Transactions on Information and Communications Technologies vol 8, © 1995 WIT Press, www.witpress.com, ISSN 1743-3517

Genetic algorithms efficiency in flow shop scheduling

S. Fichera[a], V. Grasso[b], A. Lombardo[b], E. Lo Valvo[b]
[a]*Istituto di Macchine, Università di Catania, Italy*
[b]*Dipartimento di Tecnologia e Produzione Meccanica, Università di Palermo, Italy*

Abstract

This paper concerns Genetic Algorithms efficiency in solving the pure flow-shop scheduling problem. Different formulations of the crossover operator and setting of control parameters are investigated in order to obtain enhanced performances in the determination of minimum makespan schedules. A lower bound based ending criterion is also introduced to obtain computation fastening.

1 Introduction

Heuristic scheduling algorithms have been extensively developed in the past four decades for improving the productive performances of production systems, in widely different fields of manufacturing where a set of different products requires an effective solution of the sequencing problem.

The first attempts to develop Flow Shop heuristic sequencing algorithms suitable for solving real world problems (i.e. not limited by the number of stations and/or by the number of parts to be worked), have been based on some knowledge of the influence of the entity and distribution of processing times on sequencing; the "constructive" algorithms proposed by Petrov, Gupta, Campbell Dudeck and Smith and many others [3-4], mostly based on Johnson's rule [5] adaptations, belong to this category.

Afterwards, the concept of a neighbourhood search of a seed has been introduced [6,7,15], utilising different techniques to deterministically explore a greater portion of the solution space, such as pairswapping, merging and insertion search. These techniques are based on the evidence that the larger will be the solution space explored, the better will be the solution obtained;

consequently, a reiteration of the search is very often part of the optimisation strategy, using the previous solution as the input for a new search cycle, until an improvement is found.

The conventional sequencing algorithms are characterised by a search space determined by the choice of the seed and of the swapping rule, and consequently also by a self-contained stopping rule. They can be considered as deterministic algorithms, in the sense that each of them is unmodifiable in the search strategy and that a new solution is accepted only if it leads to a goal improvement. Consequently, conventional search algorithms often get stuck in a local minimum. On the contrary, some new "methaeuristic" algorithms have been recently proposed [16], as Simulated Annealing, Tabu Search and Genetic Algorithms, that are stochastic search algorithms allowing to sample more effectively large search spaces, as occurs for scheduling problems [8]. Moreover, the stochastic nature of these algorithms allows to avoid the risk of pitfalls in local minima, typical of the conventional search ones. Genetic Algorithms [1,2] are based on selecting pairs of solutions from an actual population, and combining them to produce new solutions. Particularly, a Genetic Algorithm develops a family of solutions with individuals in competition with each other, and the evolution of that population is obtained transferring the beneficial adaptations gained during the search from the chromosomes of the parents to the ones of the offspring.

The aim of this paper is to investigate the effectiveness of some alternative formulations of the crossover genetic operator and tune the setting of control parameters in solving the flow-shop scheduling problem. The evaluation of the GA scheduling algorithms obtained combining the different genetic operators and setting of control parameters is concluded presenting a comparative analysis of the obtained results with the ones achieved using two conventional search algorithms, as well as with lower bound makespan values.

2 Manufacturing systems management and flow shop scheduling.

In recent years an ever growing effort of research in production management has taken place, with the aim of developing ever more efficient and realistic shop scheduling algorithms, to face the large capital investments needed to install automated manufacturing systems.

As a matter of fact, for manufacturing environments devoted to produce or to assembly items in high volume, but with diversified options in order to match the customer demand, flexible, highly productive but costly system configurations, as Flexible Flow Lines (FFL) and Flexible Assembly Lines (FAL) have been recently developed. Moreover, the application of Group Technology concepts to low and medium volume production of heterogeneous mix, trough the development of the cellular manufacturing organisation, led to system configurations characterised by flexible automation and unidirectional material flow, as Flexible Manufacturing Cells (FMC), allowing, even for highly diversified production, to move from a job-shop towards a flow-shop configuration. The increasing role of scheduling in production management together with the systems flexibility is established as the consequence of sensitivity of the system performances to the operative planning, that notably affects the interactions among the allowable resources. On the other hand, it is well known that the use of production systems characterised by flexible automation involves higher fixed costs for the firm and therefore a proper return of investments can be obtained only through an efficient scheduling of the system [9].

As to the schedule efficiency, it is extensively accepted in literature that in flexible manufacturing environments particularly to be pursued are the goals of reaching high production rates, together with the shortening of job lead times, the lowering of Work in Process and an enhanced material flow and production control. Consequently, the growing interest in making new and more useful tools to solve the flowshop sequencing problem actually shown in the concerned literature [10-13] is addressed to techniques allowing to improve not workstation utilisation alone, but this latter together with material handling system utilisation [14]. In this context the Genetic Algorithms approach, here developed as a first step with a fitness function based only on the first goal, seems very promising and not yet investigated enough.

2.1 The permutation flow-shop model.

For manufacturing systems characterised by unidirectional part flow and equal routing, the Pure Flow Shop model has been extensively used in developing more and more effective heuristic sequencing algorithms, the only useful for NP-complete combinatorial problems. The PFS scheduling problem can be stated as the problem of processing a set of n parts trough m different workstations, characterised by identical routing for every part to be processed and no job passing or pre-emption allowed; pursuing the objective of minimising the makespan, the well known $n/m/F/C_{max}$ scheduling problem is formulated, subjected to the following restrictive assumptions:

1. all n jobs, because of technological similarities, follow the same flow through the set of the m machines;
2. all jobs and machines are available for processing at production starting;
3. each job requires the whole set of operations, and each operation requires a different machine;
4. set up and removal times are sequence independent and are considered as included in the machining time of each job;
5. no job passing is allowed, that is, job sequences are identical at all stages and thus only permutation schedules are considered;
6. transfer times from machine to machine can be neglected;
7. intermediate storage facilities of infinite capacity are located ahead each workstation.

3 Genetic algorithms and production scheduling.

From a genetic point of view, the n! feasible sequences to be considered for a PFS problem constitutes the search space, each sequence of which being a different chromosome, and each part to be worked constituting a gene g.

The genetic operators selected for these experiments have been worked out also on the basis of a previous author's experience [17]. Particularly, to avoid feasibility problems in creating new individuals and consequent fitness decay due to a needed reconstruction, a decimal representation of the strings has been adopted as well as a preference for genetic operators ensuring directly solution feasibility. Choosing a decimal alphabet to identify each part (the positive integers from 1 to n), each sequence in which the n parts can be worked constitutes a string of length n. In the following, the symbol A(g<i>) indicates that gene g occupies position i in the string A, i.e. part g is processed as i-th in sequence A; moreover, to have a feasible sequence must be A(g<i>) ≠ A(h<j>), i.e. all parts must be represented in the string A. The fitness of each individual is evaluated by the related makespan, and lower makespan values correspond, obviously, to higher sequence fitness.

4 Genetic operators

A Genetic Algorithm, typically, constructs progressively better individuals iteratively applying the set of genetic operators described as follows:

I) *generation*: an initial population of Ns sequences is formed, using random sampling;
II) *reproduction*: a new population, deriving from the initial one, is formed, assigning to the sequences with higher fitness a higher probability to be copied in the new population, also formed by Ns individuals;
III) *crossover*: at each iteration, a pair of genitors are chosen on a fitness basis, that is, assigning a higher level of probability of reproducing to fittest individuals; the new individuals, two for all the selected operators, replace the parents only if they better fit the goal;
IV) *mutation*: a chromosome, chosen casually on a fitness basis, is randomly modified; mutation is applied during the iterative process with a probability Pm;
V) *stop test*: the operators previously described are iteratively applied, starting from the third step, until an ending criterion, based on the number of iterations or on a prefixed difference from a lower bound, is fulfilled.

The subsequent sections present the specific implementation of genetic algorithms to flow-shop scheduling, and the evaluation of their effectiveness.

4.1 Initialisation operator.

In developing a genetic algorithm, the first step is to create some initial individuals. These individuals are used as the parents to which the various genetic operators are applied to generate new individuals. The GA typically starts from a randomly generated population of candidate solutions on which a reproduction operator is applied to improve the fitness of the population before starting the evolutionary process.

4.2 Crossover operators

4.2.1 Order Crossover with Gene Swapping (OCGS). The OCGS operator consists in exchanging between the selected parents the relative order in which genes are positioned, in order to generate the offspring [2]. Several positions are randomly selected and the corresponding genes of a parent are reordered in the offspring in the relative positions corresponding to the ones occupied in the alternate parent. Once built the partial sequence corresponding to the selected parts, the remaining elements are obtained copying directly from parent B the genes that occupy unselected positions, filling the free positions in the sequence following the relative priority. If the selected crossover sites, individualised by the binary cross string as in Table I, are <2>, <4>, <5> and <7> occupied respectively in parent B by genes 2, 8, 5, 1, i.e. if B(2<2>), B(8<4>), B(5<5>) and B(1<7>) are the selected genes, while for the alternate parent A the corresponding genes are A(2<5>), A(8<8>), A(5<4>) and A(1<3>), to respect the relative position occupied in parent A, the genes 2, 8, 5, 1, must fill in offspring D the positions in the relative order 1,5,2,8; that is D(1<2>), D(5<4>), D(2<5>) and D(8<7>). The remaining position are filled as shown in Table I, where offspring C, similarly obtained, is also reported.

Table I - OCGS

Parent A	6, 4, 1, 5, 2, 3, 7, 8
cross sites	0, 1, 0, 1, 1, 0, 1, 0
Parent B	3, 2, 4, 8, 5, 7, 1, 6
Offspring C	6, 2, 1, 4, 5, 3, 7, 8
Offspring D	3, 1, 4, 5, 2, 7, 8, 6

Table II - OCGT

Parent A	6, 4, 1, 5, 2, 3, 7, 8
cross sites	0, 1, 0, 1, 1, 0, 1, 0
Parent B	3, 2, 4, 8, 5, 7, 1, 6
Offspring C	3, 4, 8, 5, 2, 1, 7, 6
Offspring D	6, 2, 4, 8, 5, 3, 1, 7

4.2.2 Order Crossover with Gene Transferring (OCGT) This Order Crossover operator is based on transmitting as a genetic information the part ordering in a sequence, by transferring directly to the new individuals the absolute positions of the parts to be worked as in the parent sequence [2]. The OCGT operator randomly selects several positions in the parent sequences and directly transmits the correspondent genes to the offspring. In the example of Table II the selected elements are A(4<2>), A(5<4>), A(2<5>) and A(7<7>) that are copied in offspring C as C(4<2>), C(5<4>), C(2<5>) and C(7<7>). The remaining positions, starting from the first free one, are filled by the missing genes, respecting the relative order that they have in the alternate parent B. Offspring D is obtained in a similar way, directly transferring the selected elements from parent B and filling the remaining positions accordingly to sequence A ordering.

4.2.3 Partially Matched Crossover (PMX) The partially matched crossover [1] operator allows to transfer important ordering similarities from a pair of parents, randomly selected from the actual population, to the offspring. The matching section of the first parent, defined randomly picking two crossing sites along the string, is directly inherited by the offspring, as with OCGT operator, while the remaining elements are positioned mapping the genes between the crossing sites.

The mapping procedure starts from the first element of the matching block: if the element A<3>=1 matches with gene B<3>=4, and the position of gene 1 is B<7>, the offspring inherit the gene 4 in position 7, i.e. C<7>=4. Because the matching sections of the two parents may contain equal elements, a recombination operator must be used to ensure the legality of the new individual. Because the candidate position for part 8 is C<5>, just occupied by part 2, the gene 8 is temporarily laid aside in the construction of the offspring. The sequence is completed copying from the second parent the remaining genes, creating, if that is the case, some gene repetitions, as C<2>=C<4>=5 in our example, positioned aside the crossing block. The missing genes, the ones previously laid aside, are then used to fill that position, i.e. C(8<2>).

Table III - PMX

Parent A	6, 4, 1, 5, 2, 3, 7, 8
cross sites	0, 0, 1, 1, 1, 1, 0, 0
Parent B	3, 2, 4, 8, 5, 7, 1, 6
Offspring C	7, 8, 1, 5, 2, 3, 4, 6
Offspring D	6, 1, 4, 8, 5, 7, 3, 2

4.3 Mutation operator.

The mutation operator used in the experiments is a gene swapping operator that performs a position based exchange [1]. It randomly selects two sites on the chosen chromosome and swaps the genes occupying these positions. Let A(g<i>) and A(h<j>) be the selected elements; the mutation operator creates a new individual whose chromosome is defined by the swapped genes B(h<i>) and B(g<j>), and by the copied genes B(l<k>)=A(l<k>) $\forall k \neq i,j$.

4.4 Fitness function and control parameters.

The definition of the fitness function and the setting of the control parameters of the GA plays a significant role in effectively solving a scheduling problem. In particular, to accurately evaluate the quality of each individual, the fitness function is here defined as the distance of the makespan of each individual from a lower bound (LB), so obtaining more appropriate levels of competition among individuals.

Moreover, the knowledge of a lower bound allows to test the efficiency of the calculated solutions, and to define a run stop test based on the distance of actual solution from the LB, to be used together with the number of iterations.

4.4.1 Lower bound The lower bound is based on the choice of couples of bottleneck workstations, and on the relaxation of the capacity constraints on the rest of workstations (i.e. considered as of an infinite capacity); each one of the so obtained two workstations fictitious problem is then solved with Johnson's rule to determine the correspondent makespan [18]. The strongest lower bound can evidently be obtained as the maximum among the m(m-1)/2 makespan values obtained coupling the workstations in all the feasible ways.

4.4.2 Fitness function. The knowledge of the previous described lower bound enables to define a fitness function that emphasises the differences in the quality of individuals, allowing to privilege the mating of the best chromosomes during the search. To this purpose, the fitness F of each individual is evaluated as the

Transactions on Information and Communications Technologies vol 8, © 1995 WIT Press, www.witpress.com, ISSN 1743-3517

difference between the related makespan, MK, and the lower bound LB.
4.4.3 Setting of control parameters. The setting levels of the control parameters of the GA, that is, the mutation probability, the population size and the maximum number of iterations Nmi, tested in the experiments, are reported in Table IV. They have been selected in the range of the ones that in [17] revealed themselves as best performing for OCGS algorithm, and have been confirmed by preliminary tests as very effective also for the newly introduced algorithms.

5 Analysis of the genetic algorithms efficiency

In order to analyse the influence of the different crossover operators and control parameters in solving a pure flow shop minimum makespan sequencing problem, considering the high number of factors to be tested, as resumed in Table IV, a statistical validation procedure has been used to assess the more effective GA configuration.

Six different classes of sequencing problems, each one characterised by the number of processed parts (n) and by the number of workstations (m), have been solved with the proposed genetic algorithms. The problems here investigated are characterised by five or ten workstations, heterogeneous mixes composed by 10, 30 or 60 parts, with processing times chosen from a discrete uniform distribution in the range [1, 99].

Table IV - Factors and related levels

Factor	Levels	Values		
m	2	5	10	
n	3	10	30	60
Pm	3	0.09	0.18	0.27
Ns	3	5	10	15
Nmi	3	5000	10000	15000
Cross	3	PMX	OCGT	OCGS

To determine the influence of the factors, type of crossover (Cross), probability of mutation (Pm), population size (Ns), and maximum number of iterations (Nmi), an ANOVA procedure has been performed. As response variable the relative distance of makespan from lower bound ((MK-LB)/LB) has been used to evaluate the effectiveness of the determined solutions. Such metric allows to overcome the variability due to the different mix size and, by standardising the problem, to compare the different results. The Analysis of Variance has been performed on the data resulting from a complete factorial design with six variables and 10 replicates.

Such analysis has shown that all the interactions between factors higher than second degree are not significant. Therefore, only the first and significant second degree interactions have been reported in Table V-i, indicating the level of significance reached with 1 or 2 stars (5%, 1%).

In order to investigate the second degree interaction between crossover and Pm, the average values of the response variable have been reported in Table VI; it can be seen that the high level of interaction is due to the presence of the Pmx crossover operator, which shows a very poor performance in front of the two order based crossover operators. Consequently, Pmx crossover can be excluded in constructing PFS sequencing algorithms.

Table V - Analysis of Variance

	V-i	Analysis with PMX				V-ii	Analysis without PMX			
Source	d.o.f.	SS	F	P		d.o.f.	SS	F	P	
m	1	2.592	4827.55	0.00	**	1	1.511	3106.7	0.00	**
n	2	1.539	1433.21	0.00	**	2	1.001	1029.3	0.00	**
Pm	2	0.011	9.96	0.00	**	2	0.004	3.8	0.023	*
Ns	2	0.007	6.86	0.00	**	2	0.001	0.7	0.49	
Nmi	2	0.017	15.53	0.00	**	2	0.009	9.3	0.00	**
Cross	2	0.065	60.67	0.00	**	1	0.000	0.04	0.85	
m-n	2	0.072	67.34	0.00	**	2	0.044	45.2	0.00	**
m-Pm	2	0.003	3.08	0.05	*	2	0.002	2.6	0.08	
m-Nmi	2	0.005	4.81	0.01	**	2	0.003	3.7	0.025	*
m-Cross	2	0.022	20.27	0.00	**	1	0.000	0.1	0.71	
Pm-Cross	4	0.009	4.00	0.00	**	2	0.000	0.05	0.95	
Ns-Cross	4	0.006	2.74	0.03	*	2	0.000	0.00	1	
Error	4798	2.576				3188	1.551			
Total	4859	6.936				3239	4.134			

Moreover, once excluded PMX from the ANOVA, as in Table V-ii, the number of significant factors substantially decreases and only two genetic operators, i.e. the maximum number of iterations and the mutation probability, result as influent on the schedule performance. As a matter of fact, the number of workstations and the number of processed parts, and their interaction, coming out as significant in the reported analysis, constitute "structural factors", related to the production system and to the mix of parts to be processed.

Table VI - Crossover/Pm interaction

Pm	PMX	OCGT	OCGS	ALL
0.09	0.0468	0.0367	0.0368	0.0401
0.18	0.0435	0.0343	***0.0339***	0.0373
0.27	0.0393	0.0355	0.0354	***0.0367***
ALL	0.0432	0.0355	***0.0354***	0.0380

The dependence of algorithm performances on that "structural factors" is well known and confirmed also analysing the results, related to the same test problems, obtained with two of the best performing traditional search algorithms, i.e. the ones proposed by Nawaz, Enscore & Ham [7] (NEH) and by Passannanti [15] (SARA).

The genetic operators which remain significant are: the iteration limit, that obviously it is better to put to highest tested level, and the mutation probability, that results better at a medium-high level.

In conclusion we suggest that the best choice of factor levels are to be selected among the combination of OCGS or OCGT for crossover and 0.18-0.27 for Pm, combined with the highest Nmi level, as shown in table VII.

The GA results, together with the ones obtained with NEH and SARA algorithms, are resumed in Table VII. The makespan values show that better results can be achieved with the Genetic approach, that allows an average decrease of about 1.2% and 0.3% in front of NEH and SARA respectively. The capability of obtaining such improvements is not negligible, considering that the obtained results are very near, on average, to the optimal ones, as stated by the distance from the related lower bound of minus than 2.5%.

As a matter of fact, the improvement of performance obtained with GA in comparison with SARA algorithm is to be considered highly significant, as stated by the Student t-test for paired result ($t = 2.9$).

Table VII - Results

m	n	Makespan				Pm	Ns	Nmi	Cross
		NEH	SARA	GA	LB				
5	10	750,4	745,1	742,90	719,70	0,18	15	15000	OCGS
5	30	1769,9	1754,2	1752,80	1747,70	0,18	15	15000	OCGT
5	60	3344,50	3328,3	3324,80	3324,50	0,27	15	15000	OCGS
10	10	1052,50	1029,4	1018,90	942,00	0,18	15	15000	OCGT
10	30	2157,60	2132,8	2114,80	2029,10	0,18	15	15000	OCGT
10	60	3665,70	3638,5	3630,30	3527,80	0,18	15	15000	OCGS
		2123,43	2104,72	2097,42	2048.47				

To reduce the computational burden without degrading the solution quality, an ending criterion based on the deviation from the lower bound previously described has been implemented. The evolutionary process is stopped when the makespan value reaches the chosen deviation from the lower bound; when the allowed deviation is not reached during the evolutionary process, the Nmi control parameter acts as subordinate ending test. The related results show that a mean reduction on the number of generated sequences of about 20% is reached using a distance of 1% for the LB based stop test, in front of a worsening of makespan values of 0.23%.

6 Concluding remarks

A number of Genetic Algorithms devoted to the flow shop scheduling problem has been presented in this paper. Through the analysis of the described genetic algorithms has been shown that the different crossover operators and the setting of control parameters significantly affect the solution of a flow shop scheduling problem. In particular, order based crossover operators, that always furnish feasible solutions, have shown themselves to be the most effective ones, as well as medium-high values of the mutation probability and the highest tested number of iterations. The comparison of the obtained performances with the ones achieved with two of the most effective conventional scheduling algorithms reported in literature allows to assert the effectiveness of the genetic approach in solving production scheduling problems.

Acknowledgements
This work was supported by MURST of Italy 40% and 60% grants.

References

Book:

1. Goldberg, D. E., *Genetic Algorithms in Search Optimization and Machine Learning*, Addison-Wesley, 1989.

2. Davis, L., *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, 1990.

Paper in a journal:

3. Campbell, H.G., Dudek, R. A., Smith, M.L., A Heuristic Algorithm for the n Job m Machine Sequencing Problem, *Management Science*, **23**, pp. 630-637, 1970.
4. Gupta, J.N.D., A Functional Heuristic Algorithm for the Flowshop Scheduling Problem, *Opl. Res. Q.*, **22**, pp. 39-47, 1971.
5. Johnson, S.M., Optimal Two and Three-Stage Production Schedules with Setup Times Included, *Nav. Res. Log. Quart.*, **1**, pp. 61-68, 1954.
6. Dannenbring, D.G., An Evaluation of Flowshop Sequencing Heuristics, *Management Science*, **23**, pp. 1174-1182, 1977
7. Nawaz, M., Enscore, E. E. Jr, Ham, I., A Heuristic Algorithm for the m Machine n Job Flow-Shop Sequencing Problem, *OMEGA*, **11**, pp. 91-95, 1983.
8. Biegel, J. E., Davern, J. J., Genetic Algorithms and Job Shop Scheduling, *Comp. Ind. Engng.*, **19**, pp. 81-91, 1990.
9. Corbey M., Measurable Economic Consequences of Investments in Flexible Capacity, *Int. J. of Production Economics*, **23**, pp. 47-57, 1991.
10. Widmer, M., Hertz, A., A new heuristic method for the flow shop sequencing problem, Europ. *J. Oper. Res.*, **41**, pp. 186-193, 1989.
11. Skorin-Kapov, J., Vakharia, A. J., Scheduling a flow-line manufacturing cell: a tabu search approach, *Int. J. Prod. Res.*, **31**, pp 1721-1734, 1993.
12. Osman, I. H., Potts C.N., Simulated Annealing for Permutation Flow-Shop Scheduling, *OMEGA Int. J. of Mgmt. Sci.*, **17**, pp 551-557, 1989.
13. Rajendran, C., Chaudhuri, D., Scheduling in n-job, m-stage flowshop with parallel processors to minimize makespan, *Int. J. of Prod. Economics*, **27**, pp 137-143, 1992.
14. Grasso, V., Noto La Diega, S., Heuristics for the General Flow-Shop Problem: an Approach to Flexible Flow Lines Scheduling, *Manufacturing Systems*, **22**, pp. 197-202, 1993.

Paper in conference proceedings:

15. Passannanti, G., SARA: An Improved Solution for the Flow-Shop Problem, *Proc. 8th Int. Conf. on Computer Aided Production Engineering*, Edinburgh, pp. 370-376, 1992.
16. Glover, F., Kelly, J.P., Overview of Metaheuristics and Recent Advances, *Proc. AIRO '93 Conf.*, pp. XXXIV-XL, 1993.
17. Fichera, S., Grasso V., Genetic Algorithms and Flow Shop Scheduling, I *Conv. AITEM*, pp. 133-140, 1993 (in Italian).
18. Grasso, V., Lo Valvo, E., Segreto, C., A Branch and Bound Flow-Shop Sequencing Approach Based on Heuristic Results, *2nd AMST Int. Conf.*, pp. 330- 336, 1990.