Volume 2, No. 4, April 2011
Journal of Global Research in Computer Science
ISSN-2229-371X
RESEARCH PAPER
Available Online at www.jgrcs.info

EFFECTIVE SOFTWARE TESTING USING GENETIC ALGORITHMS

Kulvinder Singh[1], Rekha Rani*[2], Seema Rani[3], Vedpal singh[4]
CSE Dept. Kurukshetra University,
Kurukshetra,India
kshanda@rediffmail.com[1], rekha_dove@rediffmail.com[2], seemagure7@gmail.com[3], vedpalsiet101@gmail.com[4]

Abstract: In this paper, we give the brief description about the software engineering and their methodologies. This paper also describes how we can use genetic algorithms with software engineering. The advantages of the GA approach are that it is simple to use, requires minimal problem specific information, and is able to effectively adapt in dynamically changing environments.
Keywords: Software Engineering, Genetic Algorithms**.**

INTRODUCTION

Software Engineering: The term *software engineering* first appeared in the 1968 NATO Software Engineering Conference, and was meant to provoke thought regarding the perceived "software crisis" at the time[1]. Software Engineering aim is the production of quality software, software that are delivered on time, within budget, and that satisfies user's requirements. Software Engineering is the application of science and mathematics by which the capabilities of computer equipment are made useful to man via computer programs, procedures and associated documentation. The IEEE Computer Society's *Software Engineering Body of Knowledge* defines "software engineering" as the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering to software[2].

Software Development Life Cycle: Task Scheduling in Multiprocessor [1] [2] is a term that can be stated as finding a schedule for a general task graph to be executed on a multiprocessor system so that the schedule length can be minimized. Multiprocessor scheduling [3] problems can be classified into many different categories based on characteristics of the program and tasks to be scheduled, the multiprocessor system, and the availability of information. Multiprocessor scheduling [2] problems may be divided in two categories: Static and dynamic task scheduling. A static or deterministic task scheduling is one in which precedence constraints and the relationships among the task are known well in advance While non-deterministic or dynamic scheduling [3] is one in which these information is not known in advance or not known till run time. A major factor in the efficient utilization of multiprocessor system is the proper assignment and scheduling of computational tasks among processors. *The problem can have many variations:*

(i) The scheduling algorithm can be deterministic – also known as static – or nondeterministic.

A deterministic task scheduling problem is defined as one in which the knowledge related to tasks, their relations towards each other, timing and the number of processors used are all a prior knowledge. In a nondeterministic problem on the other hand, all or some of these factors can be input-dependent and vary according to run time conditions.

(ii) The tasks can be preemptive or non-preemptive.

A preemptive task scheduling problem allows the tasks to be cut off from execution and another task to begin or continue its execution cycle [operating system example. A non preemptive problem in which task execution must be completely done before another task takes control of the processor.

(iii) The processors can be either homogenous or heterogeneous.

Heterogeneity of processors means that the processors have different speeds or processing capabilities. In a homogenous environment on the other hand, all processors are assumed to have equal capabilities. Efficient scheduling [8] of application tasks is critical to achieving high performance in parallel multiprocessor [9] systems. The objective of scheduling is to map the tasks onto the processors and order their execution. So that task precedence requirements are satisfied and minimum schedule length (or Make span). The most common heuristic methods are List Heuristics, such as Earliest Task First (ETF) algorithm, Critical Path/Most Immediate Successor First (CPMISF) algorithm, and Dynamic Critical Path (DCP) algorithm etc. Another heuristic method is genetic algorithm. A genetic algorithm [7] is a domain-independent global search technique where elements (called individuals) in a given set of solutions (called population) are randomly combined until some termination condition is achieved.

Genetic algorithms and other search techniques: Genetic algorithms [10] [11] as powerful and broadly applicable stochastic search and optimization techniques, are the most widely known types of evolutionary computation [16] [11] methods today. The father of the original Genetic Algorithm was John Holland [13] who invented it in the early 1970's.

Other search techniques are: There are various techniques available for searching and optimization GA is one of them. Genetic Algorithms used for both searching and

optimization. The techniques are shown in figure1. These are:

i) Numerical Techniques: a). Direct Methods b). Indirect Methods
ii) Guided random search techniques: a). Simulated Annealing, b). Evolutionary Algorithms
iii) Enumerative techniques: a). Dynamic Programming

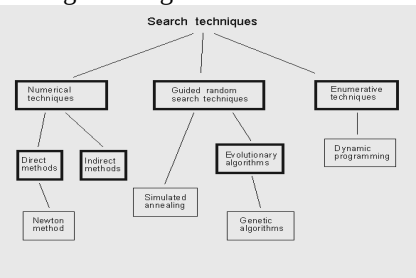

Figure1: Searching Techniques

METHODOLOGY - GENETIC ALGORITHM (GA)

Genetic algorithms (Goldberg, 1989) in particular became popular through the work of John Holland [3] in the early 1970s, and particularly his book Adaptation in Natural and Artificial Systems (1975).

Evolution flow of genetic algorithm

Genetic Algorithms (GAs) are adaptive heuristic search algorithm [21] based on the evolutionary [16] ideas of natural selection and genetics [11]. As such they represent an intelligent exploitation of a random search used to solve optimization [17] problems. Although randomized, GAs are by no means random, instead they exploit historical information to direct the search into the region of better performance within the search space [14]. Figure represents the GA evolution flow.

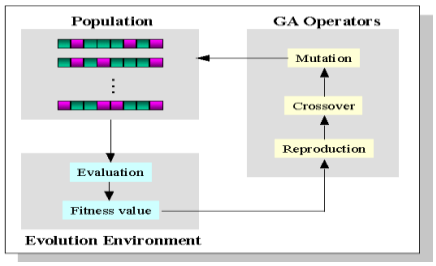

Figure 2: Evolution Flow of Genetic Algorithm

The data structure for genetic algorithms shown in figure

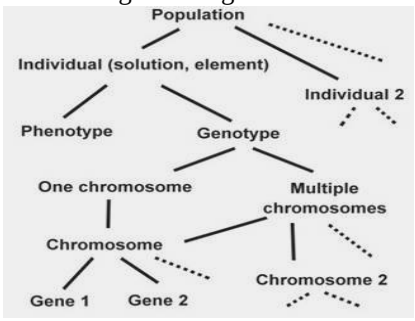

Figure 3 – Genetic Algorithms Data Structure

Basic Terms used in GA: Genes, Chromosome, Parameters, Gene number, Population size, Search Space, Generations.
Basic Principle: The working principle [30] of a simple GA is illustrated in Figure. The major steps involved are the generation of a population [26] of solutions, finding the objective function and fitness function and the application of genetic operators [26]. These aspects are described briefly below. They are described in detail in Genetic operators.

```
                    /*Algorithm GA */
formulate initial population
randomly initialize population
repeat
     evaluate objective function
     find fitness function
     apply genetic operators
          reproduction
          crossover
          mutation
until stopping criteria
```

Figure 4: The Working Principle of a Simple Genetic Algorithm

An important characteristic of genetic algorithm [27] [26] is the coding of variables that describes the problem. The most common coding method is to transform the variables to a binary string or vector; GAs performs best when solution vectors are binary [11]

Features of a Genetic Algorithm

1. Stochastic - different results from different runs
2. Most often used to solve hard problems
3. Maintains a population of solutions
4. Solutions are encoded on chromosomes
5. Reproduction creates new population members
6. Mutation and recombination occur during reproduction
7. Survival of the fittest: better individuals have better chance of reproducing.

What are the strengths of GAs? Following are the strengths of genetic algorithms:

(i) Parallel Algorithm

First and most important point is that genetic algorithms [11] are intrinsically parallel. Most other algorithms are serial and can only explore the solution space to a problem in one direction at a time, and if the solution they discover turns out to be suboptimal, there is nothing to do but abandon all work previously completed and start over. However, since GAs have multiple offspring [17], they can explore the solution space in multiple directions at once. If one path turns out to be a dead end, they can easily eliminate it and continue work on more promising avenues, giving them a greater chance each run of finding the optimal solution [14].

(ii) Many schemas at once

Due to the parallelism that allows them to implicitly evaluate many schemas at once, genetic algorithms [28] are particularly well-suited to solving problems where the space of all potential solutions is truly huge - too vast to search exhaustively in any reasonable amount of time. Most problems that fall into this category are known as "nonlinear". In a linear problem, the fitness [29] of each component is independent, so any improvement to any one part will result in an improvement of the system as a whole. Needless to say, few real-world problems [21] are like this. Nonlinearity is the norm, where changing one component

may have ripple effects on the entire system, and where multiple changes that individually are detrimental may lead to much greater improvements in fitness when combined.

(iii) Global optimum solution

Another notable strength of genetic algorithms [19] is that they perform well in problems for which the fitness landscape [23] is complex - ones where the fitness function is discontinuous, noisy, changes over time, or has many local optima. Most practical problems have a vast solution space, impossible to search exhaustively; the challenge then becomes how to avoid the local optima solutions [23] that are better than all the others that are similar to them, but that are not as good as different ones elsewhere in the solution space. Many search algorithms [25] can become trapped by local optima: if they reach the top of a hill on the fitness landscape, they will discover that no better solutions exist nearby and conclude that they have reached the best one. Evolutionary algorithms [16], on the other hand, have proven to be effective at escaping local optima and discovering the global optimum [23] in even a very rugged and complex fitness landscape. (It should be noted that, in reality, there is usually no way to tell whether a given solution to a problem is the one global optimum or just a very high local optimum. However, even if a GA does not always deliver a provably perfect solution to a problem, it can almost always deliver at least a very good solution.

(iv)Multiple parameters problem

Another area in which genetic algorithms [23] excel is their ability to manipulate many parameters simultaneously. Many real-world problems cannot be stated in terms of a single value to be minimized or maximized, but must be expressed in terms of multiple objectives [17], usually with tradeoffs involved: one can only be improved at the expense of another. GAs are very good at solving such problems: in particular, their use of parallelism enables them to produce multiple equally good solutions to the same problem, possibly with one candidate solution optimizing one parameter [26] and another candidate optimizing a different one and a human overseer can then select one of these candidates to use.

(v) Knowledge does not required

Finally, one of the qualities of genetic algorithms [23] which might at first appear to be a liability turns out to be one of their strengths: namely, GAs know nothing about the problems they are deployed to solve. Instead of using previously known domain-specific information to guide each step and making changes with a specific eye towards improvement, as human designers do, they are "blind watchmakers"; they make random changes to their candidate solutions [11] and then use the fitness function [14] to determine whether those changes produce an improvement.

What are the limitations of GAs? Following are the limitations of genetic algorithms:

Although genetic algorithms have proven to be an efficient and powerful problem-solving strategy, they are not a panacea. GAs [23] does have certain limitations; however, it will be shown that all of these can be overcome and none of them bear on the validity of biological evolution.

(i) Representation for the problem

The first, and most important, consideration in creating a GA is defining a representation [11] for the problem. The language used to specify candidate solutions must be robust; i.e., it must be able to tolerate random changes such that fatal errors or nonsense do not consistently result.

There are two main ways of achieving this. The first, which is used by most genetic algorithms, is to define individuals [23] as lists of numbers - binary-valued, integer-valued, or real-valued - where each number represents some aspect of a candidate solution. If the individuals [23] are binary strings, 0 or 1 could stand for the absence or presence of a given feature. If they are lists of numbers, these numbers could represent many different things: the weights of the links in a neural network, the order of the cities visited in a given tour, the spatial placement of electronic components, the values fed into a controller, the torsion angles of peptide bonds in a protein, and so on.

(ii) Fitness Function representation

The problem of how to write the fitness function [11] must be carefully considered so that higher fitness is attainable and actually does equate to a better solution for the given problem. If the fitness function [23] is chosen poorly or defined imprecisely, the genetic algorithm may be unable to find a solution to the problem, or may end up solving the wrong problem. (This latter situation is sometimes described as the tendency of a GA to "cheat", although in reality all that is happening is that the GA [28] is doing what it was told to do, not what its creators intended it to do.) An example of this can be found in Graham-Rowe 2002, in which researchers used an evolutionary algorithm [17] in conjunction with a reprogrammable hardware array, setting up the fitness function to reward the evolving circuit for outputting an oscillating signal. At the end of the experiment, an oscillating signal was indeed being produced - but instead of the circuit itself acting as an oscillator, as the researchers had intended, they discovered that it had become a radio receiver that was picking up and relaying an oscillating signal from a nearby piece of electronic equipment.

(iii) Problem of choosing the various parameters like the Size of the population, the rate of mutation and crossover, Selection scheme

In addition to making a good choice of fitness function [23], the other parameters of a GA - the size of the population [26], the rate of mutation and crossover [25], the type and strength of selection - must be also chosen with care. If the population size is too small, the genetic algorithm may not explore enough of the solution space to consistently find good solutions.

(iv) Deceptive fitness functions

One type of problem that genetic algorithms [23] have difficulty dealing with are problems with "deceptive" fitness functions, those where the locations of improved points give misleading information about where the global optimum is likely to be found.

(v) Premature convergence

One well-known problem that can occur with a GA [24] is known as premature convergence [23]. If an individual that is more fit than most of its competitors emerges early on in the course of the run, it may reproduce so abundantly that it drives down the population's diversity too soon, leading the algorithm to converge on the local optimum [17] that that individual represents rather than searching the fitness landscape thoroughly enough to find the global optimum.

APPLICATIONS OF GENETIC ALGORITHM

Applications

A heuristic search technique used in computing and Artificial Intelligence [3] to find optimized solutions to search problems using techniques inspired by evolutionary [16] biology: mutation, selection, reproduction [inheritance] and recombination.

(i) Engineering Design

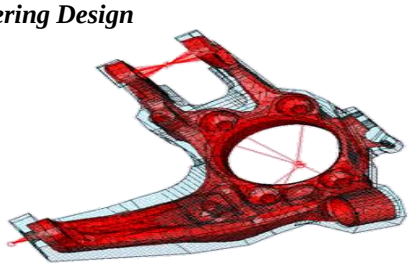

Figure 6: Engineering Drawing

Getting the most out of a range of materials to optimize the structural and operational design of buildings, factories, machines, etc. is a rapidly expanding application of GA [10]. These are being created for such uses as optimizing the design of heat exchangers , robot gripping arms, satellite booms, building trusses, flywheels, turbines, and just about any other computer-assisted engineering design application.

(ii) Robotics

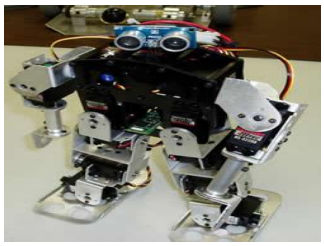

Figure 7: Robotics Designing

Robotics involves human designers and engineers trying out all sorts of things in order to create useful machines that can do work for humans.

(iii) Optimized Telecommunications Routing

Do you find yourself frustrated by slow LAN performance, inconsistent internet access, a FAX machine that only sends faxes sometimes, your land line's number of 'ghost' phone calls every month? Well, GAs [10] are being developed that will allow for dynamic and anticipatory routing of circuits for telecommunications networks..

(iv) Biometric Invention

Figure 8: Biometric Invention

Biomimicry or biomimetics is the development of technologies inspired by designs in nature. Since GAs [11] is inspired by the mechanisms of biological evolution, it makes sense that they could be used in the process of invention as well.

(v) Trip, Traffic and Shipment Routing

New applications of a GA [21] known as the "Traveling Salesman Problem" or TSP can be used to plan the most efficient routes and scheduling for travel planners, traffic routers and even shipping companies.

(vi) Computer Gaming

Those who spend some of their time playing computer Sims games (creating their own civilizations and evolving them) will often find themselves playing against sophisticated artificial intelligence GAs [10] instead of against other human players online.

(vii) Encryption and Code Breaking

Figure 9: Encryption and Code Breaking

On the security front, GAs [23] can be used both to create encryption for sensitive data as well as to break those codes. Encrypting data, protecting copyrights and breaking competitors' codes have been important in the computer world ever since there have been computers, so the competition is intense.

(viii) Gene Expression Profiling

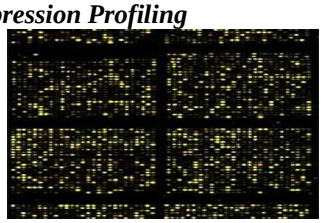

Figure 10: Gene Expression Profiling

The development of micro array technology for taking 'snapshots' of the genes being expressed in a cell or group of cells has been a boon to medical research. GAs [11] has been and is being developed to make analysis of gene [23] expression profiles much quicker and easier.

(ix) Finance and Investment Strategies

In the current unprecedented world economic meltdown one might legitimately wonder if some of those.

(x) Marketing and Merchandising

We could think the word 'merchandising' just the way Mel Brooks said it in the "Space Balls" the movie. Space Balls the toilet paper. Space Balls the lunchbox. Space Balls the flame thrower (the kids love this one). And laugh because it's close enough to reality to be funny.

(xi) Solving Multi-Objective Optimization Problems in Chemical Engineering

Figure 10: A Distillation Tower

Any real-world optimization [10] [13] problem involves several objectives. Chemical engineering [46] is no Exception. Chemical processes, such as distillation (as shown in figure), refinery operations, polymerization, etc.,

involve a number of process parameters which are to be set for achieving certain properties in the final product.

(xii) Genetic Algorithm in Polymer Science and Engineering

Multiple-objective [23] functions have been optimized simultaneously. An example is the minimization of the reaction time in a reactor (lower costs) while simultaneously minimizing [24] the concentration of side products (that affect the properties of the product adversely).

(xiii) Genetic Algorithms in Stock Market Data Mining Optimization

In stock market, a technical trading rule [59] is a popular tool for analysts and users to do their research and decide to buy or sell their shares. The key issue for the success of a trading rule is the selection [27] of values for all parameters and their combinations.

(xiv) Genetic algorithm for cluster analysis

A simple encoding scheme that yields to constant-length chromosomes is used. The objective function [58] maximizes both the homogeneity within each cluster and the heterogeneity among clusters.

CONCLUSION

The problem of scheduling of tasks to be executed on a multiprocessor system is one of the most challenging problems computing. Genetic algorithms are well adapted to multiprocessor scheduling problems. As the resources are increased available to the GA, it is able to find better solutions. GA performs better as compared to other traditional methods. Overall, the GA appears to be the most flexible algorithm for problems using multiple processors. It also indicates that the GA is able to adapt automatically to changes in the problem to be solved.

REFERENCES

[1] G. Syswerda and J. Palmucci, "**The application of genetic algorithms to resource scheduling**", Proceedings of the Fourth International Conference on Genetic Algorithms and Their Applications, pages 502-508, San Mateo, CA, July 1991.

[2] G. A. Cleveland and S. F. Smith, "**Using genetic algorithms to schedule flow shop releases**", Proceedings of the Third International Conference on Genetic Algorithms and Their Applications, pages 160-169, San Mateo, CA, June 1989.

[3] J. H. Holland, "**Adaptation in Natural and Artificial Systems**", The University of Michigan Press, Ann Arbor, MI, 1975.

[4] Cottet, F., Delacroix, J, Kaiser, C., Mammeri, Z., "**Scheduling in Real-time Systems**", John Wiley & Sons Ltd, England, 2002.

[5] Goldberg, David E, "**Genetic Algorithms in Search, Optimization and Machine Learning**", Kluwer Academic Publishers, Boston, 1989.

[6] Mitchell, Melanie, "**An Introduction to Genetic Algorithms**", MIT Press, Cambridge, MA. 1996.

[7] L.M.Schmitt, "**Fundamental Study Theory of Genetic Algorithms**", International Journal of Modelling and Simulation Theoretical Computer Science. 2001.

[8] C. V. Ramamoorthy , "**Optimal scheduling strategies in a multiprocessor system**," IEEE Trans. Computers, vol. C-2I.,Feb. 1972.

[9] I. H. Kasahara and S. Narita, "**Practical multiprocessing scheduling algorithms for efficient parallel processing**," IEEE Transactions on Computers, 1998.

[10] Carnegie-Mellon, "**Genetic Algorithms and Their Applications**", Proc. of the First Int. Conference, July 24-26, 1985.

[11] Dr. Franz Rathlauf, "**Representations for Genetic and Evolutionary Algorithms**", 2nd edition, @ Springer. 2006.

[12] S. Beaty, "**Genetic algorithms and instruction scheduling**", Proceedings of the 24th Microprogramming Workshop (MICRO-24), Albuquerque, NM, November 1991.

[13] John J. Grefenstette, "**Genetic Algorithms and Their Applications**", Proc. 2nd Int. Conf, July 28-31, 1987, MIT, Cambridge,1987.

[14] Davis, "**Handbook of Genetic Algorithms**", Van Nostrand Reinhold, 1991.

[15] E. Hou, R. Hong, and N. Ansari, "**Multiprocessor scheduling based on genetic algorithms**" Dept of ECE, New Jersey Institute of Technology, Technical Report, Aug. 1990.

[16] Michalewicz, "**Genetic Algorithms + Data Structures = Evolution Programs**", Springer, 1996.

[17] Goldberg D., "**Genetic Algorithms in Search, Optimization, and Machine Learning**", Addison-Wessley publishing company Inc., 1989.

[18] Allen, F. & Karjalainen, "**Using Genetic Algorithms to Find Technical Trading Rules**". Journal of Financial Economics, 1999.

