International Journal of Advances in Engineering Research http://www.ijaer.com

(IJAER) 2013, Vol. No. 6, Issue No. II, Aug **ISSN: 2231-5152**

OPTIMIZATION OF AN OBJECTIVE FUNCTION BY USING GENETIC ALGORITHM

TRIBIKRAM PRADHAN,DEEPAK KUMAR MOHAPATRA

MTech in Software Technology, School of Information Technology and Engineering (SITE), VIT University, Vellore, Tamil Nadu, India

Mtech in computer science ,Utkal University,Bhubaneswar,odisha,India

ABSTRACT

Non traditional search and optimization methods are becoming more popular in engineering optimization problems in the recent past. These algorithms are generally involved with the potential search and optimization algorithms for complex engineering optimization problems. Genetic algorithm is a new concept and which involves with the principle of natural genetics and natural selection to constitute search and optimization procedures. In this paper I have proposed a new concept of genetic algorithm which is used to minimize a objective function by using the different types of operators of genetic algorithm.Simulated annealing mimics the cooling phenomenon of molten metal's to constitute a search procedure.

Index terms: genetic algorithm, fitness function, GA operators, cross over, mutation

INTRODUCTION

Genetic algorithms are computerized search and optimization algorithms based on the mechanics of natural genetics and natural selection .professor john Holland of the University of Michigan, Ann arbour envisaged the concept of these algorithms in the mid sixties and published his seminar work (Holland 1975). There after a number of students and other researchers have contributed to developing this field. Simulated Annealing was originally inspired by formation of crystal in solids during cooling i.e., the physical cooling phenomenon. As discovered a long time ago by Iron Age blacksmiths, the slower the cooling, the more perfect is the crystal formed. By cooling, complex physical systems naturally converge towards a state of minimal energy. The system moves randomly, but the probability to stay in a particular configuration depends directly on the energy of the system and on its temperature. Where E stands for the energy, k is the Boltzmann constant and T is the temperature. In the mid 70s, Kirkpatrick by analogy of these physical phenomena laid out the first description of simulated annealing. As in the stochastic hill climbing, the iteration of the simulated annealing consists of randomly choosing a new solution in the neighbourhood of the actual solution. If the fitness function of the new solution is better than the fitness function of the current one, the new solution is accepted as the new current solution. If the fitness function is not improved, the new solution is retained with a probability

Where $f(y) - f(x)$ is the difference of the fitness function between the new and the old solution. The simulated annealing behaves like a hill climbing method but with the possibility of going downhill to avoid being trapped at local optima. When the temperature is high, the probability of deteriorate the solution is quite important, and then a lot of large moves arepossible to explore the search space. The more the temperature decreases, the more difficult it is to go downhill; the algorithm tries to climb up from the current solution to reach a maximum. When temperature is lower, there is an exploitation of the current solution. If the temperature is too low, number deterioration is accepted, and the algorithm behaves just like a stochastic hill climbing method. Usually, the simulated annealing starts from a high temperature, which decreases exponentially. The slower the cooling, the better it is for finding good solutions. It even has been demonstrated that with an infinitely slow cooling, the algorithm is almost certain to find the global optimum. The only point is that infinitely slow consists in finding the appropriate temperature decrease rate to obtain a good behaviour of the algorithm. Simulated Annealing by mixing exploration features such as the random search and exploitation features like hill climbing usually gives quite good results. SimulatedAnnealing is a serious competitor to Genetic Algorithms. It is worth trying to compare the results obtained by each. Both are derived from analogy with natural system evolution and both deal with the same kind of optimization problem. Gas differs by two main features, which should make them more efficient. First Gas uses a population-based selection whereas SA only deals with one individual at each iteration. Hence GAs are expected to cover a much larger landscape of the search space at each iteration, but on the other hand SA iterations are much more simple, and so, often much faster. The great advantage of GA is its exceptional ability to be parallelized, whereas SA does not gain much of this. It is mainly due to the population scheme use by GA. Secondly; GAs uses recombination operators, able to mix good characteristics from different solutions. The exploitation made by recombination operators is supposedly considered helpful to find optimal solutions of the problem. On the other hand, simulated annealing are still very simple to implement and they give good results. They have proved their efficiency over a large spectrum of difficult problems, like the optimal layout of printed circuit board, or the famous travelling salesman problem. Genetic annealing is developing in the recent years, which is an attempt to combine genetic algorithms and simulated annealing.

BASIC NOTATION OF GENETIC ALGORITHM

An algorithm is a series of steps for solving a problem. A genetic algorithm is a problem solving method that uses genetics as its model of problem solving. It's a search technique to find approximate solutions to optimization and search problems. Basically, an optimization problem looks really simple. One knows the form of all possible solutions corresponding to a specific question. The set of all the solutions that meet this form constitute the search space. The problem consists in finding out the solution that fits the best, i.e. the one with the most payoffs, from all the possible solutions. If it's possible to quickly enumerate all the solutions, the problem does not raise much difficulty. But, when the search space becomes large, enumeration is soon no longer feasible simply because it would take far too much time. In this it's needed to use a specific technique to find the optimal solution.

Genetic Algorithms provides one of these methods. Practically they all work in a similar way, adapting the simple genetics to algorithmic mechanisms. GA handles a population of possible solutions. Each solution is represented through a chromosome, which is just an abstract representation. Coding all the possible solutions into a chromosome is the first part, but certainly not the most straightforward one of a Genetic Algorithm. A set of reproduction operators has to be determined, too. Reproduction operators are applied directly on the chromosomes, and are used to perform mutations and recombinations over solutions of the problem. Appropriate representation and reproduction operators are really something determinant, as the behaviour of the GA is extremely dependant on it. Frequently, it can be extremely difficult to find a representation, which respects the structure of the search space and reproduction operators, which are coherent and relevant according to the properties of the problems.

Selection is supposed to be able to compare each individual in the population. Selection is done by using a fitness function. Each chromosome has an associated value corresponding to the fitness of the solution it represents. The fitness should correspond to an evaluation of how good the candidate solution is. The optimal solution is the one, which maximizes the fitness function. Genetic Algorithms deal with the problems that maximize the fitness function. But, if the problem consists in minimizing a cost function, the adaptation is quite easy. Either the cost function can be transformed into a fitness function, for example by inverting it; or the selection can be adapted in such way that they consider individuals with low evaluation functions as better. Once the reproduction and the fitness function have been properly defined, a Genetic Algorithm is evolved according to the same basic structure. It starts by generating an initial population of chromosomes. This first population must offer a wide diversity of genetic materials. The gene pool should be as large as possible so that any solution of the search space can be engendered. Generally, the initial population is generated randomly. Then, the genetic algorithm loops over an iteration process to make the population evolve. Each iteration consists of the following steps:

• **SELECTION**: The first step consists in selecting individuals for reproduction. This selection is done randomly with a probability depending on the relative fitness of the individuals so that best ones are often chosen for reproduction than poor ones.

• **REPRODUCTION:** In the second step, offspring are bred by the selected individuals. For generating new chromosomes, the algorithm can use both recombination and mutation.

• **EVALUATION:** Then the fitness of the new chromosomes is evaluated.

• **REPLACEMENT:** During the last step, individuals from the old population are killed and replaced by the new ones. The algorithm is stopped when the population converges toward the optimal solution.

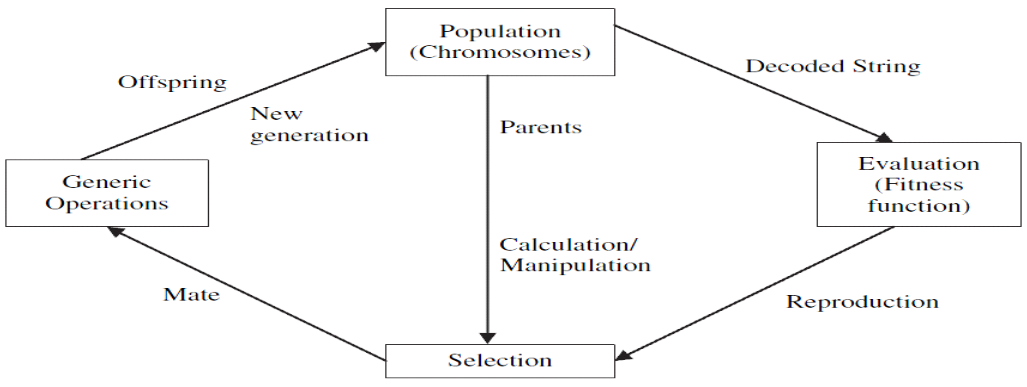
The basic genetic algorithm is as follows:
• [start] Genetic random population of n chromosomes (suitable solutions for the problem)

• [Fitness] Evaluate the fitness f(x) of each chromosome x in the population
• New population] Create a new population by repeating following steps until the New population is complete
- [selection] select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to get selected).
- [crossover] with a crossover probability, cross over the parents to form new offspring (children). If no crossover was performed, offspring is the exact copy of parents.
- [Mutation] with a mutation probability, mutate new offspring at each locus (position in chromosome)
- [Accepting] Place new offspring in the new population.
• [Replace] Use new generated population for a further sum of the algorithm.
• [Test] If the end condition is satisfied, stop, and return the best solution in current.
• [Loop] Go to step2 for fitness evaluation.

Reproduction is the process by which the genetic material in two or more parent is combined to obtain one or more offspring. In fitness evaluation step, the individual's quality is assessed. Mutation is performed to one individual to produce a new version of it where some of the original genetic material has been randomly changed. Selection process helps to decide which individuals are to be used for reproduction and mutation in order to produce new search points population.

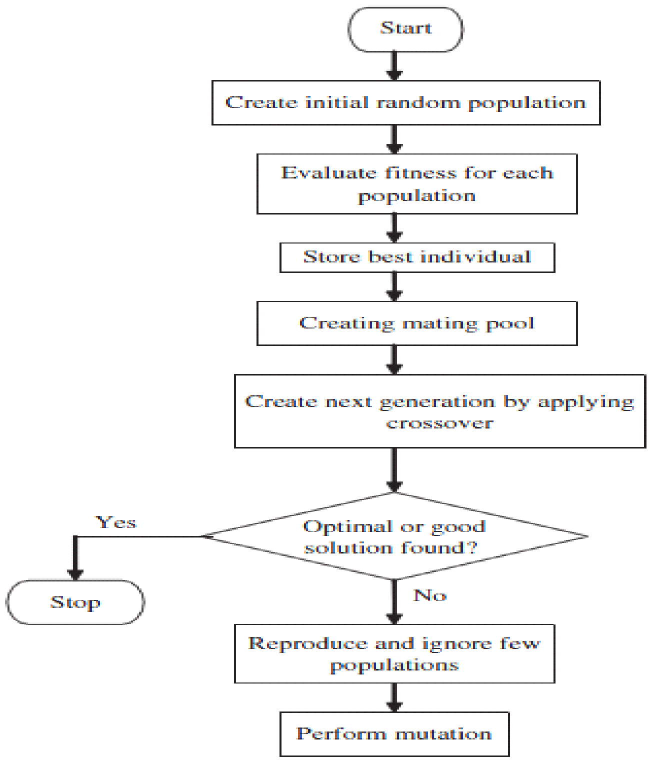
BASIC CONCEPTS OF THE GENTIC ALGORITHMS

GA's have been established as a viable technique for search, optimization, machine learning, and other problems. Theoretical developments by Holland and DeJong have laidthe foundation of GA's. GA's have been theoretically and empirically proven to provide robust search in complex space.Then, the genetic algorithm loops over an iteration process to make the populationEvolve. Each iteration consists of the following steps:



[Genetic algorithm cycle]

FLOW CHART OF GENETIC ALGORITHM:


Start
Create initial random population
Evaluate fitness for each population
Store best individual
Creating mating pool
Create next generation by applying crossover
Optimal or good solution found?
Yes
Stop
No
Reproduce and ignore few populations
Perform mutation

[Flow chart of genetic algorithm]

TERMINOLOGIES AND OPERATORS OF GA

1. GENE

Genes are the basic "instructions" for building a Generic Algorithms. A chromosome is a sequence of genes. Genes may describe a possible solution to a problem, without actually being the solution. A gene is a bit string of arbitrary lengths. The bit string is a binary representation of number of intervals from a lower bound. A gene is the GA's representation of a single factor value for a control factor, where control factor must have an upper bound and lower bound.

2. FITNESS

The fitness of an individual in a genetic algorithm is the value of an objective function for its phenotype. For calculating fitness, the chromosome has to be first decoded and the objective function has to be evaluated. The fitness not only indicates how good the solution is, but also corresponds to how close the chromosome is to the optimal one.

3. **POPULATION**

A population is a collection of individuals. A population consists of a number of individuals being tested, the phenotype parameters defining the individuals and some information about search space. The two important aspects of population used in
Genetic Algorithms are:
1. The initial population generation.
2. The population size.

4. SELECTION

Selection is the process of choosing two parents from the population for crossing. After deciding on an encoding, the next step is to decide how to perform selection i.e., how to choose individuals in the population that will create offspring for the next generation and how many offspring each will create. The purpose of selection is to emphasize fitter individuals in the population in hopes that their off springs have higher fitness. Chromosomes are selected from the initial population to be parents for reproduction. The problem is how to select these chromosomes. According to Darwin's theory of evolution the best ones survive to create new offspring.

4. CROSS OVER

Crossover is the process of taking two parent solutions and producing from them a child. After the selection (reproduction) process, the population is enriched with better individuals. Reproduction makes clones of good strings but does not create new ones. Crossover operator is applied to the mating pool with the hope that it creates a better offspring.
Crossover is a recombination operator that proceeds in three steps:
i. The reproduction operator selects at random a pair of two individual strings for the mating.
ii. A cross site is selected at random along the string length.
iii. Finally, the position values are swapped between the two strings following the
Cross site.

5, MUTATION

After crossover, the strings are subjected to mutation. Mutation prevents the algorithm to be trapped in a local minimum. Mutation plays the role of recovering the lost genetic materials as well as for randomly disturbing genetic information. It is an insurance policy against the irreversible loss of genetic material. Mutation has traditionally considered as a simple search operator. If crossover is supposed to exploit the current solution to find better ones, mutation is supposed to help for the exploration of the whole search space. Mutation is viewed as a background operator to maintain genetic diversity in the population. It introduces new genetic structures in the population by randomly modifying some of its building blocks. Mutation helps escape from local minima's trap and maintains diversity in the population. It also keeps the gene pool well stocked, and thus ensuring periodicity. A search space is said to be periodic if there is a non-zero probability of generating any solution from any population state.

HOW GENETIC ALGORITHM DIFFERS FROM OTHER CONVENTIONAL OPTIMIZATION TECHNIQUES

- GAs operate with coded versions of the problem parameters rather than parameters themselves i.e., GA works with the coding of solution set and not with the solution itself.

- Almost all conventional optimization techniques search from a single point but GAs always operate on a whole population of points(strings) i.e., GA uses population of solutions rather than a single solution from searching. This plays a major role to the robustness of genetic algorithms. It improves the chance of reaching the global optimum and also helps in avoiding local stationary point.

- GA uses fitness function for evaluation rather than derivatives. As a result, they can be applied to any kind of continuous or discrete optimization problem. The key point to be performed here is to identify and specify a meaningful decoding function.

THE ADVANTAGES OF GENETIC ALGORITHM

- Solution space is wider
- Easy to discover global optimum
- The problem has multi objective function
- Only uses function evaluations.
- Easily modified for different problems.
- Handles large, poorly understood search spaces easily
- Good for multi-modal problems Returns a suite of solutions.
- They require no knowledge or gradient information about the response surface
- Can be employed for a wide variety of optimization problems

APPLICATIONS OF GENETIC ALGORITHM

• TSP and sequence scheduling
• Control–gas pipeline, pole balancing, missile evasion, pursuit
• Design–semiconductor layout, aircraft design, keyboard configuration, communication Networks
• Scheduling–manufacturing, facility scheduling, resource allocation
• Machine Learning–Designing neural networks, both architecture and weights, Improving classification algorithms, classifier system.

EXPERIMENTATION

The objective is to minimize the function

$F(x_1, x_2) =(x_1^2+x_2-11)^2+(x_1+x_2^2-7)^2$

In the interval $0<x_1, x^2<=6$.the true solution to this problem is $(3,2)^T$ having a function value equal to zero.

STEP 1:

In order to solve this problem using genetic algorithms we choose binary coding to represent variables x1 and x2. In the calculations here ,10 bits are chosen for each variables thereby making a total string length equal to 20.with 10 bits ,we can get a solution accuracy of (6-0)/(210-1) or 0.006 in the interval (0,6).We choose roulette wheel selection a single point crossover operator and a bit wise mutation operator .the crossover and mutation probabilities are assigned to be 0.8 and 0.05 respectively.we decide to have 20 points in the population.The random population created using Knuth's random number generator3 with a random seed equal to 0.760 .we set tmax=30 and initialize the generation counter t=0.

STEP 2:

The next step is to evaluate each string in the population .we calculate the fitness function of the first string. the first substring (1100100000) decodes to value equal to $(2^9+2^8+2^5)$ or 800.Thus the corresponding parameter value is equal to 0+(6-0)*800/1023 or 4.962. the second substring (1110010000) decodes a value equal to $(2^9+2^8+2^7+2^4)$ or 912.thus the corresponding parameter value is equal to 0+(6-0)912/1023 or 5.349 .thus the first string corresponds to the point $x(1)=(4.692 ,5.349)^T$. These values can now be substituted in the objective function expression to obtain the function value. It is found that the function value at this point is equal to f(x(1))=959.680.We now calculate the fitness function value at this point using the transformation rule:F(x(1))=1.0/(1.0+959.680)=0.001.This value is used in the reproduction operation.Similarly other strings in the population are evaluated and fitness values are calculated. Table shows the objective function value and the fitness value for all 20 strings in the initial population.

STEP 3:

Since $t=0<t_{max}=30$, we proceed to step 4.

STEP 4:

At this step we select good strings in the population to form the matting pool. In order to use the roulette wheel selection procedure, we first calculate the average fitness of the population. By adding the fitness function values of all the strings and dividing the sum by the population size, we obtain F=0.008.the next step is to compute the expected count of each string as F(x)/F.The values are calculated and shown in column A of table. In other words we can compute the probability of each string being copied in to the matting pool by dividing these numbers with the population size.Once these probabilities are calculated, the cumulative probability can also be computed. These distributions are also shown in column c of table. In order to form the matting pool we create random numbers between zero and one. And identify the particular string which is specified by each of these random numbers.

For example, if random number 0.472 is created the tenth string gets a copy in the matting pool, because that string occupies the interval (0.401 ,0.549), as shown in column c.Column e refers to the selected string. similarly other strings are selected according to the random numbers shown in column D. after this selection procedure is repeated n times (n is the population size) , the number of selected copies for each string is counted. This number is shown in column F. The complete matting pool is also shown in the table. Column a and F reveal that the theoretical expected count and the true count of each string more or less agree with each other.fig shows the initial random population and the matting pool after the reproduction. This points marked with an Enclosed box are the points in the matting pool. The action of the reproduction operator is clear from this plot. The inferior points have been probabilistically eliminated from further consideration. Notice that not all selected points are better than all rejected points. For example, the 14th individual (with afitness value 0.002) is selected but the 16th individual (with a fitness value 0.005) is not selected.

Although the above roulette wheel selection is easier to Implement. It is noisy. Amore stable version of this selection operator is sometimes used .after the expected count for each individual string is calculated.The strings are first assigned copies exactly equal to the mantissa of the expected count. Thereafter, the regular roulette wheel selection is implemented using the decimal part of the expected count as the probability of selection.This selection method is less noisy and is known as stochastic remainder selection.

STEP 5:

At this step, the strings in the matting pool are used I the crossover operation. In as ingle point crossover two strings are selected at random and crossed at a random site. Since the matting pool contains strings at random, we pick pairs of string from the top of the list. Thus string 3 and 10 participate in the crossover operation. When two strings are chosen for crossover,first a coin is flipped with a probability p_c =0.8 to check whether a crossover is desired or not.

If the outcome of the coin flipping is true, the crossingover is performed. Otherwise the strings are directly placed in an intermediate population for subsequent genetic operation. It turns out that the outcome of the first coin flipping is true, meaning that a crossover is required to perform. The next step is to find a cross site at random. We choose a site by creating a random number between (0,l-1) or (0, 19).it turns out that the obtained random number is 11.thus we cross the strings at the site 11 are placed in the intermediate population. Then strings 14 and 2 are used in the crossover operation. This time the coin flipping comes true again and we perform the crossover at the site 8 found at random. The new children strings are put into the intermediate population. With the flipping of a coin with a probability pc=0.8, it turns out that fourth, seventh, and tenth crossovers come out to be false. Thus in these cases, the strings are copied into the intermediate population. The complete population at the end of the crossover operation .it is interesting to note that with pc=0.8 the expected number of crossover in a population of size 20 is 0.8*20/2 or 8.after crossover operation some good points and some not good points are created after crossover. In some cases points far away from the parent's points are created and in some case points close to the parent points are created.

STEP 6:

The next step is to perform mutation on strings in the intermediate population, for bit wise mutation operator we flip a coin with a probabilitypm=0.05 for every bit. If the outcome is true, we alter the bit to 1 or 0 depending on the bit value. With a probability of 0.05 a population size 20 and a string length 20, we can expect to alter a total of about 0.05*20*20 or 20 bits in the table. As counted from the table, we have actually altered 16 bits. In some cases the mutation operator changes a point locally and in some case it can bring a large change. The points marked with a small circle are points in the intermediate population.The points are moved along a particular variable only. Like the crossover operator the mutation operator has created some points better and some points worse than the original points. This flexibility enables GA operators to explorer the search space properly before converging to a region prematurely. Although this requires some extra computation, this flexibility is essential to slove global optimization problems.

STEP 7:

The resulting population becomes the new population .we now evaluate each string as before by first identifying the substring for each variable and mapping the decoded values of the substrings in the chosen intervals.This completes one iteration of the genetic algorithms. We increment the generation counter to t=1 and proceed to step 3 for the next iteration. The new populationafter one iteration is shown in fig.The average fitness of new population is calculated to be o.015, a remarkable improvement from that in the initial population.(recall that the average in the initial population was 0.008).the best point in this population is found to have a fitness function equal to 0.050,which is also better than that of the initial population(0.024).

This process continues until the maximum allowable generation is reached or some other termination criteria are met.The population after 25 generation is found to be the points (3.003, 1.994) T with a function value 0.001.the fitness value at this point is equal to 0.999 and the average population fitness of the population is 0.474.we also observe that the total number of function evaluations required to obtain this solution is 0.8*20*26 or 416(including the evaluation of the initial population).

CONCLUSION

Generally genetic algorithm is used for optimization, search and inductive problems. In this paper I have taken an example of objective function and by using the concept of genetic algorithm I have minimize the function .i have implemented 30 iterations of the algorithm and found the best point for the function. Genetic algorithm is more suitable for maximization problems but here I have taken the minimization problems and I have transformed it into the maximization problems by some suitable transformation. In general a fitness function F(x) is first derived from the objective function and used in successive genetic operations. For maximization problems the fitness function can be considered to be the same as the objective function or F(X)=f(x).For minimization problems the fitness function is an equivalent maximization problems chosen such that the optimum point remains unchanged .the following fitness function is often used :F(X)=1/1+f(x).this transformation does not alter the location of the minimum but converts a minimization problem to an equivalent maximization problem. The fitness function value of a string is known as the strings fitness.

REFERENCES

- Fonseca M. C. and Fleming J. P. (1998) Multi−objective Optimization and Multiple Constraint Handling with Evolutionary Algorithms-Part I: A Unified Formulation. IEEE Transactions on Systems, Man and Cybernetics - Part A: Systems and Humans, 28(1):26-37.

2. Ghosh A. and Nath B.,(2004). Multi−objective rule mining using genetic algorithms.Information Sciences 163 pp 123-133.

3. Freitas A. A., (2003) A Survey of Evolutionary Algorithms for Data Mining and KnowledgeDiscovery. Advances in evolutionary computing: theory and applications, Pp 819 -845.

4.Khabzaoui M., Dhaenes C., and Talbi E. (2005). Parallel Genetic Algorithms forMulti−Objective rule mining. MIC2005. The 6th Metahueristics International Conference,Vienna, Austria.

5. Zitzler E., Deb K., Thiele L. (1998). An evolutionary Algorithm for Multi−objective Optimization: The Strength Pareto Approach.

6.Zitzler E., Deb K., Thiele L. (2000). Comparison of Multi−objective Evolutionary Algorithms:
Empirical Results.Evolutionary Computation Vol. 8 Number 2 pp 173 -195.

7.Zitzler E., Luamanns M., Thiele L. (2001). SPEA2: Improving the Strength Pareto Evolutionary

Algorithm.Computer Engineering and Networks Laboratory (TIK), TIK-Report 103.

8.Noda, E., Freitas, A.A., Lopes, H.S.: Discovering interesting prediction rules with a genetic algorithm. In: Proc. Congress on Evolutionary Computation (CEC-1999), July 1999. IEEE, Washington (1999).

9. Pei, M., Goodman, E.D., Punch, W.F.: Pattern discovery from data using genetic algorithms. In: Proc. 1st Pacific-Asia Conf. Knowledge Discovery and Data Mining (PAKDD-1997), Febuary 1997. World Scientific, Singapore (1997).

10. Jung, Y., Jog, S.P., van Gucht, D.: Parallel genetic algorithms applied to the traveling salesman problem. SIAM Journal of Optimization 1(4), 515–529 (1991).

11. Quinlan, J.R.: Generating production rules from decision trees. In: Proc. of IJCAI- 1987, pp. 304–307 (1987).

12. Rullo, P., Cumbo, C., Policicchio, V.L.: Learning rules with negation for text categorization. In: Proc. of SAC - Symposium on Applied Computing, Seoul, Korea, March 11-15 2007, pp. 409–416. ACM, New York (2007).

13. Sebastiani, F.: Machine learning in automated text categorization. ACM Computing Surveys 34(1), 1–47 (2002).

14. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005).

15. Yang, Y., Pedersen, J.O.: A comparative study on feature selection in text categorization. In: Fisher, D.H. (ed.) Proceedings of ICML-97, 14th International Conference on Machine Learning, Nashville, US, pp. 412–420. Morgan Kaufmann Publishers, San Francisco (1997).

AUTHOR'S PROFILE

Mr. TribikramPradhan: I was born in barghat, Orissa, india. I Received My B.Tech in Computer Science And Engineering From Bhadrak Institute Of Engg. And Tech, Under BijuPattnaik University and technology, Rourkela, Orissa, India in 2010. Currently I Am Doing My M.Tech In Software Technology At Vit University, Vellore,Tamil Nadu, India. My Major Research Interests Are In Data Mining, Artificial Intelligence, Automata Theory, Compiler Design, Operating System, Computer Graphics, and VLSI Low Power Technique. i was awarded the best student award in 2005. I have published two papers in the area of image processing and vlsi low power technique. I Have Completed Successfully Research Project Entitled Vertical Fragmentation In Distributed Database System And Enhancement Of Turing Machine To Universal Turing Machine To Halt For Recursive Enumerable Language And Its JFlap Simulation. Currently I Am Doing My Final Project on Application of genetic algorithm and roughest theory for the knowledge extraction and rule induction.