Genetic Algorithms for Solving Scheduling Problems in Flexible Manufacturing Cells

ANTÓNIO FERROLHO
Electrical Engineering Department
Superior School of Technology of the Polytechnic Institute of Viseu
Campus Politécnico de Repeses, 3504-510 Viseu
PORTUGAL
http://www.estv.ipv.pt/PaginasPessoais/antferrolho

MANUEL CRISÓSTOMO
Institute of Systems and Robotics
University of Coimbra
Polo II, 3030-290 Coimbra
PORTUGAL
mcris@isr.uc.pt

Abstract – In this paper, scheduling problems in Flexible Manufacturing Cells (FMC) are studied. The scheduling objective is to minimize the makespan. We used a genetic algorithm (GA) for solving the optimization scheduling problem. We have developed one FMC with industrial characteristics with the objective of studying scheduling problems in these types of manufacturing systems (single machine scheduling, flow-shop scheduling and job-shop scheduling). The practical results obtained from the FMC for the various scheduling problems show the efficiency of GA in solving these problems.

Key-Words: - Genetic Algorithm, Scheduling, single machine, Flow-shop, Job-shop and Flexible Manufacturing Cell.

1 Introduction

This paper investigates scheduling problems in Flexible Manufacturing Cells. The scheduling objective is to minimize the makespan, and we used a genetic algorithm for solving the optimization scheduling problem.

A genetic algorithm is based on the principle of survival of the fittest. The basic idea is to start with a set of solutions and to generate a set of new solutions by applying some well-defined operators on the old ones. Then, some solutions are selected to form a new set with which another iteration is started, and so on until some stopping criterion is met. To develop an effective genetic algorithm for a specific problem, some more ingredients need to be provided [2]. In general, a GA consists of the following steps:

Step 1. Initialization.
Step 2. Evaluation of fitness function.
Step 4. Genetic operation.
Step 5. Repeat steps 2 to 4.

We have developed one FMC with industrial characteristics with the objective of studying the scheduling problems in these types of manufacturing systems. The FMC was used to study the following scheduling problems: single machine scheduling, flow-shop scheduling and job-shop scheduling. The results obtained in the scheduling problems of the FMC show the efficiency of the GA in solving these problems.

2 Flexible Manufacturing Cell

Today two or more CNC machines are considered a Flexible Manufacturing Cell (FMC) and two or more cells are considered a Flexible Manufacturing System (FMS) [4]. FMS are typical real time concurrent systems composed of a number of computer-controlled machine tools, automated material handling and storage systems [5] and [7]. FMS and Computer Integrated Manufacturing (CIM) systems using industrial control networks [6].

An FMC with industrial characteristics was developed with the objective of studying the scheduling problems in these types of manufacturing systems. Fig. 1 presents the layout of the FMC developed. The FMC is comprised of four sectors, which are controlled by PCs and different software. The four sectors are:

- The manufacturing sector, made up of two CNC machines (mill and lathe), one ABB IRB140 robot and one buffer.

- The assembly sector, made up of one Scorbot ER VII robot, one small conveyor and an assembly table.
- The handling sector, made up of one great conveyor.
- The storage sector, made up of five warehouses and one robot ABB IRB1400.

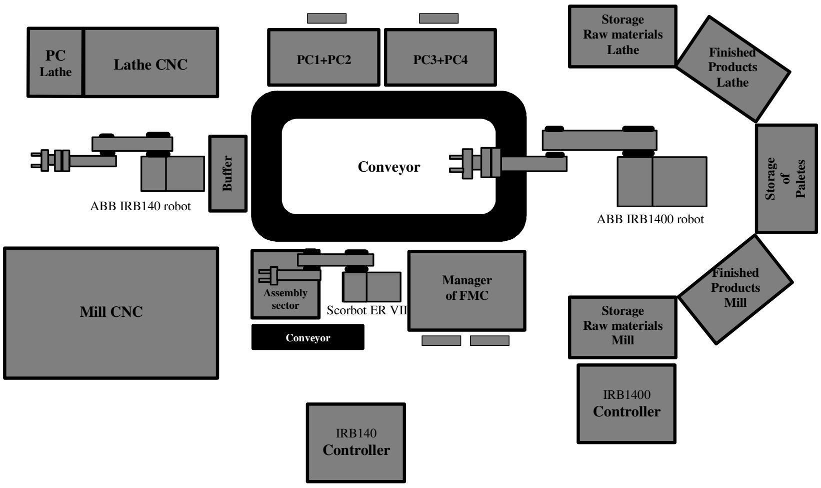


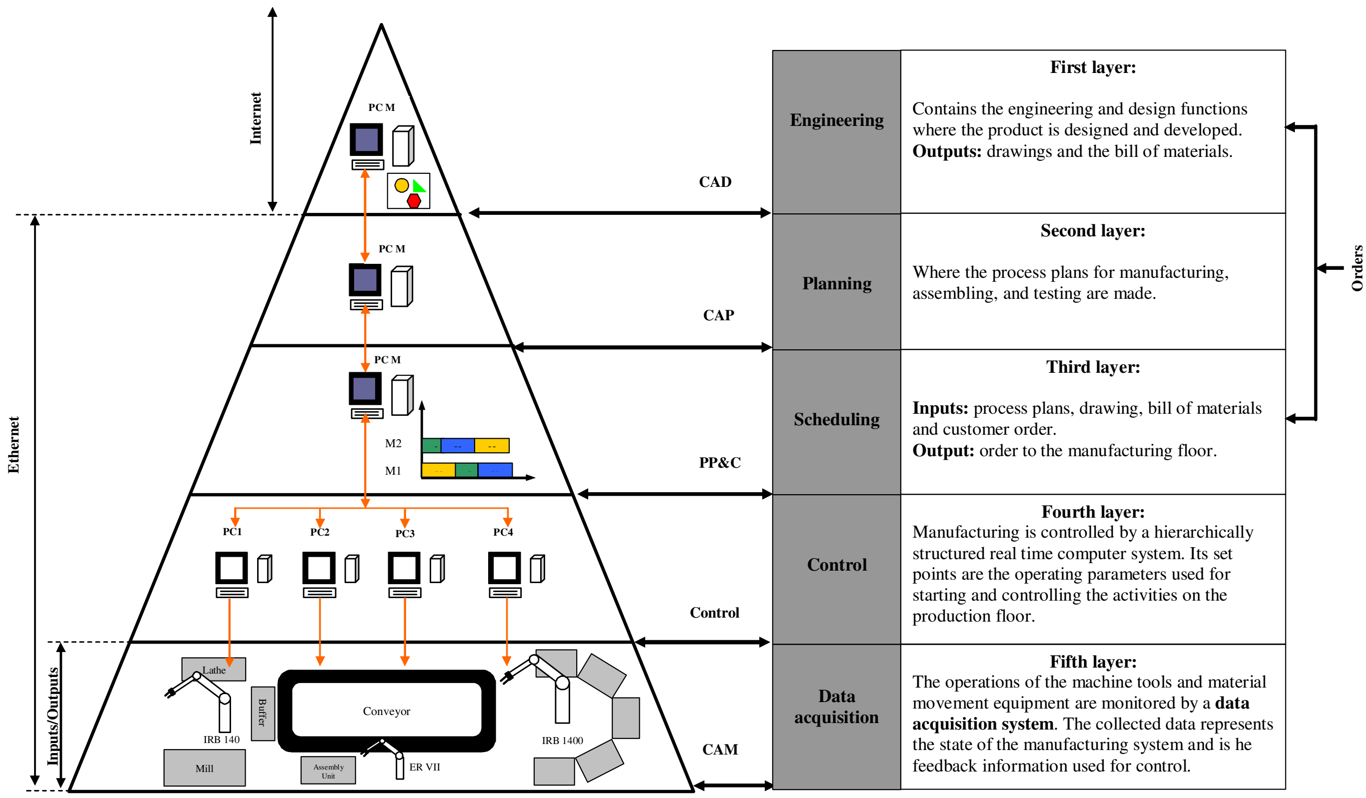
Fig. 1 – Layout of Flexible Manufacturing Cell.

Control of existing equipment in each sector is carried out by four computers: PC1 – manufacturing sector, PC2 – assembly sector, PC3 – handling sector and PC4 – storage sector. Coordination, synchronization and integration of the four sectors is carried out by the of FMC computer Manager.

Whenever a PC needs to communicate with another PC in a lower hierarchical layer, it sends a message through the Ethernet net. The messages consist of a command and a group of parameters: COMMAND|PARAMETERS. The PC that receives the message then sends two messages: the first indicates to the client that the command was received and whether or not it will be executed. The second message indicates if the command was well executed or if an error occurred.

The hierarchical structure implemented in FMC is shown in Fig. 2. Messages cannot be sent from a higher hierarchical layer to an inferior layer without going through the intermediate layers [8].

The first layer contains the engineering and design functions where the product is designed and developed. The outputs of this activity are the drawings



PCM – FMC Manager: Engineering + Planning + Scheduling
PC1 – Manufacturing sector: Mill + Lathe + Robot IRB140 + buffer
PC2 – Assembly sector: Robot Scorbot ER VII + small conveyor
PC3 – Handling sector: big conveyor
PC4 – Storage sector: Robot IRB1400 + Automatic Storage

Fig. 2 – FMC hierarchical structure.

and the bill of materials.

The second layer is process planning. Here the process plans for manufacturing, assembling and testing are carried out.

The third layer is scheduling. The process plans together with the drawing, the bill of materials and the customer orders are the input to scheduling. The output of scheduling is the release of the order to the manufacturing floor. The PCM – FMC Manager is responsible for the engineering, planning and scheduling activities.

The fourth layer is control. Manufacturing is controlled by a hierarchically structured real time computer system (PC1, PC2, PC3 and PC4). Its set points are the operating parameters used for starting and controlling the activities on the production floor.

The fifth layer is data acquisition. The operations of the machine tools and material movement equipment are monitored by a data acquisition system. The collected data represents the state of the manufacturing system and is the feedback information used for control.

The central computer (FMC Manager) controls all the production of FMC, calling the several computers and communication nets of data, that allow control and supervision in real time of the operations, picking up and processing the flows of information of the various resources. Fig. 3 presents the main window of the software developed for the PC - FMC Manager.

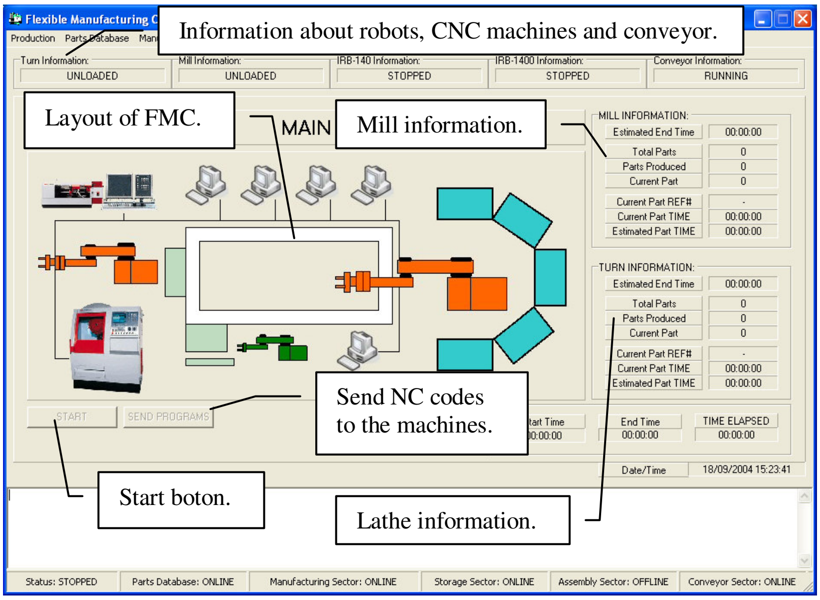


Fig. 3 – Main windows of the software developed for the PC - Manager of FMC.

The PC - FMC Manager is responsible for:

- Developing and designing new products to manufacture – the engineering layer.
- Production plans, assemblies and product tests – the planning layer.
- Finding the optimum processing sequence so as to optimize CNC machine use – the scheduling layer.
- Coordination, integration and control of all the sectors in FMC.
- Controlling and supervising the entire operation in FMC in real time.
- Maintaining a database of jobs to manufacture, including the respective NC programmes.
- Synchronizing the various sectors so as to produce variable lots of different types of parts depending on the customer's orders.
- Monitoring the current state of production.
- Guaranteeing tolerance of failures, safety and coherence of data.

3 Scheduling: problem formulation

Assuming we know the manufacturing products, their manufacturing schedule and the available resources for execution, the scheduling task consists in determining the schedule passage of the products (jobs) for the resources and defining the attribution of resources, considering the times at the beginning and end of operations, with the objective of optimizing one or more performance measures. This description is particularized for the case of the scheduling problems in production systems, as is the case of the Flexible Manufacturing Cells. In fact, scheduling problems appear associated to very different areas of activity, such as arranging school schedules, defining the order in which airplanes land at an airport, choosing the order of execution of different programs in a computer, and so on.

Supposed we have n jobs $\{J_1,...,J_n\}$ to be processed in m machines $\{M_1,... ,M_m\}$. Possible solutions are determined by $(n!)^m$ [10]. The processing job in a machine is designated by operation and characterized by the respective processing time p_{ij} (job i, machine j). For each job technological restrictions are defined, that is the necessary operations sequence of job processing. Thus, the scheduling problem consists in determining a sequence of passage of jobs to the respective machines so that it is: 1) compatible with technological restrictions and 2) optimal, according to performance measures. Table 1 and Fig. 4 clarify the notation used.

Description	Symbol	Remarks
Number of jobs	n	
Job i	J_i	
Machine j	M_j	
Operation time	O_{ij}	
Processing time	p_{ij}	
Read time for J_i	r_i	

Due date for J_i	d_i	
Allowance for J_i	a_i	$a_i=d_i-r_i$
Waiting time of J_i preceding the respective k th operation	W_{ik}	
Total waiting time of J_i	W_i	$W_i = \sum_{k=1}^{m} W_{ik}$
Completion time of J_i	C_i	$C_i = r_i + \sum_{k=1}^{m} \left[W_{ik} + p_{ij(k)} \right]$
Flow time of J_i	F_i	$F_i=C_i-r_i$
Lateness of J_i	L_i	$L_i=C_i-d_i$
Tardiness of J_i	T_i	$T_i=\max\{0, L_i\}$
Earliness of J_i	E_i	$E_i=\max\{-L_i, 0\}$
Idle time on machine M_j	I_j	$I_j = C_{\max} - \sum_{i=1}^{n} p_{ij}$

Table 1 –Scheduling problem notation.

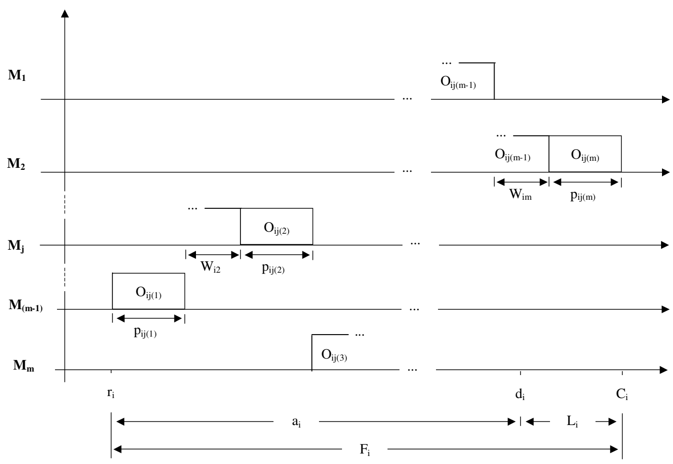


Fig. 4 – Gantt diagram of a typical job J_i.

4 A Genetic Algorithm

The pseudocode of a basic genetic algorithm is shown in Fig. 5. We can see that genetic algorithms start with a population P of n individuals, where each individual codifies a solution to the problem. The evaluation of each individual's performance is based on a function of fitness evaluation. The best ones will tend to be the progenitors of the following generation, allowing them to transmit their features to the next generations [3] and [9].

The stopping criteria can be:

- When a solution with a certain value of fitness is reached (for the type of problems demanding a cheaper solution, without finding the best solution);
- When a certain time limit is reached (for problems demanding solutions in time);
- When a certain number of generations is reached.

Algorithm GA	
{ t := 0;	*//counter*
Starts_population (P, t);	*//starts a population on n individuals*
Evaluation (P, t);	*//evaluates individuals fitness*
Repeat until (t = d)	*//tests criteria (duration, fitness, and so on.)*
{ t := t + 1;	*//increases the counter of generations*
Selection_of_parents (P, t);	*//selects the couples for crossover*
Recombination (P, t);	*//accomplishes selected couples crossover*
Mutation (P, t);	*//disturbs the group generated by crossover*
Evaluation (P, t);	*//evaluates new fitness*
Survive (P, t)	*//selects survivors*
}	
}	
t – actual generation; ***d*** – criterion to finish the algorithm; ***P*** – population	

Fig. 5 – Basic pseudocode of a Genetic Algorithm.

4.1 Genetic Operators

Genetic operators allow the manipulation chromosomes and are responsible for the evolution of the quality of solutions during the performance of the GA. From the mathematical point of view, operators allow the creation of new searching points in the solution's space, based on the actual population's members.

There are countless genetic operators [1] and [2]. In this work we used: selection, crossover, mutation and elitism.

Selection means to choose, among the individuals that belong to a population, those who will be used to create descendants for the next generation and how many descendants each one can generate. If brief, selection is the choice of parents' chromosomes. The chromosomes that belong to a population are considered to be filtered in order to generate better individuals according to a certain criterion.

Crossover is an operator of "coupling", which allows new chromosomes to be produced (offspring) through the exchange of partial information among pairs of chromosomes (parents). In order to avoid the repetition of genes in chromosomes, we used order crossover (OC).

The aim of mutation is to assure the genetic diversity of a population of chromosomes. Mutation consists of random changes of one or more genes in chromosomes. The mutation rate is very low, as in natural populations.

To assure that the best member of a population is not eliminated, there is the elitism operator [1], which passes the best chromosomes or the best ones of a generation onto the next generation.

The algorithm applied to the scheduling problems of FMC is basically defined by the following steps:

1) The population individuals constitute possible solutions for the scheduling problems.
2) Each individual is constituted by one chromosome.
3) One chromosome codes the solution through characters (genes).
4) The initial population can be generated randomly.
5) The evaluation function is determined (fitness) for each individual.

5 Scheduling Problems in the Flexible Manufacturing Cell

Genetic Algorithms were used, as a tool to solve the scheduling problems in the Flexible Manufacturing Cell we developed. The scheduling problems studied and implemented in FMC were:

- Single machine scheduling problem.
- Flow-shop scheduling problem.
- Job-shop scheduling problem.

We classify scheduling problems according to four parameters: *n/m/A/B*. The parameter *n* is the number of jobs, *m* is the number of machines, *A* describes the flow pattern (*F* for the flow-shop case and *G* for the general job-shop case), *B* describes the performance measure by which the schedule is to be evaluated. When *m*=1, *A* is left blank.

5.1 A Genetic Algorithm for Solving a Single Machine Scheduling Problem

The *n!* permutations of the *n* jobs constitute all the possible solutions for the problem *n/1//B*.

Table 2 presents the processing times and due dates of 6 jobs. After applying the Genetic Algorithm to the resolution of the problem, we obtained the following result S=(J6, J2, J1, J3, J5, J4).

J_i	1	2	3	4	5	6
d_i	7	3	8	12	9	3
p_i	1	1	2	4	1	3

Table 2 – Due dates and processing times for the jobs.

For the schedule result we calculated the optimal T_{max}. We found T_{max}=1, as shown in Table 3.

Job $J_{i(k)}$	Completion Time $C_{i(k)} = \sum_{l=1}^{k} p_{i(l)}$	Due Date $d_{i(k)}$	Lateness $L_{i(k)}$	Tardiness $T_{i(k)}$
6	3	3	0	0
2	4	3	1	1
1	5	7	-2	0
3	7	8	-1	0
5	8	9	-1	0
4	12	12	0	0

Table 3 – Completion time, due date, lateness and tardiness.

5.2 A Genetic Algorithm for Solving Flow-shop Scheduling Problem

Table 4 presents the processing times of 8 jobs in two machines (Mill and Lathe). For this problem the total number of schedules is $(8!)^2$. In this problem, we found the maximum flow time (F_{max}) for the optimal schedule.

Job	Mill	Lathe
21	2	10
22	3	7
23	8	2
24	7	5
25	9	3
26	4	4
27	1	9
28	6	8

Table 4 – Processing time on machine.

After genetic algorithm application to the problem $8/2/F/F_{max}$, we arrived at the result presented in Fig. 6, in which the sequence is S={J27, J21, J22, J26, J28, J24, J25, J23} with F_{max} = 49.

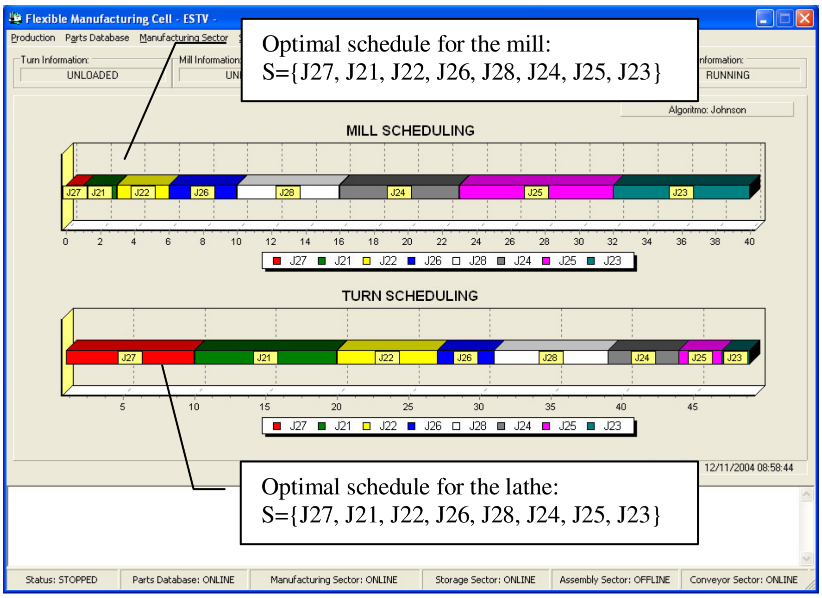


Fig. 6 – Gantt diagram for the $8/2/F/F_{max}$.

5.3 A Genetic Algorithm for Solving Job-shop Scheduling Problem

Genetic algorithm was applied in the resolution of the Job-shop scheduling problem, in FMC. Table 5 presents the processing order and the times of 9 jobs. Jobs 1, 2, 3 and 4 require mill first and then lathe. Jobs 5 and 6 require lathe first and then mill. Only job 7 is to be processed on mill alone. Jobs 8 and 9 require lathe alone. This problem is of the type *9/2/G/*F_{max}.

Processing order and times				
Job	First Machine		Second Machine	
1	Mill	8	Lathe	2
2	Mill	7	Lathe	5
3	Mill	9	Lathe	8
4	Mill	4	Lathe	7
5	Lathe	6	Mill	4
6	Lathe	5	Mill	3
7	Mill	9	-----	-----
8	Lathe	1	-----	-----
9	Lathe	5	-----	-----

Table 5 – Processing order and time.

After genetic algorithm application we arrived at the following result:

- Processing sequence for the Mill, S_{Mill}={J4, J3, J2, J1, J7, J5, J6};
- Processing sequence for the Lathe, S_{Lathe}={J5, J6, J8, J9, J4, J3, J2, J1}.

From this we see that the flow time is 44 (F_{max}=44) for an optimal schedule.

6 Conclusion

In this paper, we used Genetic Algorithms to solve scheduling problems in Flexible Manufacturing Cells (FMC). An FMC has been developed. We presented the layout and hierarchical structure of the FMC. The scheduling problems studied and implemented in this FMC were: single machine scheduling problem, flow-shop scheduling problem and job-shop scheduling problem.

Some scheduling problems are very difficult to solve. The difficulty is not in modelling but in computation, since there is a total of $(n!)^m$ possible schedules, and each one must be examined to select the one that gives the minimum makespan (completion time – C_{max}) or optimizes whatever measure of effectiveness is chosen. Scheduling problems for 3 jobs/m machines up to n jobs/m machines have proven very difficult to solve. For this type of problems, if the GA has been developed correctly, we can obtain good solutions, maybe even the optimal schedule. For any scheduling problem, the GA has a big possibility to find the optimal solution.

GA constitute a new research instrument, capable of being used in several areas of human activities. Their main advantage lies in genetic diversity: for each step of the search process, a group of candidates is considered and involved in the creation of new candidates.

Obtaining a good result by the Genetic Algorithms will depend on the code method, on the types of genetic operators used, on the fitting of parameters, and so on. A GA in its basic form can be insufficient for a certain type of problem. For example the code bit-string can be inadequate; selection through the proportionality of fitness can drive to premature convergence; crossing and mutation operators can make descending solutions unviable; and an inadequate adjustment of the parameters can decrease global performance. For this reason, the decisions on each step of the creation of the algorithm should be analysed carefully, because there are many possibilities to implement a GA and the relationship among the objects of decision are not generally sufficiently clear and simple.

References:

[1] Davis, L.; *Handbook of Genetic Algorithms*; International Thomson Computer Press, 1996.

[2] Goldberg, D. E.; *Genetic Algorithms in Search, Optimization, and Machine Learning*; Addison-Wesley, New York, 1989.

[3] Holland, J.; *Adaptation in Natural and Artificial Systems*; University of Michigan Press, 1975.

[4] Mikell P. Groover; *Automation, Production Systems, and Computer Integrated Manufacturing*; Prentice Hall, New Jersey, 2001.

[5] Najjari, H.; Steiner, S.J.*; Integrated and intelligent control system for a flexible manufacturing cell*; Industrial Electronics, 1997. ISIE'97, Proceedings of the IEEE International Symposium, Volume 1, July 1997, pp. 165-170.

[6] Kim B. H., Cho, K.H. and Park K.S.; *Analysis and feedback control of Lon works-based network systems for automated manufacturing*; 2000 IEEE International Conference on Systems. Man and Cybernetics, Univ of Ulsan, Ulsan, South Korea.

[7] Wang Jiacun and Deng Yi; *Incremental modelling and verification of flexible manufacturing systems*; Journal of Intelligent Manufacturing, vol. 10, nº 6, 1999, pp.485-502.

[8] U. Rembold, B. O. Nnaji and A. Storr; *Computer Integrated Manufacturing and Engineering*; Addison-Wesley, 1993.

[9] A. Henni, M. Koudil, K. Benatchba, H. Oumsalem, K. Chaouche*; A parallel environment using taboo search and genetic algorithms for solving partitioning problems in codesign*; WSEAS Transactions on Systems, vol. 3, pp. 8-13, January 2004.

[10] Simon French; *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop*; John Wiley & Sons, New York, 1982.