Research Journal of Applied Sciences, Engineering and Technology 4(3): 245-248, 2012
ISSN: 2040-7467

Submitted: December 10, 2011 Accepted: January 27, 2012 Published: February 01, 2012

Adapting a Heuristic Oriented Methodology for Achieving Minimum Number of Late Jobs with Identical Processing Machines

Hadi Mokhtari
Young Researchers Club, Science and Research Branch, Islamic Azad University, Tehran, Iran

Abstract: This study deals with an identical parallel machines scheduling problem where the objective is to minimize the number of jobs be late. The decision on this problem is known as a NP-Hard case. Hence, in this paper, a novel heuristic evolutionary technique which is based on a simple principle, easy to implement, with excellent evolutionary performance, is designed to achieve the optimal/near optimal solution for the considered issue. A sequence of solutions are generated by iterating over a greedy construction heuristic in terms of destruction and construction phases and then an improving local search is conducted to more improve the search performance. In order to assess the effectiveness of the heuristic, some simulation experiments are carried out which reveal out performance of the proposed heuristic as opposed to the traditional evolutionary framework.

Key words: Construction/destruction, greedy operations, heuristic algorithm, late jobs, parallel machines, production scheduling

INTRODUCTION

Most of the researches performed on parallel machines scheduling are focused on the identical case. For example Guinet (1993) presented a makespan minimization model for the identical parallel machines problem where setup times are sequence-dependent and proposed a heuristic for solving it. Moreover, Gendreau *et al*. (2001) developed lower bounds and a divide and merge heuristic algorithm with the minmax objective for multi machine scheduling problem with sequence dependent setup times. Anghinolfi and Paolucci (2007) suggested a hybrid metaheuristic integrating several features from Tabu Search (TS), Simulated Annealing (SA) and Variable Neighborhood Search (VNS) for solving a parallel machines job scheduling problem with total tardiness objective, non-zero ready times, distinct due dates, uniform machines and setup times. Shim and Kim (2007) focused on a scheduling problem with n jobs and m parallel machines and developed dominance rules and lower bounds on the tardiness of jobs for a partial schedule, and a branch and bound algorithm using them. Additionally, Rocha *et al.* (2007) proposed a model to analyze a VNS algorithm for parallel processor scheduling problems with sequence dependent setup times. Compared to a greedy randomized adaptive search procedure, the comparison results indicate the advantages of VMS in terms of generality, quality and speed for large instances. Biskup *et al*. (2008) considered the total tardiness minimization as performance measure of a parallel machines scheduling problem, and then developed a new general heuristic for finding optimal or near-optimal schedules. Driessel and Mönch (2011) discussed a job scheduling problem on identical parallel machines with ready times of jobs, precedence constraints, and sequence-dependent setup times. In their study, the total weighted tardiness that is related to on time delivery is regarded as performance measure to be minimized. Hu *et al*. (2010) developed a heuristic algorithm which is based on a combination of the largest total amount of processing first rule and the enhanced smallest machine load first rule for solving parallel machine scheduling problem with precedence constraints and machine eligibility restrictions with makespan measure. Only a few studies have adapted the case of resource-dependent processing time into more complex shop scheduling environments such as parallel machines, flow shops, and job sops. A number of parallel machine scheduling with additional limited resources were presented in literature (Li *et al.*, 2011). More recently, Mokhtari *et al*. (2011) developed a multi-objective permutation flow shop problem with resource-dependent processing times and several type of resources available.

Problem statement: In this section we aim at defining the problem considered in this paper. In the proposed problem, a set of n orders $\{J_1, J_2, \ldots, J_n\}$, each one has a customer due date, is to be processed on *m* identical parallel machines $\{M_1, M_2, \ldots, Mm\}$. The jobs can be processed by either of the machines. The processing times

are sequence independent and transportation times are negligible. Each order is to be processed without preemption. Machines have identical velocity and can process only one order at a given time. All the jobs are available at the beginning of planning period and cancelation is not allowed. All machines are available in the planning period without breakdown. The idleness is allowed for machines. The parameters are deterministic and known in advance and m and n are fixed.

An iterated heuristic solution algorithm: During recent years, different optimal seeking methods have been applied to solve complex optimization problems. The Iterated Greedy (IG) algorithm is a recent novel meta-heuristic that was first proposed by Ruiz and Stützle (2007). In its basic form, heuristic starts from some initial solution and then iterate through a main steps. There are two main phases in each IG method:

- Destruction
- Construction

In the destruction step, a partial solution is produced by randomly shifting a number of elements from the solution. In the construction step, a greedy heuristic is applied to sequentially insert the removed elements into the partial candidate solution until a complete solution is reconstructed. In sequel we describe how we have devised the method for the parallel machines scheduling problem proposed in current paper. Figure 1 shows the general framework of IG heuristic.

Solution representation form: A solution is presented by two different parts. The first one which is based on scheduling component is coded by a random key of n numbers on interval [1, m+1), in which each number represents the job assignment by its integer part, and shows the sequence of jobs on machines by its fractional part. Each job with the lower fractional part on a certain machine, has greater priority to be scheduled. For example, consider a problem with n = 5, m = 3 and processing time p = {5,10,20,5,25}. A solution encoded as RK = {3.58, 2.41, 2.06, 1.91, 3.41} can be decoded as follows. As RK vector represents, there are five jobs and four machines. Since just one job (J_4) has the RK with integer part 1, the job J_4 is only assigned to the first machine. Among the jobs that have RKs with integer part 2 (J_2, J_3), the job J_3 is planned at first J_2 is planned after that based on their RKs' fractional parts. Similarly, the sequence of jobs on third machine is ($J_5 \rightarrow J_1$).

Destruction and construction functions: The destruction and construction are two main operations of proposed method. After that the initial solution t_O is

```
Select the initial solution t at random;
While termination condition does not meet do
    Run destruction on t to generate partial solution t1;
    Run construction on t1 to generate complete solution t2;
    Apply decision criterion on t2 o reach solution t;
Endwhile
```

Fig. 1: The general framework of IG

```
Set k = 0;
While k ≤ kmax do
    S = 0;
    S max = 2;
    While S < Smax do
        Select r1, r2 at random;
        If S == 0 then t* = insert (t, r1, r2)
        elseif S == 1 then t* = interchange (t, r1, r2)
        endif
        if f(t*) ≤ f (t)then
            S = 0;
            P = P1;
        else
            S = S +1;
        endif
    endwhile
    k = k + 1;
endwhile
if f (t*) ≤ f (t) then
    t = t*;
endif
```

Fig. 2: Pseudo code of local search

chosen randomly, a string of d elements of t_o is randomly eliminated from it which creates two partial solutions; the first one of size d is called t_R and contains the removed elements. The second segment with size (n-d) is the original one without the removed elements that is called t_D. Once two partial solutions are created, the construction phase is commenced. Here, all elements in t_R are reinserted in t_D by applying the well-known dispatching rule Shortest Processing Times (SPT). The SPT orders the RKs of removed elements in t_D in ascending job of their processing times. In this step it is decided whether the reconstructed solution is accepted or not as the incumbent solution for the next iteration.

Improving with additional local search: A local search is a classical optimization method that consists of generating a local optimal solution by employing moves from one solution to another. One of the main steps when designing a local search method is the designing of the neighborhood structure. In other words, a neighborhood is delineated by modifying some components of a given solution to create a new neighboring solution. Defining an effective neighborhood structure makes efficient move from one solution to another. As it is a common idea in literature, in this section we employ a simple but efficient local search after that re-constructed solution is achieved at each iteration of suggested algorithm. In fact, such a

hybridization of construction heuristics with local search is common practice in stochastic local search (Ruiz and Stützle, 2008). The adapted local search is a kind of neighborhood search which is based on Pair-wise Interchange and Insert operators. After that destruction phase is performed and the new complete solution is reconstructed, the elements of two randomly selected positions are interchanged as Pair-wise Interchange, and then a random position of the resultant solution is chosen and its element is placed in a new position at random in Insertion phase. If this new solution results in better performance, the current solution t is replaced by the new one. This procedure iterates until a predefined number of replications is met. The pseudo code of suggested local search is illustrated in Fig. 2.

Experiments and comparisons: The Genetic Algorithm (GA) which was first introduced by Holland (1975) is one of the powerful stochastic search techniques recently used for solving complex optimization problems. This algorithm imitates the procedures of survival of fittest tenet and offspring in natural world. Typically, a random initial population is created and probabilistic operations are used to generate the subsequent generations. Trough some basic operations, the algorithm directs the population towards the optimal/near optimal solution(s). Individuals with the better fitness have higher chance to produce the offspring. The operation of evolution is usually performed many times until a predefined stopping condition is violated. In recent years, many different metaheuristics have been conducted for solving scheduling problems. Since the genetic algorithms have been effectively adapted, we apply it in current paper as a comparison metric to evaluate the effectiveness of proposed two-layered heuristic. In this regard, a vector of processing time compression is regarded as solution encoding format. The population is initialized with random generated solutions. The well-known uniform crossover and swap-based mutation are also considered for intensification and diversification strategy. In uniform crossover, every gene of parents is compared together separately and their numbers are swapped with a predefined probability. A set of uniform random numbers are generated on interval [0,1]. If the corresponding number is less than 0.5, the element from first parent is selected, and if it is larger than 0.5, the other is chosen to be moved into offspring. Consider the following example. A manufacturer with three numbers of available machines receives five jobs at the beginning of planning period.

In adapted mutation operator, one element from parent is selected randomly and its symbol is replaced with a new random generated gene. After that a new solution is achieved, the list scheduling heuristic is carried out to evaluate the fitness of that solution. Once a predefined number of generation is met, the GA is terminated.

Table 1: The computational results for small instances

Problem sizes	Benchmark GA			Proposed heuristic		
(n× m)	Best	Worst	Mean	Best	Worst	Mean
(5×2)	2.65	3.66	3.25	2.74	3.17	3.01
(5×3)	3.79	4.56	4.07	3.61	5.04	4.18
(10×2)	6.59	7.47	7.00	6.30	7.43	6.85
(10×3)	8.39	9.38	8.86	8.21	8.58	8.42
(10×5)	6.31	7.85	6.96	5.90	7.55	6.46
(10×2)	11.64	13.70	12.70	9.61	11.77	10.76
(10×3)	10.71	12.50	11.57	9.63	11.21	10.37
(10×5)	13.24	13.86	13.59	10.78	13.01	11.66
(20×2)	15.66	17.97	16.91	12.68	14.86	13.96
(20×3)	14.53	16.08	15.27	13.95	14.79	14.39

Table 2: The computational results for medium instances

Problem sizes	Benchmark GA			Proposed heuristic		
(n× m)	Best	Worst	Mean	Best	Worst	Mean
(30×10)	24.94	28.96	26.72	24.29	27.48	26.31
(30×15)	26.45	29.07	27.74	24.11	27.40	25.80
(30×20)	19.92	24.18	21.35	17.55	21.99	20.26
(50×10)	40.28	42.26	40.98	39.50	45.10	41.82
(50×15)	45.66	51.25	47.62	41.02	44.59	43.11
(50×20)	43.10	46.13	44.25	40.98	44.08	42.29
(70×10)	54.22	56.87	55.23	54.73	56.18	55.25
(70×15)	56.12	59.68	57.73	55.11	57.73	56.05
(70×20)	56.86	63.73	59.76	58.69	62.73	60.89
(70×20)	63.09	66.28	64.55	60.24	62.82	61.73

Table 3: The computational results for large instances

Problem sizes	Benchmark GA			Proposed heuristic		
(n× m)	Best	Worst	Mean	Best	Worst	Mean
(80×15)	61.45	65.45	63.89	61.53	67.53	64.24
(80×20)	67.06	70.90	69.24	63.70	67.84	66.44
(80×30)	68.11	71.88	70.01	66.66	73.11	69.21
(100×15)	73.45	84.73	80.65	74.91	80.53	78.35
(100×20)	84.47	86.63	85.48	80.71	89.86	84.67
(100×30)	84.79	89.80	87.60	84.10	88.32	86.44
(120×15)	101.54	104.99	103.83	100.26	101.68	100.92
(120×20)	99.02	101.44	100.30	95.21	97.61	96.78
(120×30)	102.64	109.68	106.80	100.40	103.71	102.43
(120×50)	108.91	111.75	109.94	103.19	107.06	105.36

To carry out the comparisons, the GA was implemented in MATLAB environment, and a set of random benchmarks are employed. Table 1 shows the experimental results in terms of minimum (Min.), maximum (Max.) and average (Ave.) of results obtained by adapted GA, and suggested heuristic.

As the results presented in Table 1-3 reveal, the suggested heuristic outperforms the GA in strong sense. Our heuristic could find better solutions than GA for all the test problems except 1 small problem, 2 medium problems, and 2 large problems. Furthermore, the average value of best solutions obtained by our method is 44.343, while that of GA is 45.853 which shows the superiority of suggested IG.

CONCLUSION

In this study, an iterated heuristic is devised for achieving near optimal schedule of job scheduling problem in parallel machine environment. It is considered that a common non-renewable resource is available to be allocated to the jobs and the objective is to simultaneously minimize the number of late jobs and the amount of resources consumed. Motivated by the computational complexity of this problem, a novel heuristic is devised to cope with the problem considered. In suggested heuristic, a sequence of solutions is generated by iterating over a greedy construction heuristic based on destruction and construction functions. Moreover, local search phases have been embedded into the heuristic to refine the search in the region close to the candidate solution and enhance the performance of algorithm in achieving optimal/near optimal solutions. To validate and verify the proposed heuristic, computational experiments were conducted on some random test problems including small, medium and large instances. The extensive experiments and simulation analyses demonstrate that the good performance of proposed algorithm against GA technique.

REFERENCES

Anghinolfi, D. and M. Paolucci, 2007. Parallel machine total tardiness scheduling with a new hybrid metaheuristic approach. Comp. Oper. Res., 34: 3471-3490.

Biskup, D., J. Herrmann and J.N.D. Gupta, 2008. Scheduling identical parallel machines to minimize total tardiness. Int. J. Prod. Econ., 115(1): 134-142.

Driessel, R. and L. Mönch, 2011. Variable neighborhood search approaches for scheduling jobs on parallel machines with sequence-dependent setup times, precedence constraints and ready times. Comp. Indus. Eng., 61(2): 336-345.

Gendreau, M., G. Laporte and E. Morais-Guimaraes, 2001. A divide and merge heuristic for the multiprocessor scheduling problem with sequence dependent setup times. Eur. J. Oper. Res., 133: 183-189.

Guinet, A., 1993. Scheduling sequence-dependent jobs on identical parallel machines to minimize completion time criteria. Int. J. Prod. Res., 31: 1579-1594.

Holland, J., 1975. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor.

Hu, X., J.S. Bao and Y. Jin, 2010. Minimizing make span on parallel machines with precedence constraints and machine eligibility restrictions. Int. J. Prod. Res., 48(6): 1639-1651.

Li, K., Y. Shi, S.L. Yang and B.Y. Cheng, 2011. Parallel machine scheduling problem to minimize the makes pan with resource dependent processing times. Appl. Soft Comp. J., VOL: pp, In Press.

Mokhtari, H., I.N.K. Abadi and A. Cheraghalikhani, 2011. A multi-objective flow shop scheduling with resource-dependent processing times: Trade-off between makes pan and cost of resources. Int. J. Prod. Res., 49(19): 5851-5875.

Ruiz, R. and T. Stützle, 2007. A simple and effective iterated greedy algorithm for the permutation flow shop scheduling problem. Eur. J. Oper. Res. 177: 2033-2049.

Ruiz, R. and T. Stützle, 2008. An Iterated Greedy heuristic for the sequence dependent setup times flow shop problem with makes pan and weighted tardiness objectives, Eur. J. Oper. Res., 187(3): 1143-1159.

Rocha, M.L., M.G. Ravetti, G.R. Mateus and P.M. Pardalos, 2007. Solving parallel machines scheduling problems with sequence-dependent setup times using variable neighborhood search. IMA J. Manage. Mathe. 18: 101-115.

Shim, S.O. and Y.D. Kim, 2007. Scheduling on parallel identical machines to minimize total tardiness. Eur. J. Oper. Res., 177(1): 135-146.