Genetic Algorithm: A Versatile Optimization Tool

Prof. Swati V. Chande[1] and **Dr. Madhavi Sinha**[2]

Abstract – Genetic Algorithms are a powerful search technique based on the mechanics of natural selection and natural genetics that are used successfully to solve problems in many different disciplines.
The good robustness of these algorithms on problems of high complexity has led to an increasing number of applications in the fields of artificial intelligence, numeric and combinatorial optimization, business, management, medicine, computer science, engineering etc.
In this paper we present a cross section of current genetic algorithm applications from diverse fields and lay a special emphasis on use of genetic algorithms in one of the most important optimization problems in computer science, database query optimization.

Index Terms - Genetic Algorithms, Query Optimization.

1. INTRODUCTION

'Genetic Algorithms' is one of the most useful, general-purpose problem-solving techniques available to developers. It has been used to solve a wide range of problems such as optimization, data mining, games, emergent behavior in biological communities etc.
Like other computational systems inspired by natural systems, Genetic Algorithms have been used in two ways, as techniques for solving technological problems, and as simplified scientific models that can answer questions about nature. [1]
In this paper we focus on the applications of Genetic Algorithms in problem solving. This paper is organized as follows: section 2 contains introductory material providing some general working principles of Genetic Algorithms, section 3 offers a view of the use of Genetic Algorithms to some real world problems, section 4 deals with applications of Genetic Algorithms to database query optimization and finally section 5 provides some concluding remarks and summary of the survey.

2. GENETIC ALGORITHMS

Genetic Algorithms, invented by John Holland, is an ABSTRACTion of biological evolution and is thus a method for moving from one population of chromosomes (strings of bits) to a new population by using a kind of natural selection together with the genetics inspired operators of recombination, mutation and inversion. Each chromosome consists of genes (bits) which are instances of allele (1 or 0) [1].
A Genetic Algorithm functions by generating a large set of possible solutions to a given problem. It then evaluates each of those solutions, and decides on a "fitness level" for each solution set. These solutions then breed new solutions. The parent solutions that were more "fit" are more likely to reproduce, while those that were less "fit" are more unlikely to do so. In essence, solutions are evolved over time [2]. This way there is evolution of the search space scope to a point where a solution can be found.
Figures 1 and 2 show the generic pseudocode and flow diagram of the complete genetic process respectively.
The steps of a general Genetic Algorithm are,

2.1 Representation:

An initial population is created from a random selection of solutions (which are analogous to chromosomes). It involves the representation of an individual (a possible solution or decision or hypothesis) in the form of its genetic structure (a data structure depicting a string of genes called chromosomes). At each point of the search process, a generation of individuals is maintained.
The initial population ideally has diverse individuals. This is necessary because the individuals learn from each other. Lack of diversity in population leads to sub-optimal solutions. The initial diversity may be arranged by uniformly random, grid initialization, non-clustering or local optimization methods [3].

2.2 Evaluation:

A value for fitness is assigned to each solution (chromosome) depending on how close it actually is to solving the problem (thus arriving to the answer of the desired problem). The fitness function is a measure of the objective to be obtained (maximum or minimum values). Fitness function is optimized using the genetic process [4] and evaluates each solution to decide whether it will contribute to the next generation of solutions [5].
Since it selects which individuals can reproduce and create the next generation of population, it is designed with care.

2.3 Selection:

Selection of individuals for the next generation to reproduce or to live on relies heavily on the evaluation function.
Those chromosomes with a higher fitness value are more likely to reproduce offspring (which can mutate after reproduction). The offspring is a product of the father and mother, whose composition consists of a combination of genes from them (this process is known as "crossing over").
After evaluating the fitness of the individuals, by applying the evaluation function, selection of 'fit' individuals for reproduction / recombination is done. The selection techniques that can be used are- deterministic selection, Proportional fitness, Tournament selection etc. each of the techniques has its own pros and cons and can be chosen depending on the problem and population at hand.

[1]*Principal (Computer Science), International School of Informatics and Management, Jaipur*
[2]*Reader, AIM and ACT, Banasthali Vidyapith*

2.4 Recombination:
Recombination or reproduction is as in biological systems, candidate solutions combine to produce offspring in each algorithmic iteration called a generation. From the generation of parents and children, the fittest survive to become candidate solutions in the next generation. Offsprings are produced by specific genetic operators, such as mutation and recombination.
Recombination is randomly picking one or more pairs of individuals as parents and randomly swapping segments (of genes) of the parents.
Solutions thus combine to form offspring for the next generation. Sometimes they pass on their worst information, but if recombination is done in combination with a forceful selection technique, then better solution results are obtained. Recombination may be performed using different methods such as 1-point recombination, n-point recombination and uniform recombination.
Mutation is the most basic way to alter a solution for the next generation. Operators from the local search techniques may be used to slightly twiddle with the solution and introduce new, random information.
It is thus brought about by randomly changing one or more digits (genes) in the string (chromosomes) representing an individual. In binary coding this may simply mean changing a 1 to a 0 and vice versa [6].
Elitism: There is a chance that the best chromosome may be lost when a new population is created by recombination and mutation. The best chromosome(s) may hence be copied to the new population. The rest is done in a classical way. This can rapidly increase the performance of the genetic algorithm, because it prevents the loss of the best-found solutions [7].
If the new generation contains a solution that produces an output that is close enough or equal to the desired answer then the problem has been solved. If this is not the case, then the new generation will go through the same process as their parents did. This will continue until a solution is reached.

```
Create initial population
    Repeat
        Evaluate each individual's fitness
        Select best-ranking individuals to reproduce
        Mate pairs at random among selected individuals
        Apply recombination operator
        Apply mating operator
Until termination condition
```

Figure 1: Pseudocode for a generic Genetic Algorithm

3. APPLICATION OF GENETIC ALGORITHMS TO SOME REAL WORLD PROBLEMS
3.1. Nutritional Counselling:
Nutrition counseling an important part of lifestyle counseling systems.
3.1.1. Gaál et al describe the method used by MenuGene, an intelligent menu planner that generates personalized dietary weekly menu plans with the emphasis on the prevention of Cardiovascular Diseases. The task of weekly dietary menu planning is considered a multi-objective optimization problem which is solved by a multi-level genetic algorithm. The algorithm decomposes the search space to sub-spaces according to the structure and nutritious content of the menu plan. [9]
3.1.2. Alkhalifa A.Y., Niccolai M.J., Nowack W.J. have developed a component of an information system for selection of a nutritionally, culturally, economically and geographically appropriate diet using Genetic Algorithm. [10]
3.2. Stylometry:
Holmes and Richard have applied genetic algorithms to create a set of rules for determining authorship and then let the most useful, or fit rules survive. They combine stylometry, the science of measuring literary or linguistic style, usually to written language, and genetic algorithms to determine authorship. [11]
3.3. Parametric Design of aircraft:
Marle F.Bramlette and Eugene E.Bonehard have discussed optimizing aircraft design when the task is posed as that of optimizing a list of parameters. They used real number representation for Genetic Algorithms and generated a large number of initial population members and worked only with the best ones. [12]
3.4. Robot trajectory generation:
This application demonstrates the application of Genetic Algorithm techniques to the task of planning the path which a robot area in moving from one point to another. Yuval Davidor uses variable-length chromosomes in this solution, & devises some novel & interesting crossover operators. [13]
3.5. Strategy acquisition for simulated airplanes:
A genetic algorithm (SAMUAL) that learns techniques for maneuvering a simulated airplane in order to evade simulated missiles has been described by John J. Grefenshelte. The SAMUEL system tries to discover rules by which a slower but more maneuverable aircraft can evade a faster but less agile missile until the missile runs out of fuel. [14]
3.6. Redistricting:
For a fair and equitable congressional redistricting of Texas, Michael Larson has given a simplified model based on the genetic algorithm technique using the TSP approach. [15]
3.7. Problem solving and in- circuit emulators. (Embedded systems).
An in-circuit emulator is a hardware debugging tool that through its on-board, modified processor, lets one emulate a line processor or family of processors on a target system.
One feature of an ICE is the ability to provide a program clock, essentially the heartbeat of the hardware being designed and tested. The setting up of the clock is done using genetic algorithm. [16].

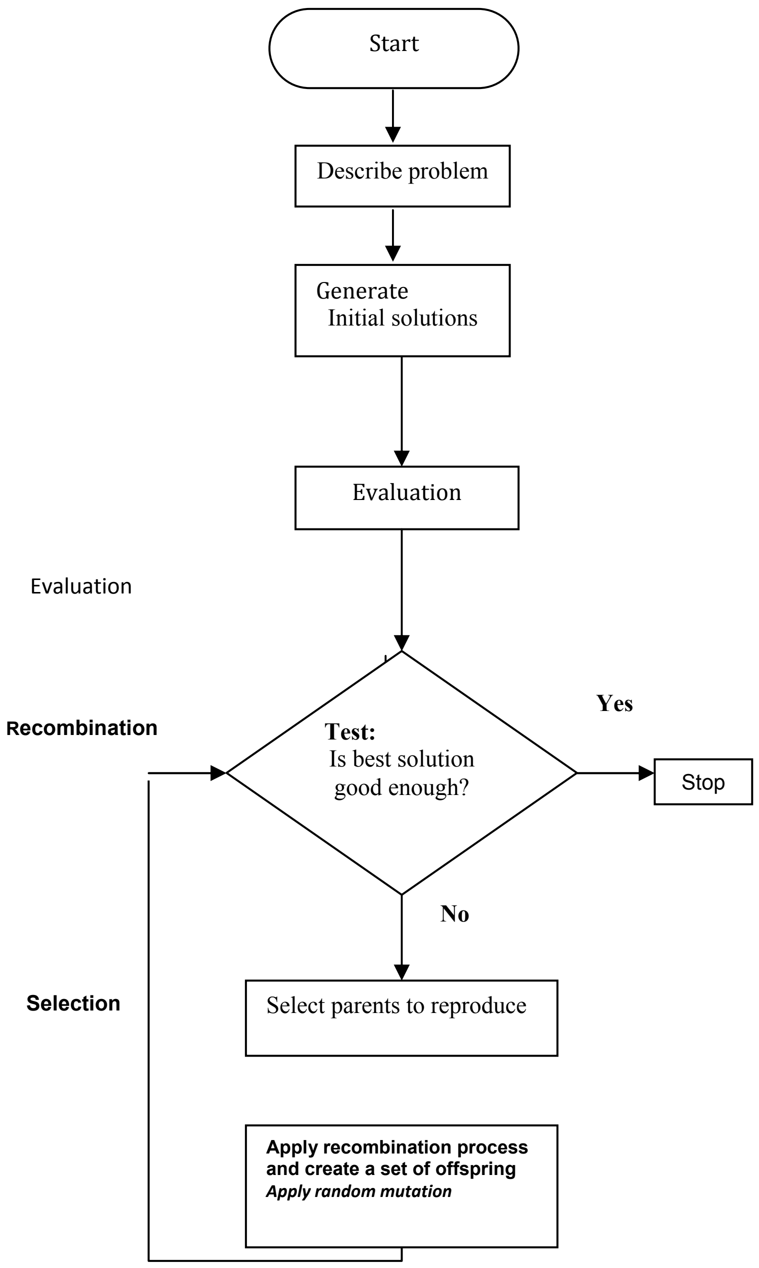
Start
Describe problem
Generate Initial solutions
Evaluation
Evaluation
Recombination
Test: Is best solution good enough?
Yes
Stop
No
Selection
Select parents to reproduce
Apply recombination process and create a set of offspring
Apply random mutation

Figure 2: Flow diagram of the genetic algorithm process Adapted [16]

3.8. Acoustics:

Sato et al used genetic algorithms to design a concert hall with optimal acoustic properties, maximizing the sound quality for the audience, for the conductor, and for the musicians on stage. [17].

3.9. Aerospace engineering:

As telecommunications technology continues to improve, humans are increasingly dependent on Earth-orbiting satellites to perform many vital functions, and one of the problems engineers face is designing their orbital trajectories.

3.9.1. Williams, Crossley and Lang 2001 applied genetic algorithms to the task of spacing satellite orbits to minimize coverage blackouts. [19]

3.9.2 Lockheed Martin has used a genetic algorithm to evolve a series of maneuvers to shift a spacecraft from one orientation to another within 2% of the theoretical minimum time for such maneuvers. The evolved solution was 10% faster than a solution hand-crafted by an expert for the same problem. [19]

3.10. Bandwidth optimization in near video on demand system:

Kriti Priya Gupta has employed Genetic Algorithm technique for minimizing the average bandwidth requirement in near video on demand system. Near video on demand system enable customers requesting the same movie to be grouped together in batches & then the broadcasted to them, using multicasting in a simple transmission stream. [20]

3. 11. Medical

Genetic Algorithms can be used throughout the medical field. Genetic Algorithms can help develop treatment programs, optimize drug formulas, improve diagnostics, and much more.

Plasma X-ray Spectra Analysis: X-ray spectroscopic analysis is a powerful tool for plasma diagnostics. Golovkin et al use genetic algorithms to automatically analyze experimental X-ray line spectra and discuss a particular implementation of the genetic algorithm suitable for the problem. Since spectroscopic analysis may be computationally intensive, they also investigate the use of case injected genetic algorithms for quicker analysis of several similar (time resolved) spectra. [21]

3.12. Scheduling:

Genetic Algorithms can be used for numerous scheduling problems. Using a Genetic Algorithm for difficult scheduling problems enables relatively arbitrary constraints and objectives to be incorporated painlessly into a single optimization method.

3.12.1. Organizers of the Paralympics Games, 1992, used genetic algorithms to schedule events. [22]

3.12.2. School Timetabling:Geraldo Ribeiro Filho and Luiz Antonio Nogueira Lorena have given a constructive approach to the process of fixing a sequence of meetings between teachers and students in a prefixed period of time, satisfying a set of constraints of various types, known as school timetabling problem. Pairs of teachers and classes are used to form conflict-free clusters for each timeslot. Binary strings representing pairs are grouped based on dissimilarity measurement. Teacher preferences and the process of avoiding undesirable waiting times between classes are explicitly considered as additional objectives. [23]

3.13. Musical Composition

An approach of evolving form in musical composition is presented by Ayesh and Hugill. They use of genetic algorithms for this. The approach presented for genetic composition uses samples of musical ideas (one note or more) and not individual MIDI notes. The focus of the approach is on evolving musical form rather than attempting to compose musical sequences. The selection process is guided by the responses of the users within an interactive process. [24]

3.14. Finance:

3.14.1. Mahfoud and Mani 1996 used a genetic algorithm to predict the future performance of 1600 publicly traded stocks. Specifically, the Genetic Algorithm was tasked with forecasting the relative return of each stock, defined as that stock's return minus the average return of all 1600 stocks over the time period in question, 12 weeks (one calendar quarter) into the future. [25].

3.14.2. Naik 1996 reports that LBS Capital Management, an American firm headquartered in Florida, uses genetic algorithms to pick stocks for a pension fund it manages. [22]

3.14.3. Coale 1997 and Begley and Beals 1995 report that First Quadrant, an investment firm in California, uses Genetic Algorithms to make investment decisions for all of their financial services. [26, 27]

3.15. Identifying criminal suspects:

The "FacePrints" software, helps witnesses identify and describe criminal suspects. It uses a genetic algorithm that evolves pictures of faces based on databases of hundreds of individual features that can be combined in a vast number of ways. The program shows randomly generated face images to witnesses, who pick the ones that most resemble the person they saw; the selected faces are then mutated and bred together to generate new combinations of features, and the process repeats until an accurate portrait of the suspect's face emerges.[22]

3.16. Seeking Routes

Texas instrument is drawing on the skills that salmon use to find spawning grounds to produce a Genetic Algorithm that shipping companies can use to let packages "seek" their own best route to their destination.[22]

4. DATABASE QUERY OPTIMIZATION USING GENETIC ALGORITHMS

Optimization of queries can be done through two approaches, one consisting of algebraic manipulations or transformations and the other including strategies to take advantage of the storage of the relations. The algebra based optimization approach is to first represent each relational query as a relational algebra expression and then transform it to an equivalent but more efficient relational algebra expression. The transformation is guided by heuristic optimization rules [28]. The basic idea of the cost-estimation-based approach is - For each query, enumerate all possible execution plans. For each execution plan, estimate the cost of execution plan [28]. Finally choose the execution plan with the lowest estimated cost [29]. Enumerative strategies can lead to the best possible solution, but face a combinatorial explosion for complex queries (e.g., a join query with more than ten relations. Join operation is not only frequently used but also expensive [30].). In order to investigate larger spaces, randomized search strategies have been proposed [31] to improve a start solution until obtaining a local optimum. Examples of such strategies are simulated-annealing [Ioannidis 87] and iterative-improvement [Swami 88]. With the same objective, genetic search strategies [Goldberg 89] can be applied to query optimization, as a generalization of randomized ones [Eiben 90]. Randomized or genetic strategies do not guarantee that the best solution is obtained, but avoid the high cost of optimization. As an optimizer might face different query types (simple vs. complex) with different requirements (ad-hoc vs. repetitive), it should be easy to adapt the search strategy to the problem [32].

The major issue in query optimization is that, the search space is complicated and genetic algorithms are theoretically and empirically proven to provide robust search in complex spaces. These algorithms are computationally simple yet powerful in their search for improvement. They are not fundamentally limited by restrictive assumptions about search space [6]. The use of Genetic Algorithm approach in addressing the Query Optimization issue, therefore seems apt.

Genetic algorithms may be employed in obtaining an optimal solution for each of the two approaches. They may contribute towards the selection of an efficient relational algebra expression and may also find near-optimal execution plans through efficient cost estimation.

All query optimization algorithms primarily deal with joins. Most studies on the use of Genetic Algorithms in Query Optimization also thus focus on joins. Selection of appropriate index for query execution is also one of the major concerns and hence substantial research has also been done in the use of Genetic Algorithms for index selection.

Bennett et al [1991] have studied genetic algorithms for join query optimization. They have given a method for encoding arbitrary binary trees as chromosomes and describe several recombination operators for such chromosomes. Their performance results show that genetic algorithms can effectively identify high quality query execution plans, and the selected plans are in general comparable to or better than the current best--known method for query optimization particularly, the output quality and the time needed to produce such solutions [33].

Farshad Fotouhi and Carlos E. Galarce [1991] of Wayne State University Computer Science Department, have proposed using genetic algorithms to search for near-optimal indexing. They have used a single table and their gene is a binary vector with a position for each of the attributes in the table. A 1 means the column is indexed; 0 means it's not. There is no attempt to accept only genes with the primary key indexed. The idea is to let the genetic process find a solution without any help.

This same chromosome pattern can also be used to represent a type of query. A 1 in a "query chromosome" means that the corresponding column is to be returned; 0 means it's not. This correspondence makes it simple to simulate query runs. The payoff formula is based on hitting or missing an index in a query. The optimal score is to ask for only indexed columns, which makes sense because there's a chance that a non-indexed column would require a sequential search of the table. Fotouhi and Galarce ran a series of random queries with a known statistical distribution against the test database of one million rows. The genes with the highest scores were saved from ("survived") that test run and used to build the next test run ("generation"). The performance of the system was measured in terms of average query-response times. The system leveled out in about ten generations with a 5-bit chromosome, but took longer with a 10-bit chromosome.

The Fotouhi-Galarce experiment gave encouraging results but was based on a single table, a rare occurrence in the real world.

Celko [1993] extended the Fotouhi-Galarce experiment to work on multiple tables. He combined columns to make tables using normal forms built from Functional Dependencies. He used a sample database, identified the functional dependencies for the database and created a query chromosome structure having genes based on attributes. To show each possible 3NF schema, he built tables where functional dependencies for the database were the key. Once the tables were defined, queries were applied against the whole database schema.

A table chromosome was made up of a subset of the original functional dependencies. Two rules were to be obeyed by the tables. First, no combination that violates the 3NF condition was allowed; second, all columns must be present in some table in the schema;

The database schemas were made up of more than one table, and one or more tables were mutated at a time. The payoff function considered joins between tables, the number of tables accessed, etc.

Since the operations on the schema were complex than those used for indexes on a single table by Fotouhi and Galarce, tables had to combine or split. The goal was to have the smallest number of tables used in the queries to avoid the cost of joins. Once the tables were determined for the set of queries, the index genetic algorithm was applied to the tables [34].

Kratica, Ljubi´c and To¡si´c [2003] have proposed a genetic algorithm for solving the ISP (Index Selection Problem) i.e. the problem of minimizing the response time for a given database workload by a proper choice of indexes. Their Genetic Algorithm is based on binary encoding, data structures for the evaluation of the objective function, on the uniform crossover, and simple mutation. They have tested the algorithm on the class of challenging instances known from the literature and demonstrate that the results obtained indicate its efficiency and reliability [35].

Utesch's [1997] model attempts to find the solution of the QO problem similar to a traveling salesman problem (TSP). He has used Postgres Query Optimizer for the research. In this module possible query plans are encoded as integer strings, each string represents the join order from one relation of the query to the next. Parts of the GEQO module are adapted from D. Whitley's Genitor algorithm. Specific characteristics of the GEQO implementation in Postgres are, usage of a *steady state* Genetic Algorithm (replacement of the least fit individuals in a population, not whole-generational replacement) allows fast convergence towards improved query plans, this is essential for query handling with reasonable time; usage of *edge recombination crossover* which is especially suited to keep edge losses low for the solution of the TSP by means of a Genetic Algorithm; mutation as genetic operator is deprecated so that no repair mechanisms are needed to generate legal TSP tours. The GEQO module allows the Postgres query optimizer to support large join queries effectively through non-exhaustive search [36].

Lanzelottel and Patrick Valduriez [1991] have given a solution to the extensibility of the query optimizer search strategy. This solution is based on the object-oriented modeling of the query optimizer, where the search space and the search strategy are independently specified. It is illustrated by applying different search strategies including the genetic algorithm approach [32].

Steinbrunn, Moerkotte and Kemper [1997] have studied different algorithms that compute approximate solutions for optimizing join orders. They extensively scrutinized optimizers from the three classes, heuristic, randomized and genetic algorithms. From their study it turns out that randomized and genetic algorithms are well suited for optimizing join expressions. They generate solutions of high quality within a reasonable running time. The benefits of heuristic optimizers, namely the short running time, are outweighed by merely moderate optimization performance. This study concentrates on the generation of low-cost join nesting orders while ignoring the specifics of join computing.

Steinbrunn et al studied several algorithms for the optimization of join expressions and inferred that randomized and genetic algorithms are much better suited for join optimizations; although they require a longer running time, the results are far better.

For adequate solution space, they found that, with the exception of the star join graph, the bushy tree solution space is preferable in spite of the fact that "pipelining" (avoiding to write intermediate results to secondary memory) can be carried out mainly by left-deep processing trees.

Another consideration is the extensibility of randomized and genetic algorithms: both can be designed to optimize not merely pure join expressions, but complete relational queries. In addition, some of them (namely Iterative Improvement and genetic algorithms) can be easily modified to make use of parallel computer architectures [37].

The authors of this paper, motivated by the applicability of genetic algorithms in a wide range of problems and in optimization in particular, are working on the implementation of genetic algorithms to database query optimization. A Genetic Algorithm involving representation of joins as chromosomes, functions for evaluation of fitness and crossover and mutation operators is considered for minimizing the response time for a given database query.

5. CONCLUSION

Genetic Algorithms are good at taking larger, potentially huge search spaces and navigating them looking for optimal combinations of things and solutions which we might never be able to find. The use of genetic algorithms to solve large and often complex computational problems has given rise to many new applications in a variety of disciplines. They have discovered powerful, high quality solutions to difficult practical problems in a diverse variety of fields.

The few examples surveyed in this paper illustrate the diversity of approaches and point to some of the considerations that have proved important in making applications successful. The use of Genetic Algorithms, for example, in difficult scheduling problems, enables somewhat arbitrary constraints and objectives to be incorporated relatively easily into a single optimization method. With genetic algorithms, the focus lies on evolving forms, rather than on creating new solutions.

The choice of appropriate encoding scheme and fitness function determine the success of a genetic algorithm. Dembski[2002] has said that the 'fitness function guides an evolutionary algorithm into the target.[38]

In recent years, relational database systems have become the standard in a variety of commercial and scientific applications. This has augmented the demand for new, cost-effective optimization techniques for minimizing the response time for query. With genetic algorithms becoming a widely used and accepted method for very difficult optimization problems, their application to database query optimization seems apt. Genetic algorithms thus seem to offer an extremely effective, general purpose, means of dealing with both complexity and scale.

FUTURE SCOPE

Genetic Algorithms are of major significance to the development of the new generation of IT applications. The potential which they offer over existing techniques is enormous. They find application in biogenetics, computer science, engineering, economics, chemistry, manufacturing, mathematics, physics and other fields. And the list will continue to grow especially if Genetic Algorithms are combined with other optimization methods.

Current query optimization techniques are inadequate to support some of the emerging database applications. Genetic algorithms however, are ideally suited to the processing, classification and control of complex queries for very-large and varied data.

REFERENCES

[1] Melanie Mitchell, "An introduction to Genetic Algorithms", Prentice Hall of India, 2004

[2] Hsiung Sam, Matthews James, "An introduction to Genetic Algorithms", 2000 http://www.generation5.org/content/2000/ga.asp

[3] Singh Marjit M, "Genetic Algorithms: Inspired by Nature", Information Technology (it) Magazine, EFY, vol. 13, no.9, July 2004.

[4] Praveen Pathak, Michael Gordon, Weiguo Fan, "Effective Information Retrieval Using Genetic Algorithms Based Matching Functions Adaptation," hicss, vol. 02, no. 2, pp. 2011, February 2000. csdl.computer.org/comp/proceedings/hicss/2000/0493/02/04932011.pdf

[5] Luger G F, "Artificial Intelligence- structure and strategies for complex problem solving", 4th edition, Pearson Education, 2002.

[6] Goldberg David E, "Genetic Algorithms in search, optimization and machine learning", Pearson Education, 2003.

[7] Rajasekaran S, Pai G A Vijaylakshmi, "Neural Networks, Fuzzy Logic and Genetic Algorithms- synthesis and applications", Prentice Hall India, 2002.

[8] Turban E, Aronson J E, "Decision Support Systems, 6th edition, Pearson Education Asia, 2000.

[9] G. Gaál, I. Vassányi, and G. Kozmann, "Automated Planning of Weekly Dietary Menus for Personalized Nutrition Counseling", Proceeding 453, Artificial Intelligence and Applications 2/14/2005-2/16/2005 Innsbruck, Austria

[10] Alkhalifa, A.Y.; Niccolai, M.J.; Nowack, W.J., "Application of the genetic algorithm to nutritional counseling", Proceedings of the 1997 Sixteenth Southern, Biomedical Engineering Conference, 1997

[11] Holmes, David and Forsyth, Richard "The Federalist Revisited: New Directions in Authorship Attribution". Linguistic and Literary Computing, vol. 10, 1995, pp. 111–127

[12] Bramlette, M., Bouchard, E., 1991, "Genetic algorithms in parametric design of aircraft", Handbook of genetic algorithms, L. Davis, ed., Van Nostrand Reinhold, New York, pp. 109-123.

[13] Yuval Davidor "A Genetic Algorithm Applied To Robot Trajectory Generation", in Lawrence Davis, editor, Handbook of Genetic Algorithms, chapter 12, pages 144-165. Van Nostrand Reinhold, New York, New York, 1991.

[14] J. J. Grefenstette "Strategy acquisition with genetic algorithms", in L. Davis, editor, Handbook of Genetic Algorithms, pages 186--201. Van Nostrand Reinhold, 1991.

[15] Michael Larson, "Genetic Algorithms & optimal solutions", Developer 2.0, Dr. Dobb's journal, June 2004.

[16] Philip Joslin, "Genetic Algorithms & real world applications", Developer 2.0, Dr. Dobb's journal, June 2004.

[17] Sato, S., K. Otori, A. Takizawa, H. Sakai, Y. Ando and H. Kawamura. "Applying genetic algorithms to the optimum design of a concert hall." Journal of Sound and Vibration, vol.258, no.3, p. 517-526 (2002).

[18] Williams, Edwin, William Crossley and Thomas Lang. "Average and maximum revisit time trade studies for satellite constellations using a multiobjective genetic algorithm." Journal of the Astronautical Sciences, vol.49, no.3, p.385-400 (July-September 2001).

[19] Gibbs, W. Wayt. "Programming with primordial ooze." Scientific American, October 1996, p.48-50.

[20] Kriti Priya Gupta, "Genetic Algorithm approach for bandwidth optimization ", Synergy – ITS journal of I.T. & management, September 2005.

Continued on page no. 36