Three Genetic Algorithm Approaches to the Unrelated Parallel Machine Scheduling Problem with Limited Human Resources

Fulvio Antonio Cappadonna[1], Antonio Costa[2] and Sergio Fichera[2]
[1]*Dipartimento di Ingegneria Elettrica, Elettronica ed Informatica, University of Catania, Viale A. Doria 6, Catania, Italy*
[2]*Dipartimento di Ingegneria Industriale e Meccanica, University of Catania, Viale A. Doria 6, Catania, Italy*

Keywords: Scheduling, Parallel Machines, Human Resources, Makespan, Genetic Algorithms.

Abstract: This paper addresses the unrelated parallel machine scheduling problem with limited and differently-skilled human resources. Firstly, the formulation of a Mixed Integer Linear Programming (MILP) model for solving the problem is provided. Then, three proper Genetic Algorithms (GAs) are presented, aiming to cope with larger sized issues. Numerical experiments put in evidence how all GAs proposed are able to approach the global optimum given by MILP model for small-sized instances. Moreover, a statistical comparison among proposed meta-heuristics algorithms is performed with reference to larger problems.

1 INTRODUCTION

The parallel machine production system is a very common environment that can be found in many manufacturing situations. In the parallel machine scheduling problem, a set of *n* jobs has to be processed by only one out of *m* machines in parallel; minimizing makespan in such a system is a NP-hard problem as demonstrated by Garey and Johnson (1979). In the more general case of unrelated parallel machines, the processing time of each job depends on the machine it is assigned to, as workstations are supposed to be non-identical. The unrelated parallel machine system has been widely addressed in literature in the past few years; many techniques have been proposed for the resolution of this problem. In Kim et al. (2002) a Simulated Annealing approach is presented for the unrelated parallel machine problem with sequence dependent setup times. Ghirardi and Potts (2005) developed a Recovering Beam search Algorithm for minimizing makespan in an unrelated parallel machine system within a polynomial time, also in case of very large instances. Recently, Vallada and Ruiz (2011) developed a genetic algorithm for solving the makespan minimization problem within an unrelated parallel machine production system with sequence dependent setup times.

Although in the last decades a number of studies have been presented with reference to the unrelated parallel machine issue, the effect of the human factor on this scheduling problem has not been properly addressed yet. Nevertheless, the impact of workforce on the performance of production systems has been widely discussed in the scheduling literature. Norman et al. (2002) considered the problem of assigning workers to manufacturing cells in order to maximize the effectiveness of the organization. In Celano et al. (2008) a first approach for solving the scheduling problem of unrelated parallel manufacturing cells with limited human resource is given, through the development of an integrated simulation framework that studies the effect brought on system performances by the variation of workers employed within the production shop.

In this paper, an unrelated parallel machine problem with limited and differently-skilled human resources is addressed with reference to the makespan minimization objective. To this aim, a Mixed-Integer Linear Programming Model (MILP) and three different Genetic Algorithms (GAs) are proposed.

The remainder of the paper is organized as follows. In Section 2 the problem statement is reported. In Section 3 the description of a MILP model able to optimally solve small instances of the aforementioned problem is given. Section 4 illustrates the genetic algorithms developed. In Section 5 obtained results are discussed. Finally, Section 6 concludes the paper.

Cappadonna F., Costa A. and Fichera S..
Three Genetic Algorithm Approaches to the Unrelated Parallel Machine Scheduling Problem with Limited Human Resources.
DOI: 10.5220/0004116501700175
In *Proceedings of the 4th International Joint Conference on Computational Intelligence* (ECTA-2012), pages 170-175
ISBN: 978-989-8565-33-4

2 PROBLEM STATEMENT

The proposed unrelated parallel machine problem can be stated as follows. Let us consider a set N of n jobs that has to be worked in a production stage made of a set M of m parallel machines, aiming at the minimization of the total completion time, i.e., makespan. Each job has to be processed by only one machine before the exit from the system is allowed. Setup operations performed on a given workstation by a single worker must precede each job processing on the same workstation. A team W of w workers is assumed to be committed to these operations being, in general, $w \leq m$; this means that operators represent a critical resource. In addition, each worker is featured by a certain skill level, on the basis of which he is able to perform setup operations slower or faster than his colleagues.

After setup operation, the job remains on a given machine until its own processing has been completed, as pre-emption is not allowed. Setup times are assumed to be sequence-independent; nevertheless, being the machines unrelated and the workers differently-skilled, setup times depend both on the worker selected for performing setup operations and the machine actually processing the job. In addition, processing times depend also on the machines jobs are assigned to.

3 MILP MODEL

A first goal of the proposed research has consisted in the development of a Mixed Integer Linear Programming (MILP) model, aiming to both optimally solve a set of small instances of the problem in hand and validate performances of the provided GAs. In the following, mathematical formulation is reported.

Indices

$j,l = 1,2,\ldots,n$	jobs
$i = 1,2,\ldots,m$	machines
$k = 1,2,\ldots,w$	workers

Parameters

T_{ij}	processing time of job j on machine i
S_{ikj}	setup time of job j performed by worker k on machine i
M	a big number

Binary variables

X_{ikj}	1 if job j is processed on machine i with setup performed by worker k; 0 otherwise
Q_{jl}	auxiliary variable for *either-or* constraint

Continuous variables

CS_j	setup completion time of job j
C_j	completion time of job j
$C_{\max}$	makespan

Model

$$\text{minimize } C_{\max}$$

subject to:

$$\sum_{i=1}^{m}\sum_{k=1}^{w} X_{ikj} = 1 \qquad \forall j \qquad (1)$$

$$CS_j \geq \sum_{i=1}^{m}\sum_{k=1}^{w} X_{ikj} \cdot S_{ikj} \qquad \forall j \qquad (2)$$

$$C_j - CS_j \geq \sum_{i=1}^{m}\sum_{k=1}^{w} T_{ij} \cdot X_{ikj} \qquad \forall j \qquad (3)$$

$$\begin{cases} CS_j - C_l \geq \sum_{k=1}^{w} X_{ikj} \cdot S_{ikj} - M \cdot (2 - \sum_{k=1}^{w} (X_{ikj} + X_{ikl}) + Q_{jl}) \\ CS_l - C_j \geq \sum_{k=1}^{w} X_{ikl} \cdot S_{ikl} - M \cdot (2 - \sum_{k=1}^{w} (X_{ikj} + X_{ikl}) + 1 - Q_{jl}) \end{cases} \qquad (4)$$

$$\forall i, j, l;\ j < l$$

$$\begin{cases} CS_j - CS_l \geq \sum_{i=1}^{m} X_{ikj} \cdot S_{ikj} - M \cdot (2 - \sum_{i=1}^{m} (X_{ikj} + X_{ikl}) + Q_{jl}) \\ CS_l - CS_j \geq \sum_{i=1}^{m} X_{ikl} \cdot S_{ikl} - M \cdot (2 - \sum_{i=1}^{m} (X_{ikj} + X_{ikl}) + 1 - Q_{jl}) \end{cases} \qquad (5)$$

$$\forall k, j, l;\ j < l$$

$$C_{\max} \geq C_j \qquad \forall j \qquad (6)$$

$$X_{ikj} \in \{0;1\} \qquad \forall i,k,j \qquad (7)$$

$$Q_{jl} \in \{0;1\} \qquad \forall j,l \qquad (8)$$

Constraint (1) ensures that each job is assigned to one and only one machine, and that its setup is performed by one and only one worker. Constraint (2) forces the setup completion time of each job to be equal or greater than the actual setup time of the job itself. Constraint (3) states that, after setup completion, the processing time must elapse before the job can be considered definitively completed. Through the couple of constraints (4) is imposed that, if two jobs are assigned to the same machine,

no overlap between that is allowed (i.e., one job must be completed before setup of the other one is started, or vice versa). Similarly, constraints (5) avoids overlapping of setup operations performed by the same worker (i.e., setup of one job must be completed before setup of the other one is started, or vice versa). Constraint (6) forces the makespan to be equal or greater than the completion time of each job. Finally, through constraints (8) and (9), the binary variables are defined.

4 PROPOSED GENETIC ALGORITHMS

The MILP model proposed is able to optimally solve small instances of the problem at issue. Nevertheless, it cannot be employed for larger examples, because of the high computational burden required. In order to solve a set of large-sized instances of the aforementioned problem, three different meta-heuristic procedures based on genetic algorithms (GAs) have thus been developed. In the following subsections, a detailed description of proposed GAs is reported.

4.1 Permutation-based GA

A first approach towards the resolution of the aforementioned problem by means of GAs consisted in the development of a genetic algorithm equipped with a permutation-based encoding scheme, hereinafter PGA. In such procedure, each chromosome directly describes the order in which jobs have to be processed in the manufacturing stage, while the assignment of jobs to machines and workers is performed by the decoding procedure, on the basis of a time-saving rule. Below, a detailed description of such algorithm is provided.

4.1.1 Encoding/Decoding Scheme

In PGA, each solution is represented by a permutation π of n elements, where n is the number of jobs to be scheduled in the manufacturing stage. More in detail, let $\pi(l)$ be the job on the l-th position of the considered permutation (l=1,2,…,n) to be scheduled on an unrelated parallel machine production system with m machines and w workers ($w \leq m$); $S_{ik\pi(l)}$ denotes the time required by worker k (k=1,2,…,w) to perform setup of job $\pi(l)$ on machine i (i=1,2,…m), while $T_{i\pi(l)}$ indicates processing time of job $\pi(l)$ on machine i. The decoding procedure considers jobs in the order they appear in the permutation and assigns them to the couple machine-worker that can complete them earlier than any other. Thus, indicating with TM_i the time at wich machine i is ready to accept a new job, and with TW_k the time at which worker k is ready to start a new setup operation after all jobs preceding $\pi(l)$ in the permutation have been scheduled, the completion time $C_{\pi(l)}$ is calculated as follows:

$$C_{\pi(l)} = \min_{i,k} \left\{ E_{ik\pi(l)} \right\} \quad (9)$$

where $E_{ik\pi(l)}$ indicates the estimated completion time of job $\pi(l)$ if processed on machine i with setup performed by worker k, calculated according to the following formula:

$$E_{ik\pi(l)} = \max \left\{ TM_i ; TW_k \right\} + S_{ik\pi(l)} + T_{i\pi(l)} \quad (10)$$

Then, denoting with i^* and k^* respectively, the machine and the worker to which job $\pi(l)$ is assigned (i.e., those minimizing $E_{ik\pi(l)}$), quantities TM_{i^*} and TW_{k^*} are updated as follows:

$$TM_{i^*} = C_{\pi(l)} \quad (11)$$

$$TW_{k^*} = C_{\pi(l)} - T_{i^*\pi(l)} \quad (12)$$

Lastly, after the aforementioned procedure has been performed for all jobs in the permutation, the makespan is calculated according to the following formula:

$$C_{\max} = \max_{l} \left\{ C_{\pi(l)} \right\} \quad (13)$$

4.1.2 Selection, Crossover and Mutation Operators

With regards to selection mechanism, the well-known roulette-wheel scheme (Michalewicz, 1994) has been adopted, assigning to each solution a probability of being selected inversely proportional to makespan value. A position-based crossover (Syswerda, 1991) has been employed for generating new offspring from a couple of selected parents. With reference to mutation procedure, a simple swap operator (Oliver, 1987) has been chosen. The algorithm has also been equipped with an elitist procedure which copies the best two individuals of each generation into the new population. Finally, a total number of makespan evaluations has been set as stopping criterion for the algorithm.

4.2 Multi-encoding GA

The proposed PGA allows considerably limited

computational burdens, because of the very simple encoding exploited. Nevertheless such algorithm moves over a space which cannot embrace the entirety of solutions, as the decoding procedure may define, for each permutation, one and one only scheme regarding assignment of jobs to machines and workers. A second approach towards the resolution of the proposed problem by means of GAs, consisted thus in the development of a genetic algorithm, hereinafter MGA, equipped with a multi-encoding scheme able to describe a wider solution space compared to PGA. In such procedure, two arrays for driving the assignment of jobs to machines and workers, respectively, are added to job permutation in the chromosome structure. Below, a detailed description of such algorithm is provided.

4.2.1 Encoding/Decoding Scheme

In order to illustrate the encoding procedure exploited by MGA, let us use the same nomenclature defined for PGA. Thus, assuming to have n jobs to be scheduled on an unrelated parallel machine system with m workstations and w workers ($w \leq m$), each chromosome is represented by the following substrings:

- a permutation π' of n elements;
- an array π'' of n integers ranging from 0 to m, driving the assignment of jobs to machines;
- an array π''' of n integers ranging from 0 to w, driving the assignment of jobs to workers.

In order to introduce the decoding procedure, let $\pi'(l)$ be the job on the l-th position of the permutation π' (l=1,2,…,n); $\tilde{i}$ indicates the element at position $\pi'(l)$ of array π''; $\tilde{k}$ indicates the element at position $\pi'(l)$ of array π'''. $S_{ik\pi'(l)}$ denotes the time required by worker k (k=1,2,…,w) to perform setup of job $\pi'(l)$ on machine i (i=1,2,…m) while $T_{i\pi'(l)}$ indicates processing time of job $\pi'(l)$ on machine i. TM_i and TW_k denote times at which machine i and worker k, respectively, are ready to start a new setup operation after all jobs preceding $\pi'(l)$ in the permutation have been scheduled.

The decoding procedure considers jobs in the order they appear in permutation π' and uses information from arrays π'' and π''' to perform the assignment of jobs to machines and workers; if no information is given by one or both arrays (i.e. if $\tilde{i}$ =0 and/or $\tilde{k}$ =0), the same time-saving rule of PGA is used. Hence, completion time $C_{\pi'(l)}$ is calculated as follows:

$$C_{\pi'(l)}= \begin{cases} E_{\tilde{i}\tilde{k}\pi'(l)} & \text{if } \tilde{i} \neq 0 \text{ and } \tilde{k} \neq 0 \\ \min\limits_{k} \left\{ E_{\tilde{i}k\pi'(l)} \right\} & \text{if } \tilde{i} \neq 0 \text{ and } \tilde{k} = 0 \\ \min\limits_{i} \left\{ E_{i\tilde{k}\pi'(l)} \right\} & \text{if } \tilde{i} = 0 \text{ and } \tilde{k} \neq 0 \\ \min\limits_{i,k} \left\{ E_{ik\pi'(l)} \right\} & \text{if } \tilde{i} = 0 \text{ and } \tilde{k} = 0 \end{cases} \quad (14)$$

where $E_{ik\pi'(l)}$ indicates the estimated completion time of job $\pi'(l)$ if processed on machine i with setup performed by worker k, calculated according to the following formula:

$$E_{ik\pi'(l)} = \max\left\{TM_i; TW_k\right\} + S_{ik\pi'(l)} + T_{i\pi'(l)} \quad (15)$$

According to such decoding procedure, the job is assigned to machine $\tilde{i}$ if $\tilde{i} \neq 0$, and to worker $\tilde{k}$ if $\tilde{k} \neq 0$; if not obtainable from arrays π'' and π''', machine and worker for processing job $\pi'(l)$ are chosen as those minimizing the estimated completion time according to formula (14). Thus, denoting with i^* and k^* respectively, the machine and the worker to which job $\pi'(l)$ is assigned, quantities TM_{i^*} and TW_{k^*} are updated as follows:

$$TM_{i^*} = C_{\pi'(l)} \quad (16)$$

$$TW_{k^*} = C_{\pi'(l)} - T_{i^*\pi'(l)} \quad (17)$$

Lastly, after the aforementioned procedure has been performed for all jobs, the makespan is calculated according to the following formula:

$$C_{\max} = \max_{l} \left\{ C_{\pi'(l)} \right\} \quad (18)$$

4.2.2 Selection, Crossover and Mutation Operators

The same roulette-wheel mechanism exploited in PGA has been adopted for selecting chromosomes, assigning to each solution a probability of being selected inversely proportional to makespan value. Crossover procedure has been performed by separately managing the mating between the three parts of the parent structures (i.e., permutation substrings, machine assignment arrays, worker assignment arrays), with three distinct probabilities p_{cross}', p_{cross}'', p_{cross}'''. Crossover between permutation substrings of two parents has been executed through a position-based operator as in PGA. With reference to arrays driving the assignment of jobs to machines and workers, a simple uniform crossover operator (Syswerda, 1989) has been employed. Similarly to crossover, mutation procedure has been performed by separately managing the three parts of the chromosome, using

three distinct probabilities p_{mut}', p_{mut}'', p_{mut}'''. Mutation of the permutational substring of chromosomes has been performed through the same swap operator exploited in PGA. With reference to assignment arrays, a simple uniform mutation operator (Michalewicz, 1994) has been adopted. The elitist procedure employed in PGA has been used as well. Lastly, the same criterion of PGA has been chosen for stopping the algorithm, i.e. the total number of makespan evaluations.

4.3 Hybrid GA

The last approach for solving the proposed problem through the employment of proper GAs consisted in the development of a hybrid genetic algorithm, hereinafter HGA, combining both the aforementioned meta-heuristics. In such technique, a first optimization phase is performed by PGA; then, after a proper encoding conversion procedure is executed, MGA is launched to complete the second part of the algorithm. Through this method, the space of solutions is quickly probed into as first, by means of the "smart encoding" adopted by PGA; then, a refined research is executed by MGA, equipped with a more accurate encoding scheme. The encoding conversion procedure occurs when a fixed percentage of the total number of makespan evaluations has been reached by PGA. It operates by adding two assignment arrays to all chromosomes of the last population obtained.

5 NUMERICAL EXAMPLES AND COMPUTATIONAL RESULTS

In order to assess the performances of proposed GAs in solving the unrelated parallel machine problem with limited and differently-skilled human resources, a comparison between the proposed meta-heuristics and the MILP model developed has been performed on a benchmark of small-sized test cases. A total of 8 classes of problems have been generated by combining the following factors:

- number of jobs (n): 2 levels (8, 10);
- number of machines (m): 2 levels (4, 5);
- number of workers (w): 2 levels (2, 3).

For each class, 10 instances have been generated letting vary, with uniform distribution, processing times in the range [1, 99] and setup times in the range [1, 49]. Thus, a total of 80 problems has been created. For each problem, the global optimum has been found through the resolution of the MILP model executed on a IBM ILOG CPLEX® Vers.12.2 (64 bit) platform. Then, the whole set of instances has been solved by the proposed GAs, with all parameters tuned after a proper calibration phase and termination criterion set at 10,000 makespan evaluations. The Relative Percentage Deviation (*RPD*) from the global optimum has been computed for each problem, according to the following expression:

$$RPD = 100 \cdot \frac{\mathrm{GA}_{sol} - \mathrm{BEST}_{sol}}{\mathrm{BEST}_{sol}} \tag{19}$$

where BEST_{sol} is the global optimum obtained through the resolution of the mathematical programming model, and GA_{sol} is the best solution provided by a given genetic algorithm after the stopping criterion is reached. Table 1 shows average RPDs obtained, grouping results by number n of jobs. Results show how all proposed GAs are able to closely approach the global optimum with a limited computational burden, as the amount of time required by all meta-heuristics for solving a given problem is, on, average, lower than 4 seconds.

Table 1: Average performances of GAs on small test cases.

Number of jobs (n)	Average *RPD*		
	PGA	MGA	HGA
8	4.368	2.532	3.676
10	3.883	3.416	3.204
Average	4.126	2.974	3.440

After having validated the performances of proposed GAs, a wider set of large-size instances has been created in order to carry out a comparison among the three methods proposed. To this end, 36 new classes of problems have been generated by combining the following factors:

- number of jobs (n): 4 levels (20, 40, 60, 100);
- number of machines (m): 3 levels (10, 15, 20);
- number of workers (w): 3 levels (5, 8, 10).

For each class, 10 problems have been generated letting processing time vary in the range [1, 99] and setup times in the range [1, 49]. Thus, a total of 360 problems has been created. All problems have been solved five times by each GA. The performance index chosen was the same RPD reported in equation (19), considering as BEST_{sol} the best solution obtained by GAs for a given problem; results obtained are reported in Table 2.

In order to infer some conclusion over the statistical significance of differences between

performances of meta-heuristics proposed, an analysis of variance (ANOVA) (Mongomery, 2007) has been performed. Figure 1 shows the LSD intervals (α=0.05) regarding RPDs obtained by each algorithm. It can be seen how HGA outperforms the other meta-heuristics on large size instances, with a statistically significant difference.

Table 2: Average performances of GAs on large test cases.

Number of jobs (n)	Average *RPD*		
	PGA	MGA	HGA
20	2.015	2.395	1.980
40	3.952	3.914	3.534
60	3.141	4.000	2.534
100	2.809	2.372	2.217
Average	2.979	3.170	2.566

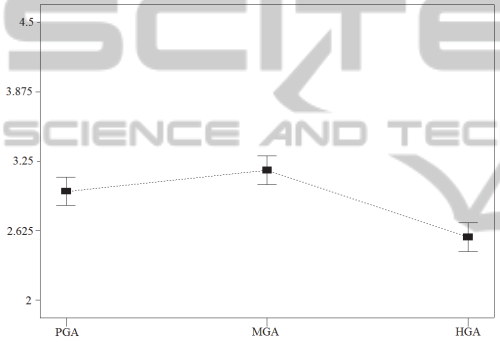

Figure 1: LSD plot for proposed GAs.

6 CONCLUSIONS

In this paper, the unrelated parallel machines scheduling problem with limited and differently-skilled human resources has been addressed with regards to the makespan minimization objective. As first, a MILP model has been developed, in order to assess the performances of three Genetic Algorithms (GAs) properly developed for the problem at issue. Then, these latter procedures have been tested on a wider set of large-sized instances, in order to carry out a comparison among them. Results obtained show how an hybrid approach, which combines two GAs exploiting different encodings, outperforms the single-encoding algorithms from which it is derived. Statistical analysis confirms the significance of difference between performances obtained, thus giving evidence of the effectiveness of the proposed hybrid procedure.

Further research should involve the consideration of sequence-dependent setup times of jobs to be scheduled in the manufacturing system.

REFERENCES

Celano, G., Costa, A., Fichera, S., 2008. Scheduling of unrelated parallel manufacturing cells with limited human resources. *International Journal of Production Research* 46(2): 405-427.

Garey, M. R., Johnson, D. S., 1979. *Computers and intractability: a guide to the theory of NP completeness*. Freeman, San francisco.

Ghirardi, M., Potts, C. N., 2005. Makespan minimization for scheduling unrelated parallel machines: a recovering beam search approach. *European Journal of Operational Research* 165(2): 457-467.

Kim, D. W., Kim, K. H., Jang, W., Frank Chen, F., 2002. Unrelated parallel machines scheduling with setup times using simulated annealing. *Robotics and Computer Integrated Manufacturing* 18: 223-231.

Michalewicz, Z., 1994. *Genetic algorithms + data structures = evolution programs*. Springer-Verlag, Berlin, 2nd edition

Montgomery, D., 2007. *Design and analysis of experiments*. John-Wiley & Sons, New York. 5th edition.

Norman, B. A., Tharmmaphornphilas, W., LaScola Needy, K., Bidanda, B., Colosimo Warner, N., 2002. Worker assignment in cellular manufacturing considering technical and human skills. *International Journal of Production Research* 40(6): 1479-1492.

Oliver, I. M., Smith, D. J., Holland, J. R. C., 1987. A study of permutation crossover operators on the TSP. In: Grefenstette J. J., *Genetic algorithms and their applications: proceedings of the second international conference.* Lawrence Erlbaum, Hillsdale.

Syswerda, G., 1989. Uniform crossover in genetic algorithms. In: Schaffer J.D., *Proceedings of the third international conference on genetic algorithms*. Morgan Kaufmann Publishers, San Mateo.

Syswerda, G., 1991. Schedule optimization using genetic algorithms. In: Davis L., *Handbook of genetic algorithms*. Van Nostrand Reinhold, New York.

Vallada, E., Ruiz, R., 2011 A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. *European Journal of Operational Research* 211: 612-622.