Volume 4, Issue 3, March 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Efficient Scheduling Using Multi-Objective Genetic Algorithm for Independent Task

D.Rajeswari* *Dept of Information Technology RMK Engineering College , India*

V.Jawahar Senthil Kumar *Dept of Electroics & Communication Engineering, Anna university, India*

R.Kanaga *Dept of Computer Science & Engineering, RMK Engg College, India*

Abstract— Distributed computing is the coordinated use of various compute resources of different capabilities to optimize certain performance features. Scheduling is one of the important steps to efficiently exploit the capabilities of distributed computing systems. Finding such optimal schedules is an NP-complete problem requires the development of meta-heuristic approaches. Scheduling is used to decide how to order these independent tasks and how to commit resources between the varieties of possible independent tasks. Schedules thus obtained can be evaluated using several criteria that may conflict with one another which require multi objective problem formulation. In this paper a Genetic Algorithm based procedure is developed for efficient scheduling of independent tasks that minimizes the makespan and flowtime simultaneously in a distributed computing system. Experimental results verify the effectiveness of the multi-objective Genetic Algorithm.

Keywords— Scheduling, Multi-objective, Genetic Algorithm, Independent task, Optimization.

I. INTRODUCTION

Generally, a distributed computing suites of different high-performance machines, interconnected with high-speed links, for efficiently solving the computationally intensive problems that have different computational requirements [1, 2]. The strength of distributed systems are derived from their ability to find appropriate resources for computing needs. Scheduling is mapping a set of tasks to a set of resources to efficiently utilize the capabilities of that resources and finding the best resource is one of the key problems in the distributed computing environment. However, finding optimal schedules for set of tasks in distributed environment is NP-complete problem [3,4]. Distributed environment with only independent tasks are consider here, i.e., there is no communications between the tasks. It is consider that many independent users submit the tasks to a collection of computational resources that are presented in the distributed environment. The scheduling of these independent tasks is being performed statically. Static Scheduling assumes that the information regarding all the resources and tasks required by an application is available at the scheduling moment. The placement of an application is static; the main advantage drawn from this approach is that computation cost can be estimated before the actual execution of the application. On the other hand, it is difficult and in some cases it is impossible, to know a priori the state of the resources and the specifications for the tasks that must be scheduled.

Static scheduling is exploited in different types of analysis and environments. One of the most common uses of static mapping is predictive analysis i.e., setting up an efficient schedule for some set of independent tasks jobs that are to be executed in the future, and it can also work out when the sufficient time or computational resources are available to complete the execution. Post mortem analysis also described by using the static scheduling. It is used to evaluate the efficiency of dynamic scheduler when it is complete their execution and also analyze how the systems are using the available resources in the distributed computing environment. Another use of static scheduling is for ``what if '' simulation studies i.e., to work out the effect that losing or gaining a particular piece of hardware will have. In future, high-powered computational grids [5] will also be able to utilize static scheduling mapping techniques to distribute resources and computational power. Hence the wide applicability of static scheduling makes it an important area for ongoing research.

It is also considered that in the distributed computing environment each and every machine executes a single task at a time, in the order assigning the independent task to each machine. There is no termination in-between the execution of the task. The effectiveness of scheduling algorithms can be analyzed based on several standards. The most important thing is makespan and flowtime. Makespan is defined by the time which the distributed system finishes the last task. Flowtime is the total of finishing time all the tasks. Hence the scheduling problem requires solving by using multi-objective scheduling. The main objective is resolving the multi-objective problem to minimizing the makespan and flowtime in the distributed computing.

In this paper, Genetic Algorithm (GA) is used to generate random optimal schedules that may not generate satisfactory solutions because of the fuzziness in considering both makespan and flowtime. Hence multi-objective is applied to select the best optimal schedule from the random schedules generated by considering both the objectives.

The remainder of this paper is organized as follows. Section 2 provides a look back at previous solution methodologies. Section 3 defines the computational environment parameters that were varied in the simulations. Description of proposed Genetic Algorithm with Multi-objective based procedure is presented in Section 4. Section 5 presents the results for different problems simulated and finally, Section 6 concludes with finishing remarks and future work.

II. RELATED WORK

Related work in the literature was surveyed to select a set of heuristics appropriate for the distributed computing environment is considered here. This section is a sampling of related literature and is not meant to be comprehensive. The heuristics presented in [4] are concerned with allotting independent tasks onto machines such that the makespan is minimized. Braun et al. [2] present eleven heuristics for mapping and scheduling independent jobs onto heterogeneous computing environment, and the makespan was minimized. SmartNet [6] is a resource management system for HC systems that employs various heuristics to allocate tasks to machines, considering resource and task heterogeneity. Other single heuristic approaches for the problem include Local Search [8], Simulated Annealing [7,9] and Tabu Search [7]. GAs for scheduling is used in several works [2, 7, 10, 11, 12].

Some hybrid heuristic approaches have also been addressed for the problem. For dynamic job scheduling on large-scale distributed systems the hybridization of GA, SA and TS heuristics is reported in [7]. In these hybridizations a population-based heuristic, like GAs, is combined with two other Local Search heuristics, such as TS and SA, which deal with only one solution at a time. Ritchie and Levine [13, 14] combined an Ant Colony Optimization algorithm and TS algorithm for the problem. Some modern heuristics approaches have been recently addressed for the problem that include the use of AI techniques [15], use of predictive models to schedule jobs [16], Particle Swarm Optimization [17] , Fuzzy based scheduling [18] and economic-based approaches [19] .

Very few research have been made to minimize flowtime or both flowtime and makespan on distributed computing environments. [20] Investigates the efficacy of 5 important heuristics for minimizing makespan and flowtime on HC environments with various characteristics of jobs and resources. However the objectives are evaluated separately here. Extensive study on the usefulness of GAs for designing efficient Grid schedulers for simultaneously minimizing makespan and flowtime is presented in [21].

III. PROBLEM STATEMENT

Real-world distributed computing systems, such as a computational cloud, are complex combinations of hardware, software and network components. Let $J = \{J_1, J_2,..., J_n\}$ denote the set of jobs that are independent of each other to be scheduled to r resources $R = \{ R_1, R_2 ,..., R_m\}$ within the distributed computing environment. As the scheduling is performed statically an estimation of the computational load of each job and the computing capacity of each resource in the distributed computing environment is assumed to be available a priori. This formulation is practically applicable as the computing capacity and computational needs can be known from user specifications, historic data or prediction. Computational load of workload of tasks are known from the Cornell Theory Center [23]. This information can be contained in an Expected Time to Compute (ETC) matrix where each position ETC[j][r] indicates the expected time to compute job j in resource r. One row of the ETC matrix specifies the estimated execution time for a given task on each machine. Similarly, one column of the ETC matrix consists of the estimated time for each job in a given machine.

Table 1. An example ETC matrix

	Machine 1	Machine 2
Job 1	4	3
Job 2	3	5
Job 3	8	6
Job 4	7	8
Job 5	5	8

In order to simulate various possible scheduling problems as realistically as possible, benchmarks of instances are generated to capture different characteristics of distributed systems such as consistency of computing, heterogeneity of resources and heterogeneity of jobs [2][22]. Provided with J, R and ETC the objective is to minimize both makespan and flowtime.

IV. PROPOSED SOLUTION

Traditional techniques of optimization and search do not fare well over a broad spectrum of problem domains and are not robust for multi-objective optimization problems [24]. Genetic Algorithm has been applied for static scheduling in distributed computing environment [21] that may not generate satisfactory solutions because of the fuzziness in considering the multiple objectives. To overcome the above problem, it is proposed to use multi-objective GA for solving the problem. The multi-objective GA is used to generate optimal schedules.

A. Genetic Algorithm

Genetic algorithms (GAs) are optimized and computerized search based on the mechanics of natural genetics and natural selection. It comes under the class of the non-traditional search and optimization techniques used for searching large solution spaces [25]. The template given in the algorithm has been used for this work

Initialization: Generate the initial population P of n individuals

Fitness: Evaluate the fitness of each individual of the population P.

while (stopping criteria not met)
{
Selection: Select a subset of individuals from P.
Crossover: Perform cross-over on each of the
Selected individuals with probability P_c
Mutation: With probability P_m, mutate each individual
Fitness: Evaluate the fitness of each individual in the new population.
}
Return Best found solution

1) *Representation*: The representation of individuals of the population is a key issue in evolutionary algorithms. Individuals referring to a solution of the problem [26]. Representation determines which type of operators could be used to insure the evolution of the individuals and also affects the feasibility of individuals. Different types of representations are addressed in the literature. A direct representation is considered here the advantage of being that it determines a feasible schedule in a straightforward way and is obtained as follows. Each individual is represented as a vector, of size equal to the number of tasks to be scheduled and entry in position j indicates the machine to which job j is assigned. The values in this vector are in the range (1 to number of machines). It is to be noted that a machine number can appear more than once.

2) *Initialization*: During initialization, a set of individuals are computed randomly from a uniform distribution to form a solution space of the desired population size.

3) *Fitness function*: The problem is taken as a multi-objective problem to achieve minimum makespan that is, the time when last task is finished and flowtime that includes minimizing the sum of finalization times of all the tasks. These two criteria are defined as follows:

$$\text{Makespan}: \quad \min_{S_j \in Sched} \{ \max_{j \in Jobs} F_j \} \tag{1}$$

$$\text{Flowtime}: \quad \min_{S_j \in Sched} \{ \sum_{j \in Jobs} F_j \} \tag{2}$$

where F_j denotes the time when job j finalizes, Sched is the set of all possible schedules and Jobs the set of all jobs to be scheduled.

Since makespan and flowtime are measured in the same unit and the weighted sum function is used here. Since the makespan and flowtime values are in incomparable ranges, due to the fact that flowtime has a higher magnitude order over makespan, and its difference increases as more jobs and resources are considered. For this reason, the value of mean flowtime, flowtime/number of resources, is used to evaluate flowtime. Both values are weighted in order to balance their importance. Fitness value is calculated as:

$$\text{Fitness} = w_1 . \text{makespan} + w_2 . \text{mean flowtime} \tag{3}$$

Where weights w_1 and w_2 are chosen such that $w_1+w_2=1$, which can be fixed by roper tuning. As the aim is to minimize the combined objective function, the individual that has a lower fitness value will be the best one.

4) *Selection*: The good individuals are selected by using selection operators and if forms a mating pool to which the crossover operators will be applied. The selection operators used here is Binary Tournament Selection. The Selection is done by making two randomly selected individuals to participate in the tournament and choose the best among them. This scheme probabilistically duplicates some chromosomes and deletes others, where better mappings have a higher probability of being duplicated in the next generation. Investigations prove that this is comparatively better than other selection methods suitable for the problem [21].

5) *Crossover:* The aim of any evolutionary algorithm is to obtain descendants of better quality that will feed the next generation and enable the search to explore new regions of solution space not explored yet. Crossover achieves this by selecting individuals from the parental generation and interchanging their genes, to obtain new individuals. Depending on the representation of individuals, a single point crossover and fitness based crossover is used in this problem.

In single point crossover, the crossover operation selects a random pair of individuals and chooses a random point in the first individual. For the sections of both individuals from that point to the end of each individual, crossover exchanges machine assignments between corresponding tasks. Every individual is considered for crossover with a certain probability.

In fitness based crossover, a crossover mask is computed using the fitness of the individual as follows.

For all tasks *i = 1 to number of tasks*

$individual[i] = individual_1[i]$,

if $individual_1[i] = individual_2[i]$ (4)

otherwise $individual[i] = individual_1[i]$ with probability p = f2/(f1 + f2);

$individual_2[i]$, with probability 1-p; (5)

where f1 = fitness($individual_1$) and f2 = fitness($individual_2$).

If the two parent individuals have similar fitness then the genes of the new descendants are calculated with probability ~0.5 .When there is a large difference in the fitness of the two parent individuals schedules, then it is quite probable that a chromosome of new structure will be obtained.

6) *Mutation:* Mutation alters one or more gene values in an individual from its initial state resulting in new individuals helping to arrive at better solution than was previously possible. This also prevents the population from stagnating at any local optima. Mutation occurs during evolution according to a user-definable mutation probability usually set fairly low otherwise the search will turn into a primitive random search. In this paper, swap mutation operator and based on the direct representation of the individual, taking into account the specific need of load balancing of resources a rebalancing mutation is used.

The swap mutation operator interchanges the allocation of two jobs since movement of jobs between resources is considered to be effective. This operator should be applied to two jobs assigned to two different resources.

In rebalancing mutation type the solution is improved by rebalancing the machine loads and then mutating it. First, a resource *r*, from most overloaded resources is chosen at random; further, we identify two jobs, j_1 and j_2 such that job j_1 is assigned to another resource r_2 whose *ETC* for the resource r_1 is less than or equal to the *ETC* of job j_2 assigned to r_1; jobs j_1 and j_2 are interchanged. Secondly, if rebalancing was not possible, then rebalancing is done by *move*. After this, a mutation is applied. This completes one generation of the GA and the best value is stored. All the individuals available at the end of the first iteration will be treated as parents for the second iteration. This procedure is repeated for the number of iterations as given by the user.

V. TEST RESULTS

The simulation model in [2] is based on expected time to compute (ETC) matrix for 512 tasks and 16 processors. The instances of the benchmark are classified into 12 different types of ETC matrices according to the three following metrics: job heterogeneity, machine heterogeneity, and consistency. In ETC matrix, task heterogeneity is defined as the amount of variance possible among the execution times of the tasks. Machine heterogeneity, represents the possible variation of the running time of a particular task across all the machines, both of which can be classified into low or high heterogeneity.

Also an ETC matrix is said to be consistent, if machine r_x has a lower execution time than machine r_y for job j_k , then the same is true for any job j_i . The ETC matrices that are not consistent are inconsistent ETC matrices. A semi consistent ETC matrix is characterized by an inconsistent matrix which has a consistent sub-matrix of a predefined size. All instances consist of 512 tasks and 16 processors and are labeled *u- m-nn-oo* that represents the following

u means uniform distribution (used in generating the matrix).

m means the type of inconsistency (*c*–consistent, *i*–inconsistent and *s* means semi-consistent).

nn indicates the heterogeneity of the jobs (*hi*–high, and *lo*–low).

oo indicates the heterogeneity of the resources (*hi*–high, and *lo*–low).

All the four combinations of crossover and mutation types namely single point crossover with swap mutation, fitness based crossover with swap mutation, and single point crossover with rebalancing mutation and fitness based crossover with rebalancing mutation are implemented on all the problem instances. It is found that single point crossover with rebalancing mutation is found to produce better results for consistent and semi-consistent matrices and single point crossover with swap mutation is found to produce good results for inconsistent matrix types. GA also requires tuning of certain parameters like crossover rate, mutation rate and weights of the objective function. This is done by varying the parameters over a range of values and testing it on the instances for an average of 10 runs.

The following values for various parameters of GA yielded the best result on average, which is used in obtaining the results on all 12 static instances. The Fig.1 shows that the comparison among different levels of consistency likes consistent, semi- consistent, in- consistent. Compare to other instances high task and machine heterogeneity having high makespan, flowtime and fitness value.

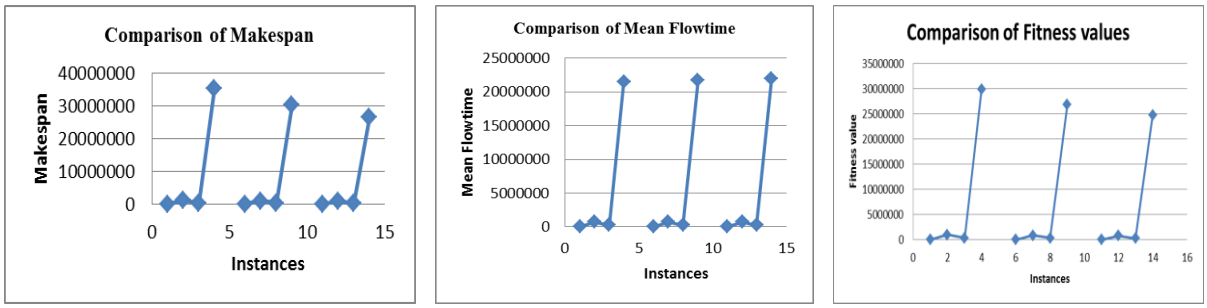

Fig.1 Comparison of makespan and mean flowtime for various consistency levels

Population size	: 75
No of generations	: 100
Selection operator	: Binary Tournament
Cross-over operator	: Fitness based crossover
Mutation operator	: Swap (consistent and semi- consistent matrix instances), Rebalancing (inconsistent matrix instances)
Cross-over probability	: 0.8
Mutation probability	: 0.1

Weights w1 : 0.6
Weights w2 : 0.4

The Fig.2 shows that the Makespan, Mean Flowtime and Fitness values for different crossover & mutation probability values.

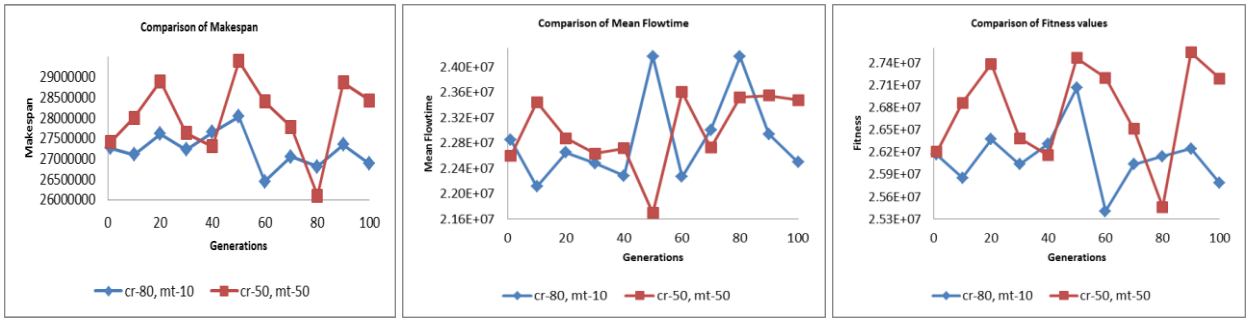

Fig.2 Makespan, Mean Flowtime, Fitness values for different probability of crossover and mutation.

Genetic algorithm is implemented with the above setting and schedules obtained in all the generations are collected. From these generated schedules 50 of schedules are selected with lower combined objective function. For all 12 types of instances tests were carried out and the combined makespan and mean flowtime are computed.

The combined fitness values for the optimal schedules are shown in Fig. 2 which shows better results for GA with multi-objective.

VI. CONCLUSION

In this paper, the proposed GA with multi-objective technique designed to find schedules for independent tasks that minimizing the makespan and flowtime simultaneously and reveals the quality of schedules in comparison to a weighted GA which also simultaneously minimize both the objectives. Since statically scheduling independent jobs in distributed computing environments is useful in predictive analyses, impact studies, and post-mortem analyses. The proposed multi-objective GA technique can find better schedules satisfying multiple objectives for several benchmark problems and it seems a promising approach to scheduling in distributed computing environments. However, further work could be carried out with the algorithm investigating different parameter settings, different genetic operators, optimizing more objectives and also testing it in a more realistic environment. Investigations also can be extended to considering several forms of distributed computing scheduling, such as scheduling jobs with precedence constraints or in dynamic environments.

REFERENCES

[1] S. Ali, T. D. Braun, H. J. Siegel, and A. A. Maciejewski, “Heterogeneous computing”, Encyclopedia of Distributed Computing, Kluwer Academic, 2001.

[2] H.J. Braun et al, “A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems” Journal of Parallel and Distributed computing, 61(6), 2001.

[3] D. Fernandez-Baca, "Allocating modules to processors in a distributed system," IEEE Transactions on Software Engineering, Vol. SE-15, No. 11, Nov.1989, pp. 1427-1336.

[4] O. H. Sbarra and C. E. Kim, "Heuristic algorithms for scheduling independent tasks on non-identical processors," Journal of the ACM, Vol. 24, No. 2, Apr. 1977, pp. 280-259.

[5] I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufman, New York, 1998.

[6] R.F. Freund, M. Gherrity, S. Ambrosius, M. Campbell, M. Halderman, D. Hensgen, E. Keith,T. Kidd, M. Kussow, J. D. Lima, F. Mirabile, L. Moore, B. Rust, and H. J. Siegel, “Scheduling resources in multiuser, heterogeneous, computing environments with SmartNet,” in ``7th IEEE Heterogeneous Computing Workshop (HCW '98)," pp. 184_199, 1998.

[7] A. Abraham, R. Buyya, and B. Nath. “Nature’s heuristics for scheduling jobs on computational grids,”. In The 8th IEEE International Conference on Advanced Computing and Communications (ADCOM 2000), India,2000

[8] G. Ritchie and J. Levine, *A fast, effective local search for scheduling independent jobs in heterogeneous computing environments*, Technical report, Centre for Intelligent Systems and their Applications, School of Informatics, University of Edinburgh, 2003.

[9] A. Yarkhan and J. Dongarra, *Experiments with scheduling using simulated annealing in a grid environment.* In 3rd International Workshop on Grid Computing (GRID2002), 232–242, 2002.

[10] J. Page and J. Naughto, “Framework for task scheduling in heterogeneous distributed computing using genetic algorithms,” Artificial Intelligence Review, 24:415–429, 2005.

[11] V. Di Martino and M. Mililotti. “Sub optimal scheduling in a grid using genetic algorithms,” Parallel Computing, 30:553–565, 2004.

[12] A.Y. Zomaya and Y.H. The, “Observations on using genetic algorithms for dynamic load-balancing,” IEEE Transactions On Parallel and Distributed Systems, 12(9):899–911, 2001.

[13] G. Ritchie, *Static multi-processor scheduling with ant colony optimization & local search.* Master's thesis, School of Informatics, University of Edinburgh, 2003.
[14] G. Ritchie and J. Levine, "A hybrid ant algorithm for scheduling independent jobs in heterogeneous computing environments," In 23rd Workshop of the UK Planning and Scheduling Special Interest Group (PLANSIG 2004), 2004.
[15] H. Casanova, A. Legrand, D. Zagorodnov, and F. Berman,"Heuristics for scheduling parameter sweep applications in grid environments," In Heterogeneous Computing Workshop, 349–363, 2000.
[16] Y. Gao, H. Rong and J.Z. Huang, "Adaptive grid job scheduling with genetic algorithms," Future Generation Computer Systems, 21(1), 151-161, 2005,
[17] A. Abraham, H. Liu, W. Zhang and T.G. Chang, *Job Scheduling on Computational Grids Using Fuzzy Particle Swarm Algorithm*, 10th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems, B. Gabrys et al. (Eds.): Part II, Lecture Notes on Artificial Intelligence 4252, 500-507, Springer, 2006.
[18] K.P. Kumar, A. Agarwal, and R. Krishnan, "Fuzzy based resource management framework for high throughput computing,". In Proceedings of the 2004 IEEE International Symposium on Cluster Computing and the Grid, 555–562, 2004. IEEE Computer Society.
[19] R. Buyya, *Economic-based Distributed Resource Management and Scheduling for Grid Computing*. PhD thesis, Monash University, Melbourne, Australia, 2002R. E. Haskell and C. T. Case, "Transient signal propagation in lossless isotropic plasmas (Report style)," USAF Cambridge Res. Lab., Cambridge, MA Rep. ARCRL-66-234 (II), 1994, vol. 2.
[20] H.Izakian , A. Abraham and V. Snasel, **"**Comparison of Heuristics for Scheduling Independent Tasks on Heterogeneous Distributed Environments," In Proceedings of International Joint Conference on Computational Sciences and Optimization 1,8-12, 2009.
[21] Javier Carretero, Fatos Xhafa and Ajith Abraham, "Genetic Algorithm Based Schedulers for Grid Computing Systems", International Journal of Innovative Computing, Information and Control 3, 6, 2007.
[22] M. Maheswaran, S. Ali, H.J. Siegel, D. Hensgen, and R.F. Freund," Dynamic mapping of a class of independent tasks onto heterogeneous computing systems," Journal of Parallel and Distributed Computing, 59(2):107–131, 1999.
[23] S. Hotovy, *Workload evolution on the Cornell theory center IBM SP2,* In Job Scheduling Strategies for Parallel Processing Workshop, IPPS'96, 27–40, 1996.
[24] K Deb, *Multi-objective Optimization using Evolutionary Algorithms*, John Wiley & Sons, Chichester, 2001.
[25] J. H. Holland, *Adaptation in Natural and Artificial Systems,* University of Michigan Press, AnnArbor, MI, 1975.
[26] M. Srinivas and L. M. Patnaik, *Genetic algorithms: A survey*, IEEE Comput. 27, 6 (June 1994),17-26

