*Oriental Journal of Computer Science & Technology* Vol. **1**(2), 137-141 (2008)

# Coping and limitations of genetic algorithms

**A. VENKATESWARA RAO[1]*, G.A.V.RAMACHANDRA RAO[1] and MANDAVA V. BASAVESWARA RAO[2]**

[1]Noble Institute of Science and Technology, Visakhapatnam (India).
[2]Sadineni Chowdaraiah College of Arts $ Science, Maddirala, Chilakaluripet, (India).

(Received: October 25, 2008; Accepted: Decembe 07, 2008)

**ABSTRACT**

Genetic Algorithms (GA's) are adaptive heuristic search algorithm premised on the evolutionary ideas of natural selection and genetic. The basic concept of GA's is designed to simulate processes in natural system necessary for evolution, specifically those that follow the principles As such they represent an intelligent exploitation of a random search within a defined search space to solve a problem. Genetic Algorithms has been widely studied, experimented and applied in many fields in engineering worlds. Not only does GA's provide alternative methods to solving problem, it consistently outperforms other traditional methods in most of the problems link. Many of the real world problems involved finding optimal parameters, which might prove difficult for traditional methods but ideal for GA's. However, because of its outstanding performance in optimization, GA's has been wrongly regarded as a function optimizer. In fact, there are many ways to view genetic algorithms.

- GA's as problem solvers
- GA's as challenging technical puzzle
- GA's as basis for competent machine learning
- GA's as computational model of innovation and creativity
- GA's as computational model of other innovating systems
- GA's as guiding philosophy

However, due to various constraints, we would only be looking at GA's as problem solvers and competent machine learning here. We would also examine how GA's is applied to completely different fields.

Many scientists have tried to create living programs. These programs do not merely simulate life but try to exhibit the behaviors and characteristics of real organisms in an attempt to exist as a form of life.

**Key words:** Coping, Genetic algorithms.

## INTRODUCTION

GA's were introduced as a computational analogy of adaptive systems. They are modeled loosely on the principles of the evolution via natural selection, employing a population of individuals that undergo selection in the presence of variation-inducing operators such as mutation and recombination (crossover). A fitness function is used to evaluate individuals, and reproductive success varies with fitness. The Algorithms.

- Randomly generate an initial population M (0)
- Compute and save the fitness u (m) for each individual m in the current population M (t)
- Define selection probabilities p (m) for each individual m in M (t) so that p (m) is proportional to u (m)
- Generate M (t+1) by probabilistically selecting individuals from M (t) to produce offspring via genetic operators
- Repeat step 2 until satisfying solution is obtained.

**Who can benefit from GA**

Nearly everyone can gain benefits from Genetic Algorithms, once he can encode solutions of a given problem to chromosomes in GA, and compare the relative performance (fitness) of solutions. An effective GA representation and meaningful fitness evaluation are the keys of the success in GA applications. The appeal of GA's comes from their simplicity and elegance as robust search algorithms as well as from their power to discover good solutions rapidly for difficult high-dimensional problems. GA's is useful and efficient when

- The search space is large, complex or poorly understood.
- Domain knowledge is scarce or expert knowledge is difficult to encode to narrow the search space.
- No mathematical analysis is available.
- Traditional search methods fail.

The advantage of the GA approach is the ease with which it can handle arbitrary kinds of constraints and objectives; all such things can be handled as weighted components of the fitness function, making it easy to adapt the GA scheduler to the particular requirements of a very wide range of possible overall objectives. GA's have been used for problem solving and for modeling. GA's are applied to many scientific, engineering problems, in business and entertainment, including:

**Optimization**

GA's have been used in a wide variety of optimization tasks, including numerical optimization, and combinatorial optimization problems such as traveling salesman problem (TSP), circuit design, job shop scheduling and video & sound quality optimization.

**Automatic Programming**

GA's have been used to evolve computer programs for specific tasks, and to design other computational structures, for example, cellular automata and sorting networks.

**Machine and robot learning**

GA's have been used for many machine-learning applications, including classifications and prediction, and protein structure prediction. GA's have also been used to design neural networks, to evolve rules for learning classifier systems or symbolic production systems, and to design and control robots.

**Economic models**

GA's have been used to model processes of innovation, the development of bidding strategies, and the emergence of economic markets.

**Immune system models**

GA's have been used to model various aspects of the natural immune system, including somatic mutation during an individual's lifetime and the discovery of multi-gene families during evolutionary time.

**Ecological models**

GA's have been used to model ecological phenomena such as biological arms races, host-parasite co-evolutions, symbiosis and resource flow in ecologies.

**Population genetics models**

GA's have been used to study questions in population genetics, such as "under what conditions will a gene for recombination are evolutionarily viable?"

**Interactions between evolution and learning**

GA's have been used to study how individual learning and species evolution affect one another.

**Models of social systems**

GA's have been used to study evolutionary aspects of social systems, such as the evolution of cooperation the evolution of communication, and trail-following behavior in ants.

Applications of Genetic Algorithms:

GA on optimization and planning

Example: - Traveling Salesman Problem:

The TSP is interesting not only from a theoretical point of view, many practical applications can be modeled as a traveling salesman problem or as variants of it, for example, pen movement of a plotter, drilling of printed circuit boards (PCB), real-world routing of school buses, airlines, delivery

trucks and postal carriers. Researchers have tracked TSP's to study bimolecular pathways, to route a computer networks' parallel processing, to advance cryptography, to determine the order of thousands of exposures needed in X-ray crystallography and to determine routes searching for forest fires (which is a multiple-salesman problem partitioned into single TSP's). Therefore, there is a tremendous need for algorithms.

In the last two decades an enormous progress has been made with respect to solving traveling salesman problems to optimality, which, of course, is the ultimate goal of every researcher. One of landmarks in the search for optimal solutions is a 3038-city problem. This progress is only party due to the increasing hardware power of computers. Above all, it was made possible by the development of mathematical theory and of efficient algorithms.

There are strong relations between the constraints of the problem, the representation adopted and the genetic operators that can be used with it. The goal of traveling Salesman Problem is to devise a travel plan (a tour), which minimizes the total distance traveled. TSP is NP-hard (NP stands for non-deterministic polynomial time) - it is generally believed cannot be solved (exactly) in time polynomial. The TSP is constrained:

- The salesman can only be in a city at any time
- Cities have to be visited once and only once.

**Disadvantages**

When GA's applied to very large problems, they fail in two aspects:

- They scale rather poorly (in terms of time complexity) as the number of cities increases.
- The solution quality degrades rapidly

**Failure of standard genetic algorithm**

To use a standard GA, the following problems have to be solved:

- A binary representation for tours is found such that it can be easily translated into a chromosome.
- An appropriate fitness function is designed, taking the constraints into account.

Non-permutation matrices represent unrealistic solutions, that is, the GA can generate some chromosomes that do not represent valid solutions. This happens in the random initialization step of the GA's a result of genetic operators (mutation and crossover).

Thus, permutation matrices are used. Two tours including the same cities in the same order but with different starting points or different directions are represented by different matrices and hence by different chromosomes, for example:

tour (23541) = tour (12354)

A proper fitness function is obtained using penalty-function method to enforce the constraints.

However, the ordinary genetic operators generate too many invalid solutions, leading to poor results. Alternative solutions to TSP require new representations (Position Dependent Representations) and new genetic operators.

**Other applications**

*GA in Business and Their Supportive Role in Decision Making:*

Genetic Algorithms have been used to solve many different types of business problems in functional areas such as finance, marketing, information systems, and production / operations. Within these functional areas, GA's have performed a variety of applications such as tactical asset allocation, job scheduling, machine-part grouping, and computer network design.

**Finance applications**

Models for tactical asset allocation and international equity strategies have been improved with the use of GA's. They report an 82% improvement in cumulative portfolio value over a passive benchmark model and a 48% improvement over a non-GA model designed to improve over the passive benchmark. Genetic algorithms are particularly well suited for financial modeling applications for three reasons:

- They are payoff driven. Payoffs can be improvements in predictive power or returns over a benchmark. There is an excellent match between the tool and the problems

addressed.

- They are inherently quantitative, and well suited to parameter optimization (unlike most symbolic machine learning techniques).
- They are robust, allowing a wide variety of extensions and constraints that cannot be accommodated in traditional methods."

**Information aystems applications**

Distributed computer network topologies are designed by a GA, using three different objective functions to optimize network reliability parameters, namely diameter, average distance, and computer network reliability. The GA has successfully designed networks with 100 order of nodes.

GA has also been used to determine file allocation for a distributed system. The objective is to maximize the programs' abilities to reference the file s located on remote nodes. The problem is solved with the following three different constraint sets:

- There is exactly one copy of each file to be distributed.
- There may be any number of copies of each file subject to a finite memory constraint at each node.
- The number of copies and the amount of memory are both limited.

**Production/Operation applications**

Genetic Algorithm has been used to schedule jobs in a sequence dependent setup environment for a minimal total tardiness. All jobs are scheduled on a single machine; each job has a processing time and a due date. The setup time of each job is dependent upon the job, which immediately precedes it. The GA is able to find good, but not necessarily optimal schedules, fairly quickly.

GA is also used to schedule jobs in non-sequence dependent setup environment. The jobs are scheduled on one machine with the objective of minimizing the total generally weighted penalty for earliness or tardiness from the jobs' due dates. However, this does not guarantee that it will generate optimal solutions for all schedules.

GA is developed for solving the machine-component grouping problem required for cellular manufacturing systems. GA provides a collection of satisfactory solutions for a two objective environment (minimizing cell load variation and minimizing volume of inter cell movement), allowing the decision maker to then select the best alternative.

**Role in decision making**

Applying the well-established decision processing phase model of Simon (1960), Genetic Algorithms appear to be very well suited for supporting the design and choice phases of decision-making. When solving multi-objective problems, GA gives out many satisfactory solutions in terms of the objectives, and then allows the decision maker to select the best alternative. Therefore GA's assist with the design phase of decision processing with multi-objective problems.

GA's can be of great assistance for examining alternatives since they are designed to evaluate existing potential solutions as well to generate new (and better) solutions for evaluation. Thus GA's can improve the quality of decision-making.

**Learning Robot behavior using genetic algorithms**

Robot has become such an prominent tools that it has increasingly taken a more important role in many different industries. As such, it has to operate with great efficiency and accuracy. This may not sound very difficult if the environment in which the robot operates remain unchanged, since the behaviors of the robot could be pre-programmed. However, if the environment is ever changing, it gets extremely difficult, if not impossible, for programmers to figure out every possible behaviors of the robot. Applying robot in a changing environment is not only inevitable in modern technology, but is also becoming more frequent. This has obviously led to the development of a learning robot.

The approach to learning behaviors, which lead the robot to its goal, described here reflects a particular methodology for learning via simulation model. The motivation is that making mistakes on real system can be costly and dangerous. In addition, time constraints may limit the extent of learning in real world. Since learning requires experimenting with behaviors that might occasionally produce undesirable results if applied to real world. Therefore,

as shown in the diagram, the current best behavior can be place in the real, on-line system, while learning continues in the off-line system.

Previous studies have shown that knowledge learned under simulation is robust and might be applicable to the real world if the simulation is more general (add more noise and distortion). If this is not possible, the differences between the real world and the simulation have to be identified.
GA's Role:

Genetic Algorithms are adaptive search techniques that can learn high performance knowledge structures. The genetic algorithms' strength come from the implicitly parallel search of the solution space that it performs via a population of candidate solutions and this population is manipulated in the simulation.

The candidate solutions represent every possible behavior of the robot and based on the overall performance of the candidates, each could be assigned a fitness value. Genetic operators could then be applied to improve the performance of the population of behaviors. One cycle of testing all of the competing behavior is defined as a generation, and is repeated until a good behavior is evolved. The good behavior is then applied to the real world. Also because of the nature of GA, the initial knowledge does not have to be very good.

## CONCLUSION

### Future work

The system described has been used to learn behaviors for controlling simulate autonomous underwater vehicles, missile evasion, and other simulated tasks. Future work will continue examining the process of building robotic system through evolution. We want to know how multiple behaviors that will be required for a higher-level task interact, and how multiple behaviors can be evolved simultaneously. We are also examining additional ways to bias the learning both with initial rule sets, and by modifying the rule sets during evolution through human interaction. Other open problems include how to evolve hierarchies of skills and how to enable the robot to evolve new fitness functions as the need for new skill arises.

## REFERENCES

1. Genetic Algorithms - by Colin R. Reeves, Jonathan E. Rowe

2. New Optimization Techniques in Engineering - by Godfrey C. Onwubolu, B.V.Babu