A New Parallel Genetic Algorithm Model

Turki F. Al-Somani[1] and Kalim Qureshi[2]

[1] Computer Engineering Department, King Fahd University of Petroleum & Minerals, Dhahran 31261, Saudi Arabia, tsomani@ccse.kfupm.edu.sa

[2] Information and Computer Science Department, King Fahd University of Petroleum & Minerals, Dhahran 31261, Saudi Arabia, qureshi@ccse.kfupm.edu.sa

Abstract

This paper presents an implementation of three Genetic Algorithm models for solving a reliability optimization problem for a redundancy system with several failure modes, a modification on a parallel a genetic algorithm model and a new parallel genetic algorithm model. These three models are: a sequential model, a modified global parallel genetic algorithm model and a new proposed parallel genetic algorithm model we called the *Trigger Model* (TM). The reduction of the implementation processing time is the basic motivation of genetic algorithms parallelization. In this work, parallel virtual machine (PVM), which is a portable message-passing programming system, designed to link separate host machines to form a virtual machine which is a single, manageable computing resource, is used in a distributed heterogeneous environment. The best result was reached and The TM model was clearly performing better than the other two models.

1. Introduction

Recently, genetic algorithms have received considerable attention regarding their potential as an optimization technique for complex problems [1] and have been successfully applied in the area of industrial engineering. The well-known applications include scheduling and sequencing, reliability design, vehicle routing and scheduling, group technology, facility layout and location, transportation, and many others.

As genetic algorithms usually require more computation time than other heuristic approaches, the basic motivation genetic algorithms parallelization is the reduction of processing time needed to reach an acceptable solution. A good method of parallelization should preserve any properties that sequential algorithm with the same genetic operation would have. It should also not introduce too many additional parameters whose value could significantly affect the genetic algorithm performance. Furthermore, it is highly desirable to eliminate any need for process intercommunication and synchronization. That it is why the asynchronous approach has been favoured in the majority of applications. In this work, parallel virtual machine (PVM) [10] is recognized as an efficient tool for transferring the genetic algorithm into parallel form in a distributed heterogeneous environment.

The main objective of this work is reducing the total processing time of the implementation by parallelization with the same parameters. Section 2 introduces the general structure of genetic algorithms by defining deferent genetic algorithms operations and functions. Section 3 gives a short survey on the previous and recent work done on parallel genetic algorithms. Section 4 describes the new proposed model. Then section 5 discusses the problem definition and the results. The conclusion and future work will be in Section 6.

2. General Structure of Genetic Algorithms

The usual form of a genetic algorithm was described by Goldberg [2]. Genetic algorithms are stochastic search techniques based on the mechanism of natural selection and natural genetics. A genetic algorithm, differing from a conventional search technique, starts with an initial set of random solutions forming what is called a *population*. Each individual in the population is called a *chromosome*, representing a solution to the problem at hand. A chromosome is a string of symbols; it is usually, but not necessarily, a binary bit string. The chromosomes evolve through successive iterations, called *generations*. During each generation, the chromosomes are evaluated, using some measures of *fitness*. To create the next generation, new chromosomes called *offspring*, are formed by either (a) merging two chromosomes from the parent generation using a *crossover* operator or (b) modifying a chromosome using a *mutation* operator. A new generation is formed by (a) selecting, according to fitness values, some of the parents and offspring and (b) rejecting others so as to keep the population size constant. Fitter chromosomes have higher probabilities of being selected. After several generations, the algorithm converges to the best chromosome, which hopefully represents the optimum or suboptimal solution to the problem (see figure 1).

Usually, initialization is assumed to be random. Recombination typically involves crossover

and mutation to yield offspring. In fact, there are only two kinds of operations in genetic algorithms:

1. *Genetic operations*: crossover and mutation
2. *Evolution operation*: selection

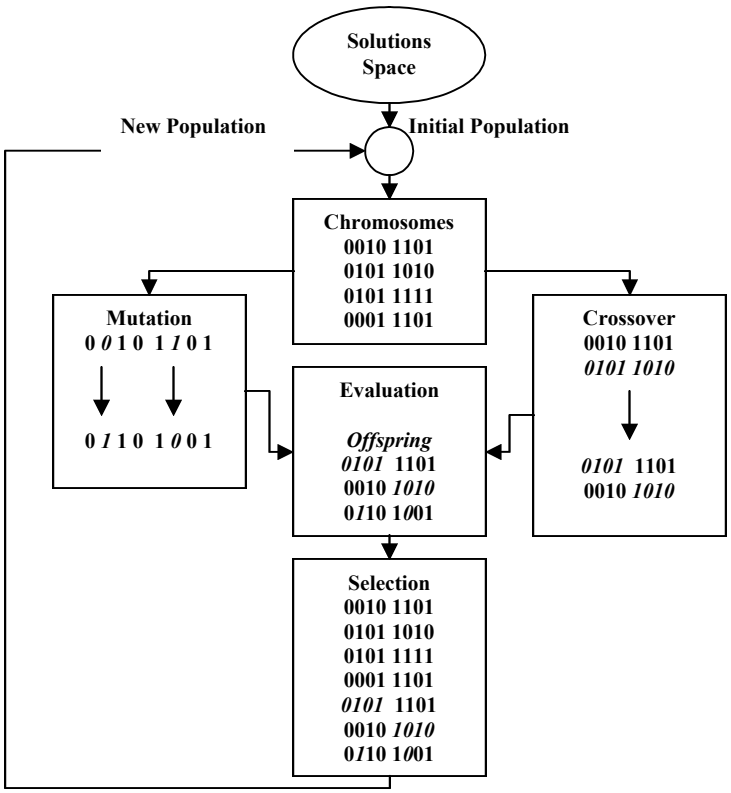

Figure 1: The general structure of a genetic algorithm

The genetic operations mimic the process of heredity of genes to create new offspring at each generation. The evolution operation mimics the process of *biological evolution* to create populations from generation to generation.

3. Parallel Genetic Algorithms

Existing parallel implementation of genetic algorithms can be classified [3 & 4] into three main types:

1. Global single-population master-slave genetic algorithms (GPGA).
2. Massively parallel genetic algorithms (MPGA).
3. Distributed genetic algorithms (DGA).

GPGA is identical to serial genetic algorithms [5] in contrast of other basic models of parallel genetic algorithms. GPGA consists of one population. The master processor stores the entire population and applies genetic operators to produce the next generation. The slave processors are used to evaluate the fitness of a fraction of the population in parallel. Golub and Jakobovic [3] model is a GPGA. In this model and in contrast of traditional master-slave genetic algorithms, the master creates random initial population, evaluates created individuals and starts the slaves. Each slave performs whole evolution process and return back only with the final results as illustrated in Figure 2. These results may probably fall in a local maximum [6] because there is only one initial population created. So, modifying this model to create new initial populations randomly for each slave independently has been done (see Figure 3).

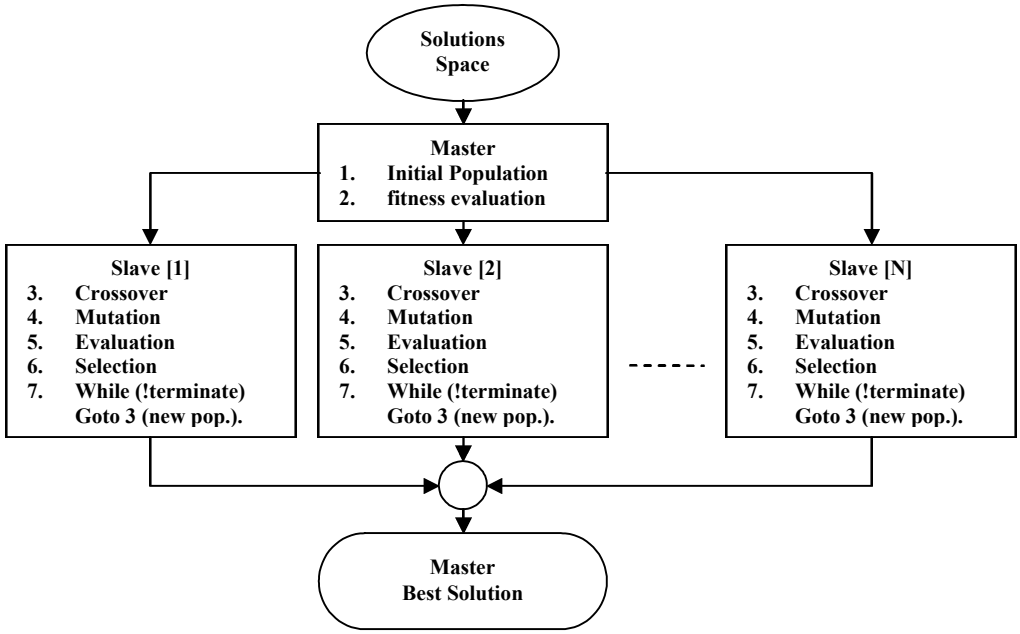

Figure 2: Golub and Jakobovic [3] model

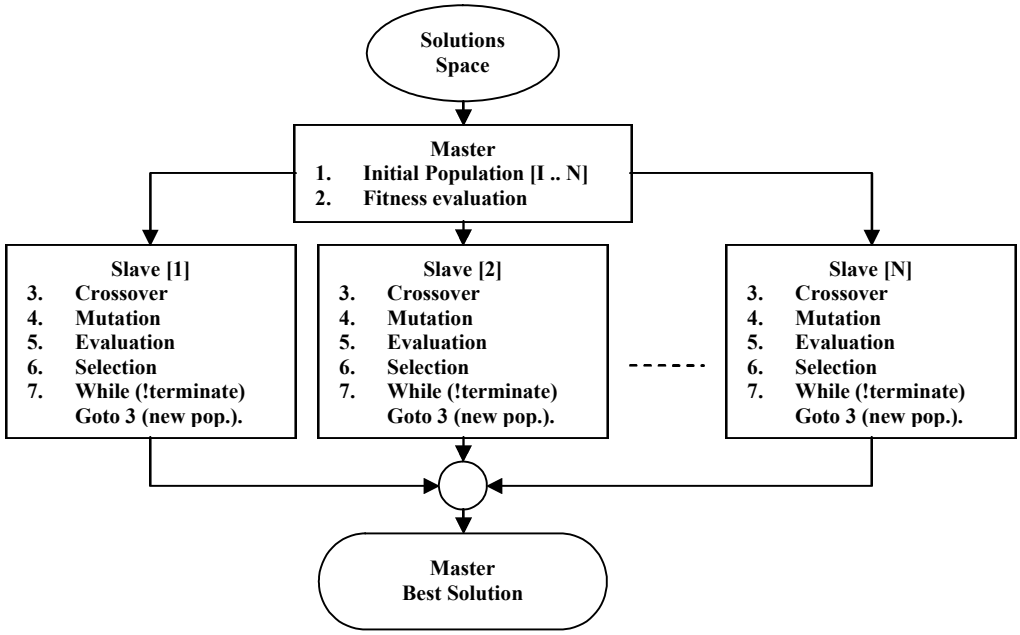

Figure 3: Golub and Jakobovic modified model

4. The TM Model

In our new proposed model, the master only triggers the slaves. Each slave, the master will be also performing as a slave instead of being idle, then creates random initial population, evaluates created individuals performs whole evolution process and then return the final results to the master (see figure 4). This eliminates the time required to generate each populations at the server for slaves, the communication overhead and allows for an exhaustive search of the solution space by the slaves random explorations. We are interested in testing the total processing time of the sequential genetic algorithm, the modified version of the master-slave genetic algorithms model proposed by Golub and Jakobovic [3] and the TM model.

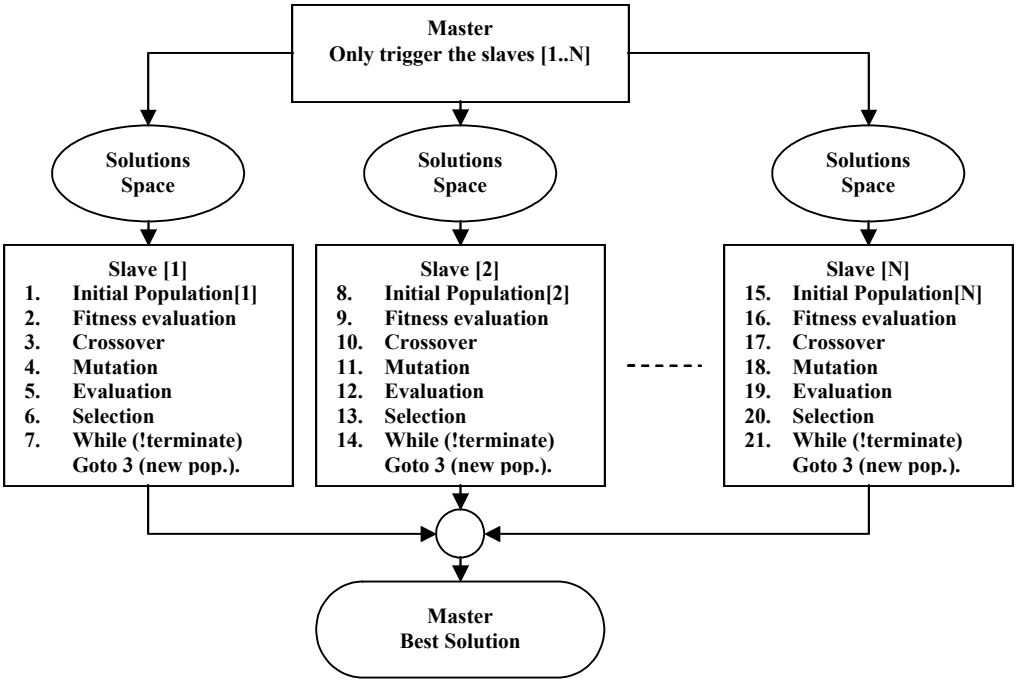

Figure 4: The TM model

5. Results

This work considers the reliability optimization problem of a redundancy system with several failure modes given by Tillman [7], which is used as a benchmark problem by many researchers. This problem was classified in [7] as having *Type 1 Failures with Parallel Redundant Components*. In this situation the redundant components are in parallel, and all of them are subject to the same mode of failure. Complex systems are usually decomposed into functional entities composed of units, subsystems, or components for the purpose of reliability analysis. This subsystem is a switching circuit with three switches in parallel, and all must remain open for it to operate. In this problem we assume that the subsystem is subject to the following mutually exclusive modes of failure:

1. The *O* failures are those where one switch closes when it should not and causes the subsystem to fail.

2. The *A* failures are those where all switches fail by not closing when they should, causing the subsystem to fail.

The problem is to maximize the system reliability subject to three non-linear constraints with parallel redundant units in subsystems that are subject to *A failures*, which occur when the entire subsystem is subjected to the failure condition.

Gen and Cheng simplified the genetic algorithms understanding and described their general structure as can be found in [8]. Detailed numerical examples of solving reliability optimization problems using genetic algorithms could be found in [9].

It was discovered while testing that both the crossover and the mutation function takes a very small amount of the time that could be ignored when compared to the total time and the time needed to generate new populations. So, the comparisons for the total processing time of the implementation have been done for the following:

1. The sequential genetic algorithm model.
2. The modified model of Golub and Jakobovic [3].
3. The TM model.

The best result found is identical with that given [7], [8] and [9]. The processing time of the implementation is shown in Table 1. Figure 5 shows a snapshot of the processing time of the modified Golub and Jakobovic model and the new proposed TM model. The TM model was clearly performing better.

Table 1: The three models proc. time in msec with population size = 20 and 50 generations

No. of slaves	4	8	12	32
Sequential Model	1322	3712	4799	13590
Golub & Jakobovic Modified Model	558	813	1223	2626
Trigger Model	526	624	941	2223

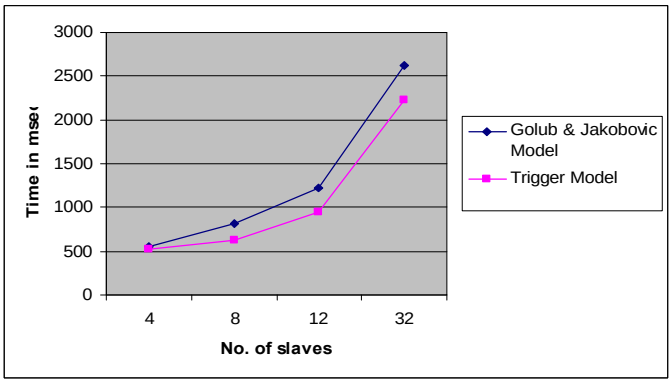

Figure 5: The parallel models processing time with population size = 20 and 50 generations.

6. Conclusion & Future Work

In this paper we presented an implementation of three genetic algorithm models for solving a reliability optimization problem. The two parallel genetic algorithms performed better than the sequential one. A modification to an existing model (Golub and Jakobovic [3]) has been done. And a new model (TM) has been proposed. The testing results show that the TM model was performing better. The testing has been done using PVM [10] in a distributed heterogeneous network. The main objective was reducing the total processing time of the implementation and doing an exhaustive search in the solution space searching for a better solution than the existing one. In future work, more than one task distribution strategy will be tested in both heterogeneous and homogeneous distributed environments. Also, other implementations will be considered testing different implementation natures and behaviours.

References

[1] Haupt, R. L. and Haupt, S. E. (1998) *Practical Genetic Algorithms*, New York: John Wiley & Sons.

[2] Goldberg, D. E. (1989) *Genetic Algorithms in Search, Optimization and Machine Learning*, Longman.

[3] Golub, M. and Jakobovic, D. (2000) *A New Model of Global Parallel Genetic Algorithm*, 22nd International Conference on Information Technology Interfaces IYI, pp 363-368

[4] Cant ´ u-Paz, E. (1998) A Survey of Parallel Genetic Algorithms, *Calculateures Parallels*, Vol. 10, No. 2. Paris: Hermes.

[5] Cant ´ u-Paz, E. and Goldberg, D. E. (1999). On the Scalability of Parallel Genetic Algorithms, *the Massachusetts Institute of Technology, Evolutionary Computation* 7(4): 429-449

[6] Punch III, W. F. and Miagkikh, V. V. (1999) Global search in combinatorial optimization using reinforcement learning algorithms, *Submitted to: 1999 Conference on Evolutionary Computation,* http://garage.cps.msu.edu.

[7] Tillman, F. A. (1969) Optimization by integer programming of constrained reliability problems with several modes of failure. *IEEE Transactions on Reliability,* Vol.R-18 (No. 2): 47-53.

[8] Gen, M. and Cheng, R. (1997) *Genetic Algorithms and Engineering Design*, New York: John Wiley & Sons.

[9] Al-Somani, T. (2000) Reliability Optimization Using Genetics Algorithms, M. Sc. thesis, King Abdul-Aziz University, Saudi Arabia.

[10] Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R. and Sunderam, V. (1994) *PVM: Parallel Virtual Machine { A Users' Guide and Tutorial for Networked Parallel Computing,* The MIT Press.

[11] Cant ´ u-Paz, E. (1998) Designing an Efficient Master-slave Parallel Genetic Algorithms, *Genetic programming Proc. Of the Third Annual Conference,* CA, pp 455-460.

[12] Koza, J. R. (1994) *Genetic Programming II*, Cambridge, Massachusetts: MIT Press.

[13] Qureshi, K., Terasaki, H. and Hatanaka, M. (1996) A Network Parallel Distributed Processing System using PCs and WSs, *Journal of Geotechnology*, Research Association of Japan, Muroran Institute of Tech. , 38 1-8.

[14] Qureshi, K. and Hatanaka, M. (1996) PC-UNIX Parallel Distributed processing on Heterogeneous Hardware, Research *Conference at Muroran Institute of Technology*, Hokkaido Japan, 1-2.