solution quality (6.9116). With 4-subpopulations the GA is unable to converge past (0.1592) for migration periods of 20 and 40 generations.

Results for Ackley's Path Function (5) are shown in Figure 6 and Figure 7. A single population with no migration gives the lowest convergence velocity and solution quality (1.142). A migration period of 5 generations provides better convergence velocity and solution quality throughout the entire run for both 4- and 8-subpopulations.

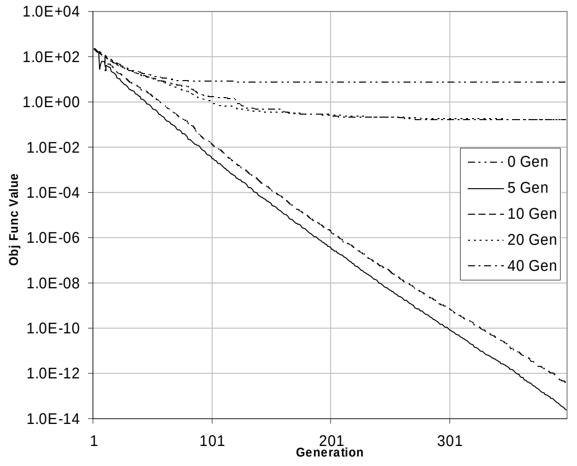

Figure 4: Rastrigin's Function with 4-subpopulations and variable migration period.

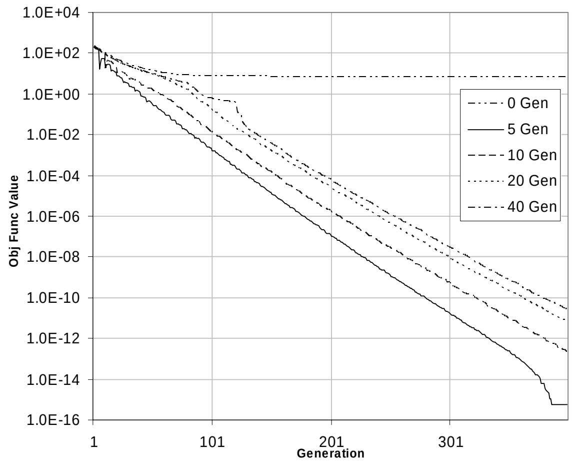

Figure 5: Rastrigin's Function with 8-subpopulations and variable migration period.

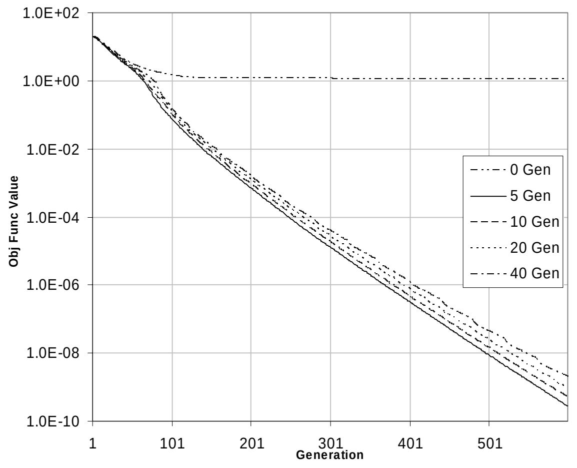

Figure 6: Ackley's Path Function with 4-subpopulations and variable migration period.

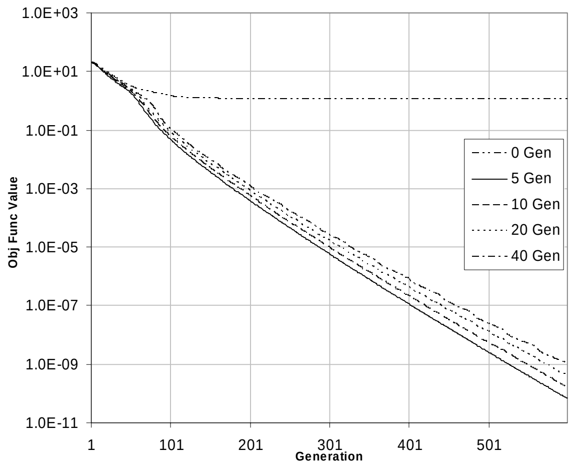

Figure 7: Ackley's Path Function with 8-subpopulations and variable migration period.

Experimental results suggest that 8-subpopualtions provides fastest convergence velocity and higher solution quality for each test function, compared with 4-subpopulations. This result is independent of migration period above 0.

6 Conclusion

This paper presented a performance evaluation of a multi-deme, parallel genetic algorithm with adaptive mutation probability. Using a small set of common unimodal and multimodal objective functions, GA performance was determined by the resulting convergence velocity and solution quality.

The migration period has a direct relationship with the resulting convergence velocity and solution quality for all test functions. A migration period of 5 generations provides the best performance, particularly for the difficult multimodal objective functions.

The combination of high mutation and migration rate early in the search creates a diverse population of individuals, in each subpopulation, allowing for greater exploration of the search space. As the search progresses mutation probability decreases exponentially (2), protecting highly fit individuals. With highly fit individuals migrating every 5 generations, exploitation of the search space occurs, increasing solution quality. Furthermore, the underlying network topology with a 1000Mbps bandwidth and low latency provided low cost migration between demes.

For the highly multimodal objective functions, a single population (no migration) or low migration rate (20 and 40 generations) reduces convergence velocity and results in poor solution quality. Also, as the number of isolated populations increased from 4 to 8, convergence velocity became more rapid and less likely to settle on local optima, providing higher solution quality. The likelihood of premature convergence reduced, since the distributed and independent nature of the search was performed in several regions of the problem space and was thus better able to escape from unfit local minima. Strictly local optimisation algorithms using gradient or Newton methods, for example, most likely get trapped in local optimum, which are frequently distributed in the search space of these highly multimodal functions. The multi-deme, parallel GA presented in this paper was able to scan a larger neighbourhood and to cross intervening valleys towards better optima and hence escape from local optima.

7 Acknowledgements

I wish to thank Dr Matthew Gaston for his assistance with the Linux Computing Cluster Environment.

8 References

[1] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, vol. 1, 16 ed: Addison Wesley Longman, Inc, 1997.

[2] M. Mitchell, *An Introduction to Genetic Algorithms*, vol. 1, 2 ed: Bradford, 1996.

[3] L. H. Lee and Y. Fan, "Developing A Self-Learning Adaptive Genetic Algorithm," presented at Intelligent Control and Automation, 2000. Proceedings of the 3rd World Congress on, Hefei China, 2000.

[4] K. A. De Jong, "An analysis of the behaviour of a class of genetic adaptive systems," University of Michigan, 1975.

[5] J. J. Grefenstette, "Optimization of control parameters for genetic algorithms," *IEEE transactions on Systems, Man and Cybernetics*, vol. SMC-16, pp. 122-128, 1986.

[6] R. Hinterding, Z. Michalewicz, and E. E. Agoston, "Adaptation in Evolutionary Computation: A Survey," presented at Evolutionary Computation, 1997., IEEE International Conference on, Indianapolis, IN USA, 1997.

[7] D. Thierens, "Adaptive Mutation Rate Control Schemes in Genetic Algorithms," presented at Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on, Honolulu, HI USA, 2002.

[8] A. Buczak and H. Wang, "Optimization of Fitness Functions with Non-Ordered Parameters by Genetic Algorithms," presented at Evolutionary Computation, 2001. Proceedings of the 2001 Congress on, Seoul South Korea, 2001.

[9] Q. H. Wu and Y. J. Cao, "Stochastic Optimisation of Control Parameters in Genetic Algorithms," presented at Evolutionary Computation, 1997., IEEE International Conference on, Indianapolis, IN USA, 1997.

[10] C. W. Ho, K. H. Lee, and K. S. Leung, "A Genetic Algorithm Based on Mutation and Crossover with Adaptive Probabilities," presented at Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on, Washington, DC USA, 1999.

[11] T. Back and M. Schutz, "Intelligent mutation rate control in canonical genetic algorithms," presented at Proc. of the International Symposium on Methodologies for Intelligent Systems, 1996.

[12] E. Cantú-Paz, "A Survey of Parallel Genetic Algorithms," Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, 117 Transportation Building, 104 S. Mathews Avenue Urbana, IL 61801, Report 97003, May 1997.

[13] J. Stender, *Parallel Genetic Algorithms: Theory and Practice*. Berlin: IOS Press, 1993.

[14] M. Glickman and K. Sycara, "Reasons for Premature Convergence of Self-Adapting Mutation Rates," presented at Evolutionary Computation, 2000. Proceedings of the 2000 Congress on, La Jolla, CA USA, 2000.

[15] T. Back, *Evolutionary Algorithms in Theory and Practice*. New York: Oxford University Press, 1996.

[16] T. Hiroyasu, "A New Model of Distributed Genetic Algorithm for Cluster Systems: Dual Individual DGA," presented at The 2000 International Workshop on Cluster Computing - Technologies, Environments, and Applications (CC-TEA'2000), Monte Carlo Resort, Las Vegas, Nevada, USA, 2000.