J. Basic. Appl. Sci. Res., 1(10)1411-1414, 2011

ISSN 2090-424X
Journal of Basic and Applied Scientific Research
www.textroad.com

Hybrid Genetic Algorithm for Minimizing the Maximum Lateness on Uniform Parallel Machines

Mohammad Akhshabi [1]*, Mostafa Akhshabi [2], Fraydoon Rahnamay Roodposhti [3]

[1]Department of Industrial Engineering, Islamic Azad University Tonekabon Branch , Tonekabon, Iran
[2]Young Researchers Club, Neyshabur Branch, Islamic Azad University, Neyshabur,Iran
[3]Department of Management, Science and Research Branch, Islamic Azad University ,Tehran, Iran

ABSTRACT

This paper considers the uniform parallel machine scheduling problem which is to minimize the maximum lateness. This problem is equivalent to the uniform parallel machine scheduling problem, which is to minimize the maximal completion time of n jobs whose release times are zero, processing times depend on the speed of the machine to which they are assigned, and their delivery times are different. This problem is NP-hard, even if the machines' speeds are identical and all the delivery times equal to zero. We propose a hybrid genetic algorithm, named GA-VNS. Based on a large set of instances, an experimental study has been carried out to evaluate our methods.

KEY WORDS: Uniform parallel machine, hybrid genetic algorithm, maximum lateness.

INTRODUCTION

We consider the uniform parallel machine scheduling problem to minimize the maximal lateness. It is described generally as Qm||Lmax according to the standard three parameter notation proposed by Graham et al. [1]. Each scheduling problem is denoted by the standard three-field notation α|β|γ . The first field α describes the scheduling type, the second field β is reserved for the information and conditions of scheduling, while the third field γ contains the performance criteria. This problem is NP-hard, even if its simpler P2||Cmax case, in which there are only 2 machines and all of the jobs have the same due date [2]. This problem is equivalent to the uniform parallel machine problem with delivery times to minimize the maximal completion time, i.e. Qm|qj|Cmax problem. In many practical scheduling problems, the delivery times must elapse after the jobs are processed on the machine. The delivery times may be brought by the transportation times or the need of some special products, such as the course of cooling for steel and iron products or the course of drying for painted products. The uniform parallel machine scheduling problems are fundamental to numerous complex real-world applications.

Although much research has been devoted to the parallel machine scheduling problems with identical machine speeds, little research has been done on uniform parallel machines. Koulamas and Kyparisis [3] showed that an extension of the EDD (Earliest Due Date first) rule to the Qm||Lmax problem yields a maximum lateness value does not exceed the optimal value by more than pmax, where pmax is the maximum job processing time. They also showed that the LDT (Largest Delivery Time first) heuristic is a $(m-1)s_1/\sum_{i=1}^{m} s_i + 1$-approximation algorithm for Qm|qj|Cmax problem, where m is the number of the machines, si is the speed of the i-th machine, and s1 is the maximum speed. Dessouky [4] considered Qm|rj, pj = 1|Lmax problem, in which the job processing times are identical. He proposed six simple heuristics, and then developed a branch-and- bound procedure which could solve the problems within 5 machines and 80 jobs within a reasonable time.

Some other papers considered the corresponding identical parallel machine scheduling problem in which the machines have the same speed. The literature in recent years mainly focused on the problem with unequal release dates, i.e. Pm|rj|Lmax problem. Carlier [5] and Néron et al. [6] developed some exact branch-and-bound algorithms for the problem. Vakhania [7] and Gharbi and Haouari [8] considered the development of heuristics. Mastrolilli [9] and Carlier and Pinson [10] proposed some approximation algorithms. Eren [11] considered the m-identical parallel machine scheduling problem with a learning effect to minimize the maximum lateness. He proposed a model, which can optimally solve the problems with 18 jobs and 4 machines within 7000 s on a personal computer with Pentium IV/2 512 Ram. For the single machine with minimizing the maximum lateness, recent literature mainly considered some extensive problems. For example, Wu et al. [12] considered the single-machine maximum lateness minimization problem with a learning effect. The simulated annealing algorithm they proposed

***Corresponding Author:** Mohammad Akhshabi, Department of Industrial Engineering, Islamic Azad University Tonekabon Branch , Tonekabon, Iran, Email: m.akhshabi@toniau.ac.ir

outperforms the traditional heuristic algorithm in terms of quality and execution time for a large number of jobs. It can solve the problems with 200 jobs within 21.24 s on a Pentium IV personal computer. Uzsoy and Velasquez [13] addressed the problem of scheduling a single machine subject to family dependent set-up times in order to minimize maximum lateness. The incomplete dynamic programming heuristic they developed can solve the problems with 50 jobs within 334.21 s on a Pentium II, 266 MHz notebook with 64 MB of RAM.
Considering the NP-hardness of our scheduling problem, we introduce a hybrid genetic algorithm approach to generate the near-optimal solutions with high quality.Only a few papers have focused on parallel machine scheduling problems.

The remainder of this paper is organized as follows. In the next section, we describe Qm||Lmax problem and Qm|qj|Cmax problem. In the third section, we consider Variable-neighbourhood search (VNS) approach to solve our problem. Then, a hybrid algorithms (GA-VNS) is described in sections 4. The results are presented in Section 5. This paper ends with conclusions in Section 6.

Problem description

The problem under consideration is the problem of scheduling uniform parallel machines so as to minimize the maximal lateness. We are given a set of n jobs, j_1, …. , j_n, each of them has to be scheduled without interruption on one of m machines, M_1,…..,M_m. Machine M_i (i = 1, . . . ,m) has a speed s_i ($s_i > 0$). Without loss of generality, we assume that $s_1 \geq s_2 \geq \ldots\ldots\ldots \geq s_m$. A machine can process at most one job at a time, and a job can run on only one machine at a time. All jobs and machines are available at time 0. If a job J_j is processed on a machine Mi, it will take a positive processing time p_{ij} and $p_{ij} = p_j/s_i$, here p_j is the length of job J_j. Each job has a distinct due date d_j for j = 1,… ,n. The objective is to determine a schedule so that the maximum lateness L_{max} = max L_j $1 \leq j \leq n$ is minimized, where $L_j = c_j - d_j$ is the lateness and c_j is the completion time of job J_j. Because the objective function L_{max} is not always positive in Qm||Lmax problems, the equivalent form Qm|qj|Cmax is frequently considered in the literature. Let $q_j = d_{max} - d_j$for all j and d_{max} = max d_j , $1 \leq j \leq n$ is the maximum due date [14].

In this paper, we mainly focus on the Qm|qj|Cmax problem too. Here each job J_j has a distinct positive delivery time q_j that must elapse between its completion on the machine and its exit from the system. Let $c_j$denote the completion time of job J_j on a machine, then for the consumers the effective completion time $C_j = c_j + q_j$. The objective is to minimize the largest Completion time c_{max}= max c_j = max ($c_j + q_j$). Note that the makespan corresponds to c_{max}= max c_j, and it is different from the maximum completion time here.denote that the job J_j is processed on the machine M_i.c_{max} ($J_{j'} \rightarrow M_{i'}$, $J_j \rightarrow M_i$) is the larger completion time between job $J_{j'}$ and $J_{j'}$, when they are all scheduled on machine M_i, and the job $J_{j'}$ is processed before job J_j. ‘‘c_{max} ($J_{j'} \rightarrow M_{i'}$, $J_j \rightarrow M_i$) ’’ denotes the larger completion time between job $J_{j'}$ and J_j, when they are proposed on the machine $M_{i'}$ and $M_i$respectively. The sequence of the jobs on the same machine forms a sub-schedule.

Variable neighborhood search

Variable-neighbourhood search (VNS) is a novel meta-heuristic approach for combinatorial optimization [15]. VNS is different from most local search heuristics. Indeed, it uses two or more neighbourhoods, instead of one, in its structure. For this reason, escaping from a local optimum can be done by changing the neighbourhood structure. Despite its recent development, VNS has been successfully applied to a wide variety of optimization problems such as vehicle routing problems, project scheduling, automatic discovery of theorems, graph colouring and the synthesis of radar polyphase codes [15]. In order to reduce the computational cost, the best number of neighbourhoods is usually fixed to 2 [16]. In our algorithm, we prefer to follow this experimental adjustment. Thus, the two neighbourhoods used in our algorithm are defined as follows:
1. Insertion moves (k = 1): it identifies randomly one job and assigns it to an arbitrary machine. We must respect the WSPT rule to fix this job in the appropriate position.
2. Swap moves (k = 2): it identifies randomly two jobs from two different machines and places each one in the WSPT order after swapping their assignments.

The hybrid genetic algorithm (hGA)

The overall structure of our GA can be described as follows:
1. Coding: The genes of the chromosomes describe the assignments of jobs to the machines. Each chromosome represents a solution to the problem.
2. Initial population: The initial chromosomes are obtained by random assignments.

3. Fitness evaluation: The total weighted completion time is computed for each chromosome in the current generation.
4. Selection: The best chromosomes are chosen for reproduction by a linear ranking.
5. Offspring generation: Survivors are randomly chosen, with a replacement procedure, to produce offspring until the size of the population returns to its original level. Each mating produces a single offspring, as with sexual reproduction, an offspring is created from the genes of the two parents. Each allele is randomly chosen from one parent or the other. Mating is accomplished using uniform crossover with a p_{cr} crossover probability. In addition, a mutation probability p_m is applied to each allele. This means that after crossover the GA allows, with a pm probability, for each job being reassigned to a randomly chosen machine.
6. Acceptance policy: In order to enhance the solution obtained by the genetic algorithm after the reproduction phase (crossover and mutation) we compare the total weighted completion time of the new offspring $F(X_{\text{new of fspring}})$ to $(1+\alpha)\, F_{best}$ where F_{best} is the best result and α is a parameter fixed by the experimentation. If the $F(X_{\text{new of fspring}}) < (1+\alpha)\, F_{best}$ then we use the VNS algorithm.
7. Stop criterion: We stop when the fixed number of generations is reached. If the stop criterion is satisfied, the algorithm stops and the best chromosome, together with its corresponding schedule, are given as output. Otherwise, the algorithm repeats again steps 3-6.

Computational experiments

To evaluate the proposed meta-heuristics, we generated instances with various sizes that are determined by n and m, where n = 20, 50, 100, 200 and m =2, 5, 10, 20. For each combination of n and m, 10 instances are randomly generated. These 10 problems have the same structure and are tested as a group in order to produce an instance for a given combination (n, m). First, the speeds s_i are uniformly generated in [1, 10]. Secondly, processing times pi are generated from the uniform distribution [1, 100]. Weights wi are randomly generated in [1, 10]. In total, $4 \times 4 \times 10 = 160$ instances are created. Due to the main influence of the selected values of parameters on the obtained results, we carried out preliminary extensive computational tests. Thus, we used the following tuning of parameters: For GA, the population size is fixed at 100, pcr = pm = 1. The stopping criterion is Tmax = n/10 for all the algorithms. Finally, α = 3% is used for the hybrid approaches. The proposed meta-heuristics were implemented in the C++ language and run on a on a laptop with Pentium IV Core 2 Duo 2.53 GHz CPU. The algorithms have been run ten times for each instance. We compute the maximum lateness obtained by each algorithm for its instances. Then, based on the best solution obtained for each instance by any of the two algorithms, the relative percentage deviation is computed in three versions: the average (Δ_{avg}), the minimum (Δ_{min}) and the maximum (Δ_{max}). The results of the hybrid algorithms are shown in Table 1. In the first column we present the instances classes. In the other columns we report the relative mean percentage deviations and the average computation time for a given algorithm and a given class. In the last row, we present the mean relative percentage deviation for a given algorithm for all the classes of instances.
Results show that the hybrid genetic algorithm has better performance compared to the genetic algorithm.

Table 1 comparative results of GA-VNS and GA

algorithm	GA-VNS					GA		
instance	Δ_{min}	Δ_{avg}	Δ_{max}	t_{avg}	Δ_{min}	Δ_{avg}	Δ_{max}	t_{avg}
20-2	0.00	0.00	0.00	1.00	0.00	0.00	0.00	1.00
20-5	0.00	0.00	0.00	1.00	0.00	0.00	0.00	1.00
20-10	0.00	0.00	0.00	1.00	0.00	0.00	0.00	1.00
20-20	0.00	0.01	0.02	1.50	0.00	0.02	0.03	1.50
50-2	0.00	0.01	0.02	2.50	0.00	0.01	0.02	2.50
50-5	0.00	0.00	0.00	2.50	0.01	0.02	0.03	2.50
50-10	0.00	0.00	0.00	2.50	0.00	0.00	0.00	2.58
50-20	0.00	0.00	0.00	2.50	0.01	0.02	0.00	2.68
100-2	0.00	0.00	0.00	5.00	0.00	0.00	0.00	5.12
100-5	0.00	0.00	0.00	5.23	0.00	0.00	0.00	5.36
100-10	0.00	0.03	0.05	5.46	0.02	0.04	0.06	5.78
100-20	0.02	0.04	0.06	5.79	0.03	0.06	0.08	5.93
200-2	0.01	0.00	0.00	10.26	0.01	0.03	0.05	10.32
200-5	0.00	0.00	0.00	10.78	0.00	0.00	0.00	10.81
200-10	0.02	0.05	0.08	11.03	0.03	0.04	0.08	11.50
200-20	0.00	0.00	0.00	11.38	0.00	0.00	0.00	11.83
average	0.00	0.00	0.01	4.96	0.00	0.01	0.02	5.09

Conclusion

In this paper we propose a hybrid meta-heuristics (GA-VNS). This algorithm was applied to solve the Uniform Parallel Machine Scheduling Problem with the aim of minimizing the maximum lateness. In order to compare the performance of our algorithms we generated a large set of instances with different sizes (small, medium and large). Our results show that the hybrid genetic algorithm yields the best results. As a perspective, we will try to solve this type of problem using other hybrid meta-heuristics involving exact approaches.

REFERENCES

[1] R.L. Graham, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey,Annals of Discrete Mathematics 5 (1979) 287–326.

[2] J.K. Lenstra, A.H.G. Rinnooy Kan, P. Brucker, Complexity of machine scheduling problems, Annals of Discrete Mathematics 1 (1977) 343–362.

[3] C. Koulamas, G.J. Kyparisis, Scheduling on uniform parallel machines to minimize maximum lateness, Operations Research Letters 26 (2000) 175–179.

[4] M.M. Dessouky, Scheduling identical jobs with unequal ready times on uniform parallel machines to minimize the maximum lateness, Computers & Industrial Engineering 34 (4) (1998) 793–806.

[5] J. Carlier, Scheduling jobs with release dates and tails on identical parallel machines to minimize the makespan, European Journal of Operational Research 29 (1987) 298–306.

[6] E. Néron, F. Tercinet, F. Sourd, Search tree based approaches for parallel machine scheduling, Computers & Operations Research 35 (4) (2008) 1127–1137.

[7] N. Vakhania, Single-machine scheduling with release times and tails, Annals of Operations Research 129 (2004) 253–271.

[8] A. Gharbi, M. Haouari, An approximate decomposition algorithm for scheduling on parallel machines with heads and tails, Computers & Operations Research 34 (2007) 868–883.

[9] M. Mastrolilli, Efficient approximation schemes for scheduling problems with release dates and delivery times, Journal of Scheduling 6 (2003) 521–531.

[10] J. Carlier, E. Pinson, Jackson's pseudo preemptive schedule for the P|ri, qi|Cmax, Annals of Operations Research 83 (1998) 41–58.

[11] T. Eren, A note on minimizing maximum lateness in an m-machine scheduling problem with a learning effect, Applied Mathematics and Computation 209 (2009) 186–190.

[12] C-C. Wu, W-C. Lee, T. Chen, Heuristic algorithms for solving the maximum lateness scheduling problem with learning considerations, Computers & Industrial Engineering 52 (2007) 124–132.

[13] R. Uzsoy, J.D. Velasquez, Heuristics for minimizing maximum lateness on a single machine with family-dependent set-up times, Computers & Operations Research 35 (2008) 2018–2033.

[14] J.K. Lenstra, Sequencing by enumerative methods, in: Mathematical Centre Tracts, vol. 69, entre for Mathematics and Computer Science, Amsterdam, 1977.

[15] Mladenovic N., and P., Hansen. Variable neighbourhood search. Computers and Operations Research, 24:11, 1097-1100 (1997).

[16] Glover F., and G.A., Kochenberger. Handbook of metaheuristics. Boston, MA:Kluwer Academic Publisherh (2003).