A new position based Fitness Evaluation for Genetic Algorithm

P.Vijayalakshmi [1], M.Ayyappan [1], K. Annaram[2]
[1]Department of Information Technology
[2]Department of Electronics and Communication Engineering
Kamaraj College of Engineering and Technology
Virudhunagar, TN, India
vijikcet@gmail.com

ABSTRACT: *In this paper, a new approach for fitness evaluation of Genetic Algorithm (GA) is discussed. The approach is based on position of the genes in a chromosome. A random initial population sometimes may generate recessive strings that may increase the number of iterations. Also, it is hard to decide time to arrive at a particular solution. Chromosome's strength is dependent on the arrangement of its genes. A new position based fitness evaluation technique is introduced, where fitness is assigned to each gene's position to compute the strength of a chromosome. Also selecting the weakest position for crossing over avoids the probability of dummy iterations and increases the efficiency. We have tested the new approach on "N" Queen Problem that represents NP hard problems. The experimental results are comparatively good with other state of art GAs, simple genetic algorithm (SGA), enhanced improved genetic algorithm (EIGA) and adaptive genetic algorithm (AGA). The strength of new fitness approach increases in fitness in each iteration and efficient. Potential application includes search techniques and machine learning.*

Keywords: Population, Breeding Cycle, Fitness Evaluation, Cross Over and Mutation

Received: 21 April 2011, Received 26 May 2011, Revised 4 June 2011

1. Introduction

Genetic Algorithms have proven to be a powerful tool in various areas of computer science, including machine learning, search, and optimization. The principles of G As were developed more than thirty years ago. It is a search technique used in computing to find exact or approximate solutions to optimization and search problems. The two important aspects of population used in Genetic Algorithms are, initial population generation and the population size which depends on the complexity of the problem. The breeding process is the heart of the genetic algorithm. The breeding cycle consists of three steps: selecting parents, crossing the parents to create new individuals, and replacing old individuals in the population with the new ones. In simple genetic algorithm selecting individuals for reproduction is the first step. 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. In reproduction, offspring are bred by the selected individuals. For generating new chromosomes, the algorithm use both recombination and mutation. In evaluation, the fitness of the new chromosomes is evaluated. In the replacement, individuals from the old population are killed and replaced by the new ones. The draw backs are more time, more memory and more iterations. The main idea of parallel genetic algorithm (PGA) is to distribute expensive tasks across slaves controlled by a master process to be executed in parallel. In a classic configuration, the master maintains a population and executes genetic operators (selection, crossover and mutation), while slaves perform fitness evaluation. Master assigns a part of population to each slave and waits for them to finish. PGA can achieve significant increase in speed, especially for expensive evaluation functions on large populations. However, due to communication between the master and slaves, there is an upper limit for the number of slave

processes. So, speed gains are limited by master-slave communication overhead.

A Hybrid Genetic Algorithm has been designed by combining a variant of an already existing crossover operator with a number of heuristics. One heuristics is used for generating initial population and other two are applied to the offspring either obtained by crossover or by shuffling. Two heuristics were applied to offspring to prevent impasse at local optimum. The hybrid genetic algorithm was designed to use heuristics for initialization of population and improvement of offspring produced by crossover. Adaptive genetic algorithms use many parameters, such as the population size, crossing over probability and the mutation probability. These parameters are varied while the algorithm is running. For example, the mutation rate is changed according to changes in the population; the longer the population does not improve, the higher the mutation rate is chosen and vice versa, it is decreased again as soon as an improvement of the population occurs. The draw backs of AGA includes one where the crossing point is selected randomly and the population retains with out any improvement for longer time [13], [14],[15].

In [1], a statistics-based adaptive non-uniform crossover (SANUX) was proposed. SANUX used the statistics information of the alleles to calculate the swapping probability for crossover operation. The crossover is an operation in GA that can not generate special off springs from their parents because it uses only acquired information. In [2], PGA was used to solve N-Queen problem with custom genetic operators to increase GA speed. Speed gains were limited by master-slave communication overhead. In [3], a state of the art PGA was reviewed and a new taxonomy of PGA was proposed. The idea was to have one individual for every processing element. It demanded more computational load and memory and the disadvantage of involving high communication cost. In [4], a modification of GA, enhanced improvement of individuals was discussed. The approach was to maintain good -individuals by local search techniques, tabu search, simulated annealing, and iterated local search. These heuristics were time-consuming. After producing and improving the offspring, again poor off springs were improved by additional iterations. In [5], a selective mutation method for improving the performance of GA was proposed, individuals were ranked and then mutated one bit based on their ranks. This selective mutation helped GA to quickly escape local optima. The mutation is an aiding process that can not change many bits in the individuals and hence its performance.

In [6], GA for information retrieval (IR) was discussed. The aim was to help an IR system to find a good respond to a query expressed by the user in a huge documents collection. GA based IR method is not faster, difficult to implement and understand. In [7], an approach to identify factors that affect the efficiency of GA was reviewed; optimal values were obtained using the developed approach. In this paper, we have concentrated on the construction of initial population space. Two new techniques have been introduced: to generate an initial population space and a new fitness evaluation function.

In [8], a detection system employing GAs and neural networks were presented. In first method, neural networks were trained to recognize the characters and in second method, template matching was used. To control size of the neural network inputs and template, a GA search was applied. The drawback was that a huge color database was to be created manually extracting colors from license plates. In [9], a modification of GA for shape detection using edge detection algorithm was presented. The use of GA had reduced the time needed for detection task. The time and storage complexity of the method were linear function of number of the detected features. In [10], a tree based Genetic Programming (GP) for classification methods was reviewed and analyzed. Strengths and weaknesses of various techniques were studied and a framework to optimize the task of GP based classification was provided. The major drawback of that approach was a conflict between more than one classifier needed to be handled. In [11], an approach using GP based object detection to locate small objects of multiple classes in large pictures was described. It used a feature set computed from a square input field that contained objects of interest. There were more errors in detection in complex images. The remaining paper is organized as follows: In section 2, the proposed position based fitness evaluation function and algorithm are discussed. In section 3, experimental results are analyzed and a comparative study is performed. Finally in section 4, conclusion and future work are presented.

2. Proposed approach

Some of the challenges faced by Genetic Algorithms include random initial population, random selection of crossover point and dummy iterations etc. The state of art GA algorithms evaluates the string's over all fitness value for the strings either for parent selection or for cross over. The computation of string's overall fitness is shown in figure 2.1.

In this paper, a new position based fitness evaluation function for parent selection and cross over is presented. Reproduction alone could not promote a new search space for exploration. Since position of a gene in a chromosome is more significant, fitness value is assigned to each gene position that greatly increases efficiency of the algorithm. Here, we employ the position based

fitness to parent selection and cross over. To improve the efficiency of the algorithm and to avoid the probability of dummy iterations, crossing site along the string length is rather computed.

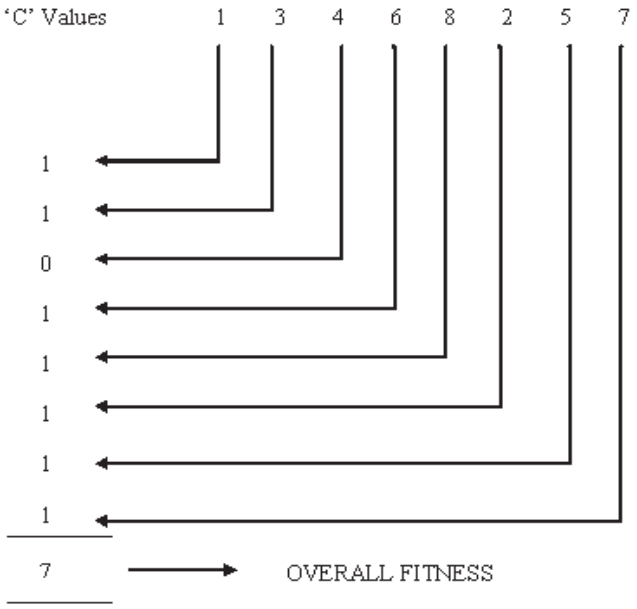

Figure 2.1. (b) Computation of string's overall fitness

2.1 Position based fitness evaluation approach

The initial population contains $\{P_1, P_2 \ldots\ldots P_n\}$.where 'n' is the population size and P_i represents an individual string. The fitness of an individual in a GA 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 function is obtained by a simple combination of the different criteria that gives good result for the given problem domain. The fitness value indicates good solution and corresponds to how close the chromosome is to the optimal one. Position of genes in a chromosome is very vital in deciding its strength.

For computation of chromosome fitness, new position based fitness is introduced. The process of fitness evaluation employing position based and overall fitness are shown in the Figure 2.2

The position based fitness PF_i of a string is computed using the following formula (1),

$$PF_i = 1 + \sum_{j-1}^{i-1} C[Q_i, Q_j]$$

$$PF_i \text{ is position fitness of } i^{th} \text{ element} \quad (1)$$

where the value of $C[Q_i, Q_j]$ is computed using formula (2) and (3).

$$C[Q_i, Q_j] = \begin{cases} 1 & \text{for } Q_i \neq Q_j \text{ and } |Q_i - Q_j| \neq |i \neq j| \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

i, j are the position of queen

Q_i, Q_j are queen

$$C[Q_i, Q_j] = \begin{cases} 1 & \text{for strongest gene} \\ 0 & \text{for weakest gene} \end{cases} \quad (3)$$

The overall fitness of the string OF_i is computed using equations. (4) and (5).

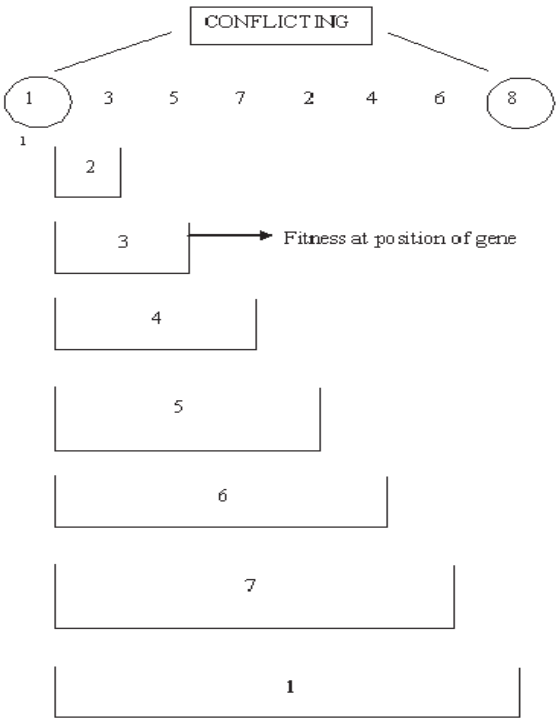

Figure 2.2. Computation of string's position based fitness

$$P_i = \begin{cases} 1 & PF_i = POS(Q_i) \\ 0 & otherwise \end{cases} \quad (4)$$

$$OF_i = \sum_{j\text{-}1}^{n} P_j \quad (5)$$

OF_i is overall fitness of i^{th} string

N is length of string

2.2 Cross over and Mutation Process

Reproduction alone can promote some search space for exploration. So, a cross over is demanded. The crossover generates normal off springs from their parents. Mutation also cannot change many bits in the individuals. Here, weakest point along the string length is chosen for cross over by computing the position based fitness value. Cross over is applied using the equation (6).

Consider POS_I ,.....POS_{sl} is the Position of element in parent Q_i ,Q_{i+1} C_1,....,C_n are conflit position of parent Q_i

$$newQ_i = \begin{cases} Q_i[POS_I] = Q_{i+1}[C_I]\, POS_I = C_I \\ Q_i[POS_I] = Q_i[POS_I]\, POS_I \neq C_I \end{cases} \quad (6)$$

The mutation process is applied using equation. (7).

Consider cr1, cr2,.., crn are crossed position

e_i ,... e_N are repeated element in Q_i

ne_i ,.... ne_N are non redundant element Q_i

pos_i .....,pos_{sl} are position of Q_i

$$Q_i = \begin{cases} Q_i[pos_i] = ne_i\, pos_i \neq cr_i \;\&\; Q_i[pos_i] = e_i \\ Q_i[pos_i] = Q_i[pos_i]\, pos_i = cr_i \end{cases} \quad (7)$$

2.3 Algorithm

begin

Generate initial population;

Evaluate fitness values for each chromosome;

begin

Initialize i as one;

While (i not equal to population size)

begin

Take P_i and P_{i+1} from the population;

Store values of P_i into temp;

Compute position based fitness

Apply crossover at the weak position

Apply multi point mutation;

Calculate fitness of new P_i;

Improve the fitness;

end{*new population construction)*

end*{expected population}*

end {new GA}

The flow of new position based fitness evaluation approach is shown in Figure 2.3.

3. Analysis of experimental results

The position based fitness approach has been tested to solve N Queen Problem. For testing purpose, a custom C program has been developed on Intel Core to Duo Processor @ 2 GHZ, 512MB RAM with Windows XP. Average fitness of the popualtion is a performance indicator of any GA and is computed as the ratio of total fitness of the population to the population size. After reproduction, individuals with a fitness value greater than average grow and those with less than average die. Parameters that are used to test N Queen Problem are tabulated in Table3.1. We have observed that the population's average fitness is 12.18 for position based fitness approach compared to 8.5 for SGA, 8.5 for AGA and 10.12 for EIGA. We have learnt that over all fitness function is evaluated in all state of art GA algorithms. Experimental results of various algorithm for single solution are compared in Table.3.2 and plotted in Figure 3.1. The average fitness for a single solution is 10.5 for new approach; 7.6 for EIGA, 7.1 for SGA and AGA. We have also observed that the other GAs, take comparatively more number of iterations to converge and they take a little more time.

Parameters	Values
Selection Method	**Fitness value**
Crossover Probability	0.4
Mutation probability	0.001
Population Size	8 - 16
String Length	8 - 16

Table 3.1. Parameters used in the Experiment

Time is another important performance indicator for any computational algorithm. In the new approach, expected solutions are

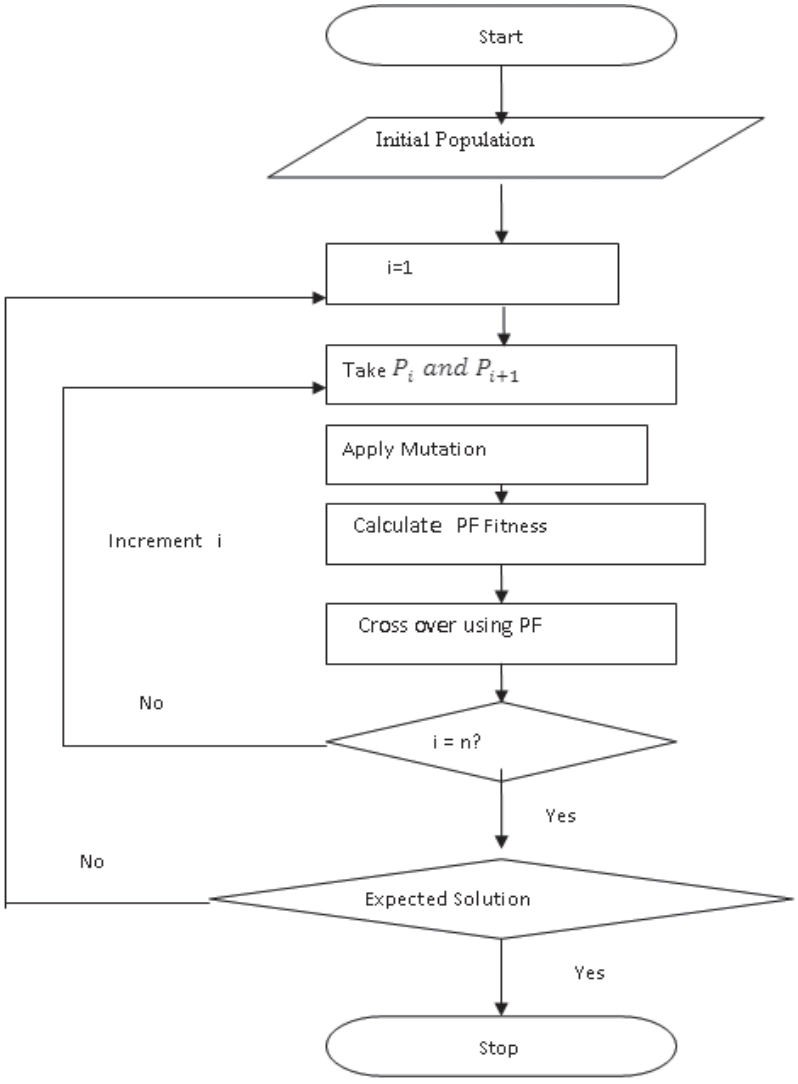

Figure 2.3. System flow

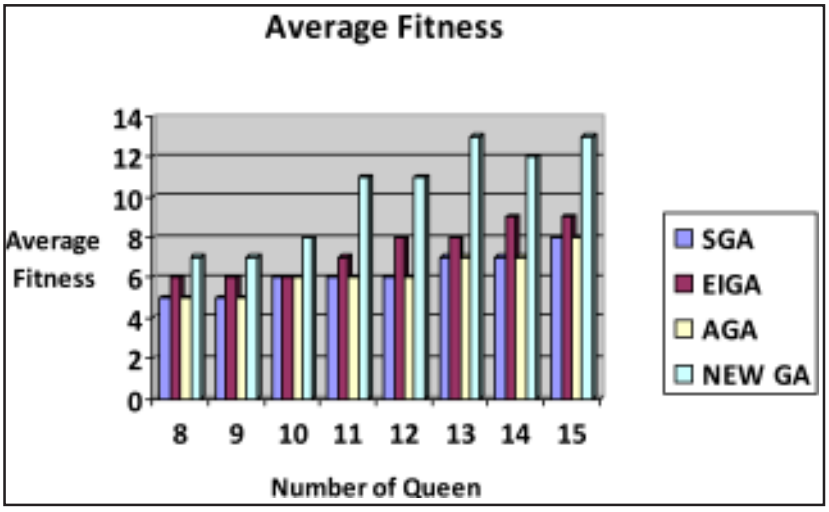

Figure 3.1. Comparison of Average Fitness value

Number Of Queen	Population's average Fitness			
	SGA (with over all fitness)	EIGA (with over all fitness)	AGA (with over all fitness)	New Approach (with position based fitness)
8	5	6	5	7
9	5	6	5	7
10	6	6	6	8
11	6	7	6	11
12	6	8	6	11
13	7	8	7	13
14	7	9	7	12
15	8	9	8	13
16	9	10	9	13
Average fitness	7.1	7.6	7.1	10.5

Table 3.2. Comparison of Average Fitness value

achieved in lesser time compared to other GAs. It is also possible to compute time to arrive at a particular solution. Due to limitation of computational resources, the algorithm was tested for values of N ranging from 8 to 16. Experimental results of various algorithms for computation efficiency are tabulated in Table 3.3 and plotted in Figure 3.2 .The average computational time to arrive expected solutions is 1.93 seconds for new approach, 2.7 seconds for AGA, and 3.72 seconds for EIGA and 7.79 seconds for SGA. It is observed that performance of GA is improved by cross over site selection by computation. The weakest point is selected for crossing to avoid the probability of either reduction in fitness value or dummy iterations. We have also observed that cross over at improper location may sometimes either retain the fitness value of string over iterations.

	Time Taken in Seconds			
No. of Queen	SGA (using over all fitness)	EIGA (using over all fitness)	AGA (using over all fitness)	New Approach (using position based fitness)
8	0.25	0.15	0.00	0.00
9	0.50	0.36	0.20	0.12
10	0.90	0.50	0.30	0.17
11	1.60	0.85	0.40	0.35
12	2.01	1.08	0.72	0.53
13	3.05	2.0	1.35	0.90
14	4.60	3.29	2.54	1.57
15	6.09	5.02	4.10	3.05
16	51.26	20.27	15.09	10.72
Average	7.79	3.72	2.7	1.93

Table 3.3. Comparison of Computational Efficiency

4. Conclusion

A new position based fitness evaluation for Genetic algorithm has been formulated by considering gene's position in a

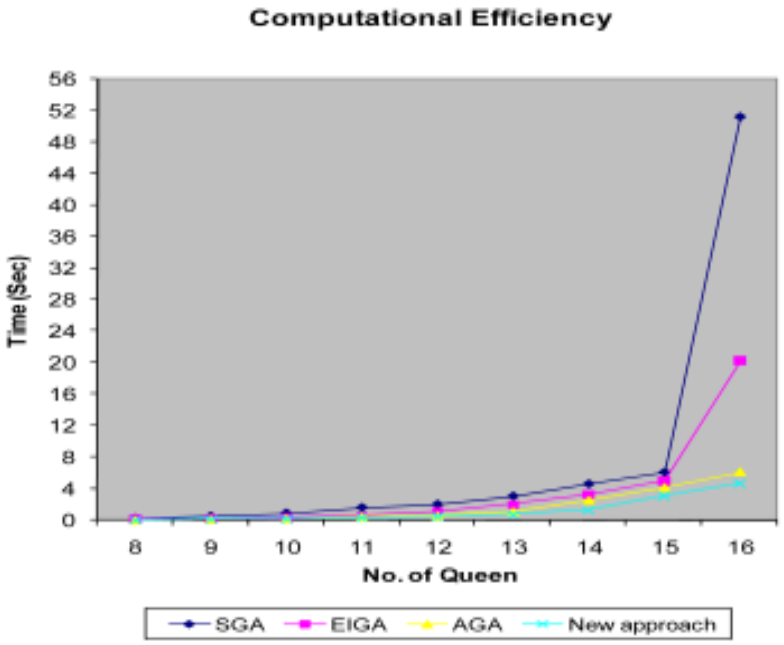

Figure 3.2 Comparison of computational efficiency

chromosome and selecting the weakest point along the string length for crossing over. It was observed that position based fitness has avoided dummy iterations to a larger extent compared with GA employing overall fitness evaluation function. The experimental results show that the new approach performs comparatively better in terms of computational efficiency. A GA based segmentation and number plate recognition system using the new position based fitness approach is presented separately in [12]. This system could be improved by employing data base concepts.

References

[1] Shengxiang Yang, Abbass, Bedau, et al. (2002). Adaptive Crossover in Genetic Algorithms Using Statistics Mechanism, Artificial Life VIII, (MIT Press), p 182.
[2] Marko Bo•ikovic, Marin Golub, Leo Budin, et al. (2003). Solving n-Queen problem using global parallel genetic algorithm, EUROCON Ljubljana, Slovenia.
[3] Mariusz Nowostawski, Riccardo Poli, et al. (1999). Parallel Genetic Algorithm Taxonomy, Kes'99.
[4] Alfonsas Misevicius, Dalius Rubliauskas, et al.(2008). Enhanced Improvement Of Individuals In Genetic Algorithms, *Information Technology And Control*, 37 (3).
[5] Sung Hoon Jung et al. (2009). Selective Mutation for Genetic Algorithms, World Academy of Science, Engineering and Technology.
[6] Dana Vrajitoru et al. (2000). Crossover Improvement For The Genetic Algorithm *In*: Information Retrieval, Université de Neuchâtel, Institut interfacultaire d'informatique, Pierreà.
[7] Andrei Petrovski, Alex Wilson and John Mccall, et al. Statistical identification and optimisation of significant GA factors, The Robert Gordon University School of Computer and Mathematical Sciences, Scotland.
[8] Stephen Karungaru, Minoru Fukumi, Norio Akamatsu, (2009). Detection and Recognition of Vehicle License Plates Using Template Matching, Genetic Algorithms And Neural Networks, *International Journal of Innovative Computing, Information and Control*, 5 (7) 1975—1985.
[9] Abdel- gaied, S. M. (2008). Employing Genetic Algorithms for qualitative Shapes Detection, ICGST-GVIP, 8 (4).
[10] Hajir Jabeen, Abdul Rauf Baig, (2010). Review of Classification Using Genetic Programming, *International Journal of Engineering Science and Technology*, 2 (2) 94-103.
[11] Junior, O. L., Nunes, U. (2008). Improving the Generalization Properties of Neural Networks: an Application to Vehicle Detection, Intelligent Transportation Systems, *In:* 11th International IEEE Conference on, p.310-315, 12 -15.
[12] Vijayalakshmi, P., Ayyappan, M., et al. (2011). GA based Number plate Recognition System, submitted to *Journal of Intelligent Computing*.

[13] Sivanandam, S. N., Deepa, S. N., et al. (2008). Introduction to Genetic Algorithms, Springer publications.
[14] David, E., Goldberg, et al. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley Professional; 1 edition.
[15] Kim-Fung Man, Kit-Sang Tang, Sam Kwong,(1999), Genetic Algorithms: Concepts and Designs, Springer.