Intelligent genetic algorithm: a toy model application

Jaime Mora Vargas[1], Neil Hernández Gress[1], and Miguel González Mendoza[1]

Instituto Tecnológico y de Estudios Superiores de Monterrey,
Campus Estado de México, Carr. Lago de Guadalupe Km. 3.5
Atizapán de Zaragoza, Estado de México, México
{jmora, ngress, mgonza}@itesm.mx
http://www.cem.itesm.mx

Abstract. We argue that the performance of a genetic algorithm can be improved by the codification of its operative rates into the chromosome. In the case of the flowshop problem the claim is that mutation and crossover rates allow the genetic algorithm to adapt better and faster than the traditional genetic algorithm. We support our claim with a simple "toy model" with two instances of flowshop problem, an special case of scheduling with multiple applications to industrial problems. We refer to that genetic algorithm as Intelligent Genetic Algorithm (IGA) since its ability to self-modify its operative characteristics.

1 Introduction

Genetics algorithms have been applied to various optimization problems (Goldberg [5]). In this paper, a genetic algorithm is improved using local search procedures, and self-adaptation rates of genetic operators.

In the literature, many hybrid algorithms ([10, 4, 7, 14]) of GA's were proposed for flowshop optimization problems, those algorithms are a combination of traditional GA and artificial intelligence techniques (*e.g* tabu search, simulated annealing). In those studies, it was clearly shown that the performance of GA's for scheduling problems was improved using neighborhood search algorithms.

Flowshop problems are included into scheduling problems. Great efforts are devoted to its economical importance. Unfortunately, finding optimal scheduling for a general production process is an NP-hard problem (Garey and Johnson, [3]). This means that traditional operations research techniques such as integer programming (branch and bound techniques [13]) or dynamic programming (Bellman and Dreyfus [1]) are not adequate to deal with large scale problems. Therefore, the interest of many researchers has been oriented to find good solutions (not always a global optimum) in a reasonable time. Considering this, the use of metaheuristics techniques are well suited.

A major issue for metaheurists is the fine-tuning for parameters, *i.e* tabu list length (for tabu search), initial temperature (for simulated annealing) or crossover and mutation rates (for genetic algorithms). In this article, a modification is made to traditional GA, creating the "intelligent" Genetic Algorithm

Advances in Artificial Intelligence Theory
Research on Computing Science 16, 2005, pp. 175-184

(IGA) which doesn't need fine tuning of crossover and mutation rates. This technique was originally used with simple optimization landscapes ([17]) and to travelling salesman problem instances ([18]) with encouraging results.

This article is organized as follows: section 2 includes a general description of the flowshop problem as well as details for the 2 treated problems, also a brief introduction to GA is provided. Section 3 contains specific details about representation of flowshop problem, Section 4 includes details about the experiments done, Section 5 shows the most representative graphics and comments about experimentation and Section 6 has the conclusions and future work of this article.

2 Problem Description

Flowshop problems in particular, are a special case of scheduling problems and scheduling problems arise in the combinatorial optimization area. General assumptions for flowshop are (more details in Dudek *et al.* [2]): 1) jobs are to be processed by multiples stages sequentially, 2) there is one machine at each stage, 3) machines are available continuously, 4) a job is processed on one machine at a time without preemption, and 5) a machine processes no more than one job at a time. For this paper, n jobs are processed in the same order on m machines. Considering this, this paper works with a sequencing problem of flowshop scheduling of n-jobs.

Considering notation from Ishibuchi [8], the completion time and processing time of job j on machine i are $t_C(i,j)$ and $t_P(i,j)$ respectively. The n-dimensional vector $x = (x_1, x_2, \ldots, x_n)$ represents the sequence of n jobs to be processed, where x_k denotes the k-th processing job. Completion time for each sequence x is calculated by:

$$t_C(1, x_1) = t_P(1, x_1) \tag{1}$$

$$t_C(i, x_1) = t_C(i-1, x_1) + t_P(i, x_1), i = 2, 3, \ldots, m, \tag{2}$$

$$t_C(i, x_k) = t_C(i-1, x_{k-1}) + t_P(i, x_k), k = 2, 3, \ldots, n, \tag{3}$$

$$\begin{aligned} t_C(i, x_k) = max\{t_C(i-1, x_k), t_C(i, x_{k-1})\} + t_P(i, x_k), \\ i = 2, 3, \ldots, m; k = 2, 3, \ldots, n \end{aligned} \tag{4}$$

Flowshop scheduling problems are to determine the sequence of x of n jobs based on a given scheduling criterion. According to Johnson's work [11] the reduction of makespan if one of the most extended criteria, also reduction of tardiness is employed. The makespan is the completion time of the last job:

$$Makespan(\mathbf{x}) = t_C(m, x_n) \tag{5}$$

Also maximum tardiness is other criteria used, it is defined as the maximum tardiness of the n jobs to schedule, that is:

$$\begin{aligned} & Tardiness = \\ & max\{t_C(m,1) - d_1, t_C(m,2) - d_2, \ldots, t_C(m,n) - d_n\} \\ & t_C > d \end{aligned} \tag{6}$$

where d_i represents due date for job i.

2.1 Genetic Algorithms

Genetic algorithms are one of the heuristic optimization algorithms widely used by many researchers in solving various problems, were introduced by Holland [6]. Genetic algorithms mimic the mechanism of genetic evolution in biological nature. In biological terms, it consist of a chromosome composed of genes, each one of them with several alleles, into the optimization field, this chromosome is a string that usually represents a possible solution to some optimization problem, each string is composed of bits with specific values. Initially, a number of chromosomes form an initial pool of solutions. The process of crossover and mutation will be carried out in the pool, after that an evolution is completed and new chromosomes (offspring) will be generated.

GAs have two major processes. First, GAs randomly generate new solutions. Second, the evolution of those initial solutions is done according to the genetic operators such as reproduction (selection of the fittest), mutation (exploration operator) and crossover (exploitation operator).

3 Problem Representation

3.1 Chromosome

Configuration for flowshop problem using GA uses a string base codification, where each individual in the population represents a possible sequence of jobs to be done. For example, the sequence $x = (1, 3, 2, 4)$ represents a sequence of 4 jobs, where job 1 is done first, followed by jobs 3, 2 and 4. That kind of representation is currently used to solve scheduling problems using GA.

3.2 Genetic operators

In this paper, two genetic operators were used : crossover and mutation in order to exploit results (crossover) and explore solutions (mutation). The crossover operator is the two-point order crossover and for mutation, it is used the shift change, details for such operators can be found in [14]. Such operators work selecting a random number and compare to the operation rates if it is smaller then the operator is applied to that individual.

In order to improve search for new solutions, a local search procedure was used, this procedure consisted in the permutation of size 1 for every population

element, selecting the best one. For example, for individual $(1, 2, 3)$, the possible neighbors would be $(2, 1, 3), (3, 2, 1)$ and $(1, 3, 2)$ this local search avoid the use of large populations, also it doesn't require important computational resources.

The reproduction of individuals is made using the so-called tournament reproduction of size t, where t=2, it function by selecting by random N/t sets of t elements and passing the element with the highest fitness of each set to the next generation, this procedure is done t times to assure that the population number N remains constant. Ties broken by random.

For the IGA, the standard genetic operators for binary codification [5] were used.

3.3 Fitness function

The fitness function used combines two objectives, minimize makespan and max tardiness. Using equations 5 and 6 it is possible to create a global equation for the i-th individual, that is:

$$f(i) = -log(MaxTardiness_i) - log(Makespan_i) \tag{7}$$

The *log* function is used in order to re-normalize the values of makespan and tardiness. As the GA nature is maximize, the use of "-" allows to get better results (*i.e.*, small makespan and small tardiness)

3.4 Intelligent GA

The "Intelligent Genetic algorithms" are a modification of traditional genetic algorithms in which the crossover and mutation rates are codified into the chromosome, for this paper, a string of 5 bits was used to codify in binary. In this manner, the max value (in decimal) is $2^5 = 32$ so it is possible to configure value rates between 0 and 1 with an interval of $1/32 = 0.03125$. The translation process consists in translate from binary to decimal and divide that value by the max value possible. Using this configuration, a complete individual is by example $[1, 2, 3, 4|00101|11000]$ representing that the first job to be processed is job 1, followed by jobs 2, 3 and 4, also the mutation probability is $00101 = 0.15625$ and the crossover rate is $11000 = 0.75$. Two types of genetics operators were used, those applied to the flowshop configuration and those applied to the operators rates configuration, for the flowshop configuration, two-point order (for crossover) and shift change (for mutation) were used. The traditional two parents, two points crossover and change between 0 and 1 mutation operator were used to the chromosome section that codifies operators rates. The sequence used was: first apply operators to flowshop section followed by the application of operators to codification rates section.

This self-codification allows the algorithm to avoid selecting optimal mutation and crossover rates, a time-consumption task that must be completed before run any standard GA. Special attention must be on IGA since self-adaptation capacity allows to apply GA into time-dependent landscapes.

4 Experiments

The experiments were realized using a flowshop problem of 5 machines-10 jobs (5M10J) and 5 machines-30 jobs (5M30J).

Table 1. Processing times 5M10J

	j1	j2	j3	j4	j5	j6	j7	j8	j9	j10
m1	32	1	61	42	62	61	3	97	26	9
m2	21	27	87	45	59	24	71	34	20	28
m3	10	42	66	75	41	24	3	36	85	74
m4	51	19	23	85	86	81	93	31	75	23
m5	33	45	58	97	91	85	30	38	17	51

Table 2. Due date times, 5M10J

job	Due date
j1	674
j2	396
j3	431
j4	369
j5	626
j6	597
j7	790
j8	437
j9	656
j10	780

Table 3. Processing times 5M30J (jobs 1-10)

	j1	j2	j3	j4	j5	j6	j7	j8	j9	j10
m1	32	1	61	42	62	61	3	97	26	9
m2	21	27	87	45	59	24	71	34	20	28
m3	10	42	66	75	41	24	3	36	85	74
m4	51	19	23	85	86	81	93	31	75	23
m5	33	45	58	97	91	85	30	38	17	51

Table 1 shows the processing times for job n in machine m, for example, job 2 in machine 2 takes 27 time units, job 10 in machine 5 takes 51 time units. Table 2 shows the due date for each job. Tables 3,4 and 5 include the processing times for 5M30J, table 6 shows the due dates for 5M30J problem.

Table 4. Processing times 5M30J (jobs 11-20)

	j11	j12	j13	j14	j15	j16	j17	j18	j19	20
m1	47	5	35	88	84	40	79	94	56	13
m2	29	35	81	94	77	30	19	75	47	30
m3	43	19	49	85	79	55	34	93	64	50
m4	25	30	83	80	83	32	45	88	49	62
m5	50	40	85	78	77	49	80	48	44	20

Table 5. Processing times 5M30J (jobs 21-30)

	j11	j12	j13	j14	j15	j16	j17	j18	j19	20
m1	43	38	65	92	78	45	67	71	57	31
m2	38	8	76	93	83	32	37	80	53	37
m3	47	27	41	79	79	35	38	69	55	33
m4	49	28	51	78	78	46	78	80	45	15
m5	43	17	78	83	82	46	35	86	61	33

Table 6. Due date times, 5M10J

job	Due date	job	Due date	job	Due date
1	674	11	674	21	436
2	396	12	707	22	456
3	431	13	569	23	764
4	369	14	671	24	645
5	626	15	509	25	738
6	597	16	465	26	451
7	790	17	490	27	611
8	437	18	492	28	746
9	656	19	429	29	420
10	780	20	613	30	651

5 Results

The experiments carried out where done considering the problems mentioned in previous section, the objective of that experiments was to compare standard GA versus Intelligent GA. Both GA types used a population size of 500 individuals, and 510 generations. The provided results considers the average value for 20 runs per experiment.

For the standard genetic algorithms several experiments where done using different crossover and mutations rates. In this paper, results for fixed mp (mutation probability) and cp (crossover probability) are provided, the graphs shows results for $mp = 0.4$-$cp = 0.1$, $mp = 0.01$-$cp = 0.01$ which are compared with the IGA performance.

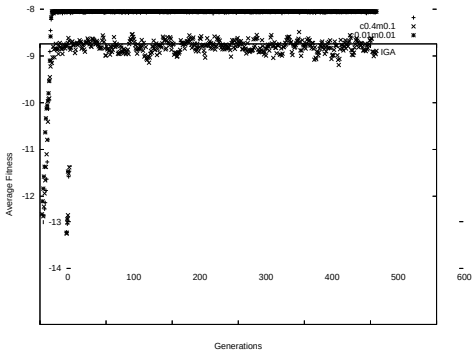

Fig. 1. Fitness comparison, 5 machines, 10 jobs.

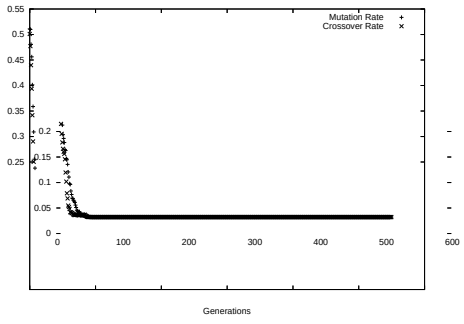

Fig. 2. Operative Rates, 5 machines, 10 jobs.

Figure 1 shows the comparison between average fitness for $mp = 0.4$-$cp = 0.1$, $mp = 0.01$-$cp = 0.01$ and IGA considering 5M10J problem, data showed are the average fitness for the entire population. For this case, $mp = 0.01$-$cp = 0.01$ and

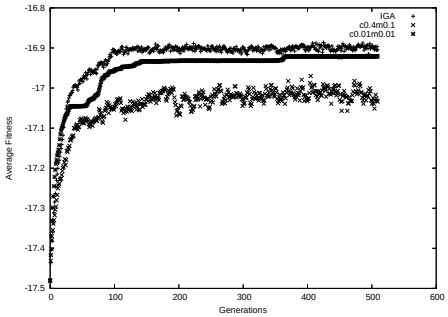

Fig. 3. Fitness comparison, 5 machines, 30 jobs.

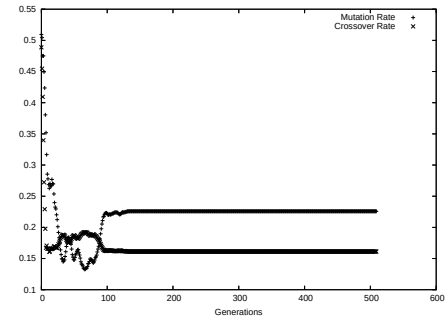

Fig. 4. Operative Rates, 5 machines, 30 jobs.

IGA have similar results. All three configurations find rapidly the optimal value (at generation 25 aprox). It is important to mention that 5M10J problem has a feasible space 10! = 3628800 size, which is very simple to solve using exhaustive-search procedures, the reason to use such landscape is to gain experience with the IGA and track their results.

The operative rates for genetic operators are showed in figure 2, one the IGA has find the optimum, it reduces its mutation and crossover rate, the latter achieving its stable value faster than mutation rate, this is related with the impact of the operator, *i.e.* mutation is a more destructive operator than crossover, then the search for new possible solutions continues by more time than the exploitation of results already found.

Figure 3 shows the comparison between $mp = 0.4$-$cp = 0.1$, $mp = 0.01$-$cp = 0.01$ and IGA considering 5M30J problem, data showed are the average fitness for the entire population. As this graph shows, it is clear that IGA have better performance than $mp = 0.01$-$cp = 0.01$. The convergence of IGA take more time than the others, the reason is that the IGA have to modified its operative ranges. Also, $mp = 0.01$-$cp = 0.01$ has a better performance than $mp = 0.4$-$cp = 0.1$, although $mp = 0.4$-$cp = 0.1$ goes faster to a local optimum, moreover $mp = 0.4$-$cp = 0.1$ has more changes between every generation this is because the mutation and crossover rates are relatively high, allowing to loose good solutions.

Figure 4 shows the mutation and crossover rate along the 510 generations of the experiment, note the changes in the rates, first descending to values of 0.15 for mutation and 0.16 for crossover. Again and similar to 5M10J mutation rate takes more values before get stable. Both crossover and mutation rates remains with the same value once an optimum is reached, and off course by the population effect (all individual have the same operation rates).

6 Conclusions

This article presents an application of a called "Intelligent Genetic Algorithms", a type of genetic algorithm which is able to modify its operational rates in order to achieve a global optimum. Such characteristic could be very important specially for problems in which the environment changes over time. Also IGA avoid fine-tuning of parameters, mostly always a time-consuming task.

The examples treated in the article are flowshop problems with 5 machines-10 jobs and 5 machines-30 jobs problems, in both examples treated, IGA have a better performance than standard GA, however it is possible to prove that by adjusting standard GA parameters it could perform better that IGA. Then the main application for IGA seems to be problems in which the environment changes over time, since the IGA can adapt to changes modifying its operative rates. In the case standard GA once the change occurs and since the majority of population is in the "old optimum" it can not be able to move to the new optima.

References

1. Bellman, R.E. and Dreyfus, S.E., Applied Dynamic Programming, Princeton University Press (1962)
2. Dudek R.A., Panwalkar S.S. and Smith M.L., The Lessons of Flowshop Scheduling Search, Operations Research, Vol. 47 No. 1 (1992) 65-74
3. Garey, M. and Johnson, D.,Computers and intractability: A guide to the theory of NPCompleteness. Freeman and Co.San Francisco (1979)
4. Glass C. A., Potts C. N., and Shade P., Genetic algorithms and neighborhood search for scheduling unrelated parallel machines, Univ.Southampton, Southampton, U.K., Preprint Series OR47 (1992)
5. Goldberg, D.E., Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley Publishing Company. Reading, Massachusetts (1989)
6. Holland, J.H., Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor MI (1975)
7. Ishibuchi H., Yamamoto N., Murata T., and Tanaka H., Genetic algorithms and neighborhood search algorithms for fuzzy flowshop scheduling problems, Fuzzy Sets Syst., Vol. 67 (1994) 81-100
8. Ishibuchi H., Murata T. and Tomioka S., Effectiveness of Genetic Local Search Algorithms, Proc. 7th International Conference on Genetic Algorithms (1997) 505-520
9. Ishibuchi H. and Murata T., A Multi-Objective Genetic Local Search Algorithm and Its Application to Flowshop Scheduling IEEE Transactions on systems, man, and cyberneticspart C: applications and reviews, Vol. 28 No. 3 (1998) 392-403
10. Jog P., Suh J. Y., and Gucht D. V., The effects of population size, heuristic crossover and local improvement on a genetic algorithm for the traveling salesman problem, Proc. 3rd Int. Conf. Genetic Algorithms (1989) 110-115
11. Johnson S. M., Optimal Two- and Three-stage Production Schedules With Setup Times Included, Naval Research Logistics Quarterly, Vol. 1 No. 1 (1954) 61-68
12. Kursawe F., A variant of evolution strategies for vector optimization, Parallel Problem Solving from Nature, H.-P. Schwefel and R. Manner,Eds., Vol. 15 , Berlin Germany (1993) 754-770
13. Lawler, E.L. and Wood, D.E., Branch and Bounds Methods: A survey", Operations Research, Vol. 14 (1966)
14. Murata T. and Ishibuchi H., Performance evaluation of genetic algorithms for flowshop scheduling problems, Proc. 1st IEEE Int. Conf. Evolutionary Computat. (1994) 812-817
15. Murata T. and Ishibuchi H., MOGA: Multi-objective genetic algorithms, Proc. 2nd IEEE Int. Conf. Evolutionary Computat. (1995) 289-294
16. Schaffer J. D., Multi-objective optimization with vector evaluated genetic algorithms, Proc. 1st Int. Conf. Genetic Algorithms (1985) 93-100
17. Stephens C.R. and Mora J., Effective Fitness as an Alternative Paradigm for Evolutionary Computation I: General Formalism, Genetic Programming and Evolvable Machines, Vol. 1, No. 4 (2000) 363-378
18. Stephens C.R. and Mora J., Effective Fitness as an Alternative Paradigm for Evolutionary Computation II: Examples and Applications, Genetic Programming and Evolvable Machines, Vol. 2, No. 1 (2001) 7-32