Multi Objective Genetic Algorithm for multimode job shop scheduling problem

Giada La Scalia (giada.lascalia@unipa.it)
Dipartimento di Ingegneria Chimica, Informatica, Gestionale, Meccanica, (DICGIM) Università di Palermo, Facoltà di Ingegneria, Viale delle scienze Ed. 8, 90128 Palermo, Italy

Rosa Micale
Dipartimento di Ingegneria Chimica, Informatica, Gestionale, Meccanica, (DICGIM) Università di Palermo, Facoltà di Ingegneria, Viale delle scienze Ed. 8, 90128 Palermo, Italy

Giuseppe Aiello
Dipartimento di Ingegneria Chimica, Informatica, Gestionale, Meccanica, (DICGIM) Università di Palermo, Facoltà di Ingegneria, Viale delle scienze Ed. 8, 90128 Palermo, Italy

Antonello Giallanza
Dipartimento di Ingegneria Chimica, Informatica, Gestionale, Meccanica, (DICGIM) Università di Palermo, Facoltà di Ingegneria, Viale delle scienze Ed. 8, 90128 Palermo, Italy

Abstract

Multimode Job Shop Scheduling Problem (MJSSP) aims at finding the start times and execution modes for the operations of different jobs that optimize a given set of objective functions while verifying precedence and resource constraints. In this paper, we focus on this problem and develop a Multi Objective Genetic Algorithm (MOGA) to solve it. Its main contributions are the mode assignment procedure in the chromosome generation and the use of three fitness functions. Its performance is demonstrated by computational results obtained on a set of standard instances and against the best currently available algorithms.

Keywords: Multimode Job Shop Scheduling Problem, Multi Objective Genetic Algorithm.

Introduction

Scheduling is one of the most important issues in the planning and operation of manufacturing systems (Chen, Ihlow, & Lehmann, 1999), and it has gained much attention increasingly in recent years (Ho, Tay, Edmund, & Lai, 2007). The classical job-shop scheduling problem (JSP) is one of the most difficult problems in this area. It consists of scheduling the operations of a set of jobs on a set of machines with the objective to optimize one or more criteria. Each machine is continuously available from time zero, processing one operation at a time. Each job has a specified processing order on the machines which are fixed and known in advance. Moreover, processing time is also

fixed and known. Each machine may have the ability of performing more than one type of operations, i.e., for a given operation must be associated with at least one machine. Unlike the classical JSP where each operation is processed on a predefined machine, each operation in the MJSSP can be processed on one out of several machines. This makes MJSSP more difficult to solve due to the consideration of both routing of jobs and scheduling of operations. Moreover, it is a complex combinatorial optimization problem. The problem of scheduling jobs in MJSSP could be decomposed into two sub-problems: the routing sub-problem that assigns each operation to a machine selected out of a set of capable machines, the scheduling sub-problem that consists of sequencing the assigned operations on all machines in order to obtain a feasible schedule to optimize the predefined objective functions. In this paper, we propose an effective MOGA to solve the MJSSP. The rest of the paper is organized as follows. In the first paragraph we present a literature review with regard to classification of multi-project scheduling and associated scheduling models and methodologies. Then we introduce a multi-objective optimization formulation for modelling the scheduling of multimode job shop scheduling, which allows multi-objective optimization with respect to the minimization of the job duration (makespan), the balanced use of the resource over time (utilization coefficient) and the minimization of the cost associated with the job delay (penalty cost). Hence, the design of the multi objective genetic algorithm to solve the problem is described. Finally, the computational result developed to evaluate the performance of the MOGA are presented and the conclusions and the future research are reported.

Literature review

Genetic Algorithm (GA) is an effective meta-heuristic to solve combinatorial optimization problems, and has been successfully adopted to solve the MJSSP. Recently, more and more papers are talking about this topic. They differ from each other in encoding and decoding schemes, initial population method, and offspring generation strategy. Several heuristic procedures such as dispatching rules, local search and meta-heuristics such as tabu search, simulated annealing and GAs have been developed in recent years for MJSSP. They can be classified into two main categories: hierarchical approach and integrated approach. The hierarchical approach attempts to solve the problem by decomposing it into a sequence of sub problems, with reduced difficulty. A typical decomposition is assign-then-sequence, coming from the trivial observation that once the assignment is done, the resulting sequencing problem is a JSP. This approach is followed by Brandimarte 1993, Paulli 1995, Barnes and Chambers 1996, among the others. They all solve the assignment problem using some dispatching rules, and then solve the resulting JSP using different tabu search heuristics. Integrated approach is much more difficult to solve, but in general achieves better results, as reported in Vaessens et al. (1994), Dauzére-Pérés and Paulli (1997) and Hurink et al. (1994). They all adopt an integrated approach, proposing different tabu search to solve the problem. Mastrolilli and Gambardella (1996), show computational results proving that their tabu search performs better than any other heuristic developed so far, both in terms of computation time and solution quality. Recently, GAs have been successfully adopted to solve MJSSP, as proven by the growing number of papers on the topic. Chen et al. (1999) used integrated approach to solve the MJSSP. The genes of the chromosomes respectively describe a concrete allocation of operations to each machine and the sequence of operations on each machine. Yang (2001) proposed a GA-based discrete dynamic programming approach. Zhang and Gen (2005) proposed a multistage operation-based genetic algorithm to deal with the problem from the point view of dynamic programming. Jia et al. (2003) propose a modified GA able to solve distributed scheduling problems and can be adapted for MJSSP. Ho and Tay (2004) propose an efficient methodology called GENACE based on a cultural evolutionary architecture for solving MJSSP with recirculation. Kacem et al. (2002) use a chromosome representation that combines both routing and sequencing information, and develop an approach by localization to find promising initial assignments. Finally, Pezzella et al. (2008) integrate different strategies for generating the initial population, selecting the individuals for reproduction and reproducing new individuals. They are all integrated approaches, and differ

from each other for different coding scheme, initial population generation, chromosome selection and offspring generation strategies.

Problem formulation

The multimode job-shop scheduling problem can be formulated as follows. There is a set of *M* machines $M = \{M_1, M_2, \ldots, M_k, \ldots, M_M\}$ and a set of *N* jobs $J = \{J_1, J_2, \ldots, J_i, \ldots, J_N\}$ in which each job *Ji* is formed by a sequence of operations $O_{ij,}$ where $j= 1,2,\ldots n_i$ and n_i is the number of operations of the job J_i. Each operation has to be performed one after the other according to the given sequence and its processing time is machine-dependent. We denote with d_{ijk} the processing time of operation O_{ij} when executed on machine M_k. Pre-emption is not allowed, i.e., each operation must be completed without interruption once started. Furthermore, machines cannot perform more than one operation at a time. All jobs and machines are available at time 0. The problem is to assign each operation to an appropriate machine (routing problem), and to sequence the operations on the machines (sequencing problem) in order to minimize the objective functions considered. Problem data can be organized in a table, where rows correspond to operations and columns correspond to machines. In the example given in Table 1 there are 2 jobs and 5 machines, where each cell denotes the processing time of that operation on the corresponding machine.

Table 1- Processing time table

Job	**operation**	**M_1**	**M_2**	**M_3**	**M_4**	**M_5**
J_1	o_{11}	3	4	5		
	o_{12}		2	3	3	
	o_{13}	4		4		6
	o_{14}				4	5
J_2	o_{21}	2		2	4	
	o_{22}		5			
	o_{23}				5	6

Establish the model of MJSSP

The advantage of GA with respect to other local search algorithms is due to the fact that more strategies could be adopted together to find good individuals to add to the mating pool in a GA framework, both in the initial population phase and in the dynamic generation phase (Pezzella et al., 2008). In this paper, the proposed GA adopts an improved chromosome representation and a multi objective approach, in order to consider simultaneously three different functions of fitness.

Chromosome representation

Better efficiency of GA-based search could be achieved by modifying the chromosome representation and its related operators so as to generate feasible solutions and avoid repair mechanism. Ho et al. (2007) developed extensive review and investigated insightfully on chromosome representation of MJSSP. Mesghouni et al. (1997) proposed parallel job representation for solving the MJSSP. The chromosome is represented by a matrix where each row is an ordered sequence of each job. Each element of the row contains two terms, the first one is the machine processing the operation, and the second one is the starting time of this operation. The approach requires a repair mechanism and the decoding representation is complex. Chen et al. (1999) divided the chromosome into two parts: A-string and B-string. A-string denotes the routing policy of the problem, and B-string denotes the sequence of the operations on each machine, however this method needs to consider the order of operations and require a repair mechanism. Kacem et al. (2002) represented the chromosome by an assignment table representation. A data structure of the assignment table must necessarily describe the set of all machines. This increases the overall computational complexity due to the presence of redundant assignments. Ho et al. (2007) also

divided the chromosome into two strings, one represents the operation order, the other represents the machine by an array of binary values. This structure can represent the problem clearly and conveniently, but the binary-coded increases the memory space and the operation is not convenient, so when the scale of problem is oversized, the memory space and computational time will increase tremendously (Liu H. et al. 2007). Based on the analysis of the approach from the above literature, we design an improved chromosome representation that requires no repair mechanism. Our chromosome representation, in relation to the example reported in table 1, has two segment, as shown in the figure below.

I segment	II segment
o_{11} o_{23} o_{12} o_{14} o_{21} o_{13} o_{22}	**2 4 3 4 1 3 2**

Figure 1 - Chromosome representation

For the first segment we use a operation priority list representation, in which the solution is encoded as a precedence list of operations randomly generated. All the operations are interpreted according to an interactive eligible list based on the precedence graph in order to always obtain feasible scheduling. The length L of the first segment is equal to the sum of all the operations of the jobs considered. For the instance in Table 1, one possible encoding of the Operation priority list is shown in Fig. 1. To represent the second segment we use an array of integer with the same length of the first segment. Each integer value equals the index of the array of alternative machine set of each operation. For the problem in Table 1, one possible encoding of the machine selection is also shown in Fig. 1. For instance, M_2 is selected to process operation O_{11} since the value in the array of alternative machine set is 2. The value could also equal 1 or 3 since operation O_{11} can be processed on the machines M_1 and M_3.

Multi objective approach

In the proposed approach, three different aspects of MJSS problem are taken into account: makespan, utilization coefficient and job delay. The following objectives are therefore considered:

1. minimization of makespan,
2. maximization of the utilization coefficient,
3. minimization of the job delay.

The first objective function expresses the total length of the schedule to be minimized:

$$F_1 = \min[max_i De_i]$$

Where De_i is the Makespan of the job J_i *(1)*

The second objective has been modelled by the maximization of the utilization coefficient:

$$F_2 = \max \frac{\sum_{i=1}^{N} \sum_{j=1}^{n_i} d_{ijk}}{M \cdot [max_i De_i]} \quad \forall\, k = 1 \ldots . M \qquad (2)$$

Finally, the third aspect taken into account is the minimization of the cost associated with the job delay:

$$F_3 = \min[\sum_{i=1}^{N} max_i(Cp_i; 0)] \qquad (3)$$

where

$$Cp_i = p_i \cdot (De_i - Dp_i) \qquad (4)$$

In equation (4), p_i is the penalty unit cost associated to the delay of the job J_i while C_{pi} is its penalty cost. D_{pi} is the delivery time of the job J_i specified by the customer.

Computational results

The GA procedure described in the precedent section has been implemented and tested on the problem instances from literature (http://www.idsia.ch/monaldo). The best results are selected after ten runs from different initial populations. We have considered the problem instances of Brandimarte, in which the data set consists of 10 problems with number of jobs ranging from 10 to 20, number of machines ranging from 4 to 15 and number of operations for each job ranging from 5 to 15. It must be pointed out that comparing different approaches for benchmarking purposes not only means employing the same input parameters, but also trying to uniform the search space. Provided that perfectly comparable conditions are almost impossible to obtain when two solution approaches differ substantially, the effort has been to reproduce the most similar testing conditions, in order to obtain fairly comparable results. For such reason the reference case and the solution proposed employ a single objective formulation considering the makespan only. Subsequently the solutions referred to the multi-objective context previously described have additionally been determined. The most significant genetic parameters are given in the table below (table 2).

Table 2 - Genetic Parameters

Most significant Genetic Parameters	
Mutation probability	2%
Population size	5.000
Number of generations	1.000
Crossover probability	45%

Table 3 compares our GA to the algorithms proposed by Pezzella et al. 2008, Chen et al. 1999, Ho and Tay 2004 and Jia et al. 2003 on 10 MJSSP problem instances from Brandimarte 1993. The first column reports the instance name; the second and third columns report the number of jobs and the number of machines for each instance, respectively. The fourth column reports our best makespan over ten runs of GA. The remaining columns report the best results of the four algorithms we compare with, together with the relative deviation with respect to our algorithm. The relative deviation is defined as:

$dev = [(MK - MKGA)/MK] \times 100\%$,

where MKGA is the makespan obtained by our algorithm and MK is the makespan of the algorithm we compare to.

Table 3 - Comparison with other GAs on 10 MJSSP instances from Brandimarte

Brandimarte	**n**	**M**	**GA**	**Pezzella**	**Dev(%)**	**Chen**	**Dev(%)**	**Ho**	**Dev(%)**	**Jia**	**Dev(%)**
MK01	10	6	41.0	40.0	2.4	40.0	2.4	41.0	0.0	40.0	2.4
MK02	10	6	27.0	26.0	3.7	29.0	-7.4	29.0	-7.4	28.0	-3.7
MK03	15	8	204.0	204.0	0.0	204.0	0.0	204.0	0.0	204.0	0.0
MK04	15	8	62.0	60.0	3.2	63.0	-1.6	67.0	-8.1	61.0	1.6
MK05	15	4	170.0	173.0	-1.8	181.0	-6.5	176.0	-3.5	176.0	-3.5
MK06	10	15	73.0	64.0	12.3	60.0	17.8	68.0	6.8	62.0	15.1
MK07	20	5	140.0	139.0	0.7	148.0	-5.7	148.0	-5.7	145.0	-3.6
MK08	20	10	524.0	523.0	0.2	523.0	0.2	523.0	0.2	523.0	0.2

MK09	20	10	335	311.0	7.2	308.0	8.1	328.0	2.1	310.0	7.5
MK10	20	15	232	212.0	8.6	212.0	8.6	231.0	0.4	213.0	8.2

Result shows that our algorithm outperforms the other three GAs in some cases.

The first solution given in table 4, refers to the makespan function only, while the second solution reported refers to the results obtained when the other objective functions has been enforced. In both cases the proposed algorithm relates to the MK2 instance and outperforms the reference results, with 7,4% and 3,57 % makespan reduction in the best cases. Computation has been performed employing a general purpose workstation, in less than ten minutes (the determination of a reliable value for the computation time would require specific machine-time analysis which is outside the scope of this paper).

Table 4 - Comparisons of the results.

	Proposed GA	**Proposed GA (multi-objective)**
Makespan	27	28
Utilization coefficient	//	0,97
Job Delay	//	2

The first solution given in figure 2, hence refers to the makespan function only, while the solution reported in figure 3, refers to the results obtained when all the fitness functions have been enforced.

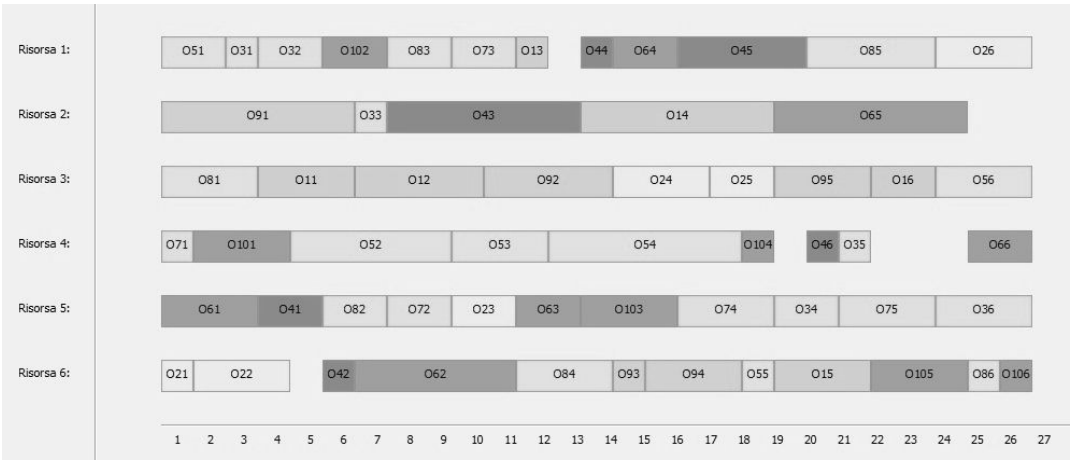

Figure 2 - Gantt chart corresponding to the optimal solutions mono objective conditions.

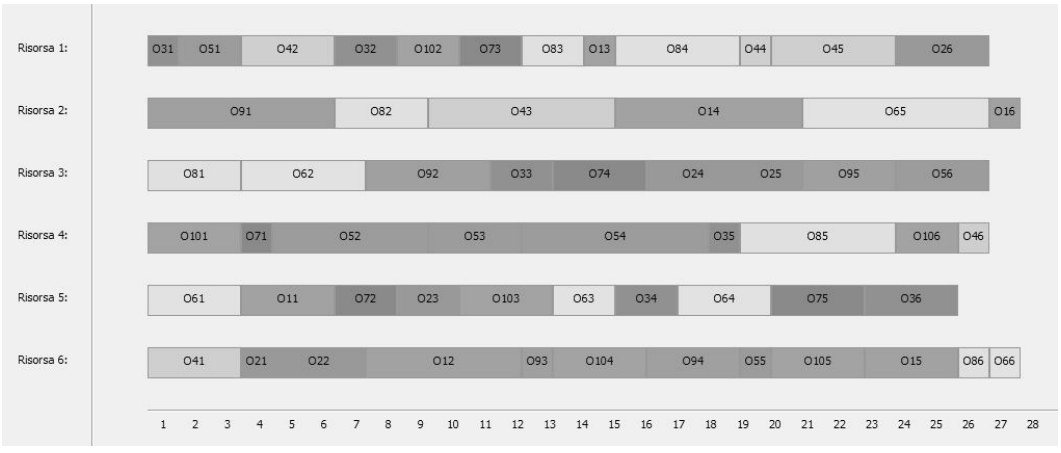

Figure 3 - Gantt chart corresponding to the optimal solutions multi objective conditions.

Adding the objective functions related to the penalty for the delay and the utilization coefficient (Fig. 3), the corresponding scheduling has changed substantially and the makespan, still shows an improvement compared to the reference case. Additionally, the comparison between figure 2 and figure 3 shows that the second scheduling has a better distribution of the workload in terms of minor fragmentation.

Conclusions and future research

In this paper we have developed a multi objective genetic algorithm (MOGA) for the Multimode Job Shop Scheduling Problem (MJSSP). A computational study shows that our algorithm gives results comparable with the best algorithm known so far. As a consequence, the GA framework is effective for developing efficient algorithms for MJSSP, when different assignment procedure in the chromosome generation and the use of more fitness functions, are adopted. The proposed approach is capable of finding a set of Pareto-optimal solutions that optimizes the objective functions simultaneously throughout the entire evolutionary process, giving the decision maker a restricted number of solutions among which he can chose those that he considers the best.

Further improvements of the proposed methodology will include the employment of a structured multi-criteria decision procedure in order to approach the decision process with a more comprehensive procedure, including the aspects related to the intrinsic uncertainty and referring to the typical methodologies of the approximate reasoning, such as the fuzzy theory.

References

Barnes JW, Chambers JB. Flexible Job Shop Scheduling by tabu search. Graduate program in operations research and industrial engineering. Technical Report ORP 9609, University of Texas, Austin; 1996. _http://www.cs.utexas.edu/users/jbc/_.

Brandimarte P. Routing and scheduling in a flexible job shop by tabu search. Annals of Operations Research 1993;41:157–83.

Chen, H., Ihlow, J., & Lehmann, C. (1999). "A genetic algorithm for flexible job-shop scheduling". In *IEEE international conference on robotics and automation*, Detroit 1999 (Vol. 2, pp. 1120–1125).

Dauzére-Pérés S, Paulli J.An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. Annals of Operations Research 1997;70:281–306.

Ho, N. B., Tay, J. C., Edmund, M., & Lai, K. (2007). "An effective architecture for learning and evolving flexible job shop schedules". European Journal of Operational Research, 179, 316–333.

Ho NB, Tay JC. GENACE: an efficient cultural algorithm for solving the Flexible Job-Shop Problem. IEEE international conference on robotics and automation 2004;1759–66.

Hurink J, Jurish B, Thole M. Tabu search for the job shop scheduling problem with multi-purpose machines. OR-Spektrum 1994;15:205–15.

Jia HZ, Nee AYC, Fuh JYH, Zhang YF. A modified genetic algorithm for distributed scheduling problems. International Journal of Intelligent Manufacturing 2003;14:351–62.

Kacem I, Hammadi S, Borne P.Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. IEEE Transactions on Systems, Man, and Cybernetics, Part C 2002;32(1):1–13.

Liu, H., Abraham, A. and Grosan, C., (2007) A novel Variable Neighborhood Particle Swarm Optimization for multi-objective Flexible Job-Shop Scheduling Problems., , ICDIM 138- 145

Mastrolilli M, Gambardella LM. Effective neighbourhood functions for the flexible job shop problem. Journal of Scheduling 1996;3:3–20.

Mesghouni, K., Hammadi, S., & Borne, P. (1997). Evolution Programs For Job-Shop Scheduling. In: Proceedings of the IEEE International Conference On Computational Cybernetics And Simulation (pp. 720-725)

Paulli J. A hierarchical approach for the FMS scheduling problem. European Journal of Operational Research 1995;86(1):32–42.

Pezzella F., Morganti G., Ciaschetti G., A genetic algorithm for the Flexible Job-shop Scheduling Problem, Computers & Operations Research 35 (2008) 3202 – 3212

Vaessens RJM, Aarts EHL, Lenstra JK. Job Shop Scheduling by local search. COSOR Memorandum 94-05. Eindhoven University; 1994.

Yang, J. B. (2001). GA-based discrete dynamic programming approach for scheduling in FMS environments. IEEE Transaction on Systems, Man, and Cybernetics, Part B, 31(5), 824–835.

Zhang, H. P., & Gen, M. (2005). Multistage-based genetic algorithm for flexible jobshop scheduling problem. Journal of Complexity International, 48, 409–425.