Heuristics for Scheduling Uniform Machines

Domenico De Giovanni, Johnny C. Ho, Giuseppe Paletta, and Alex J. Ruiz-Torres

***Abstract*—A new Modified Longest Processing Time algorithm and an Iterated Local Search algorithm are developed for the scheduling problem in which independent jobs are nonpreemptively scheduled on uniform parallel machines with the objective of minimizing the makespan, i.e., the completion time of the last job. Our computational results show that the Modified Longest Processing Time algorithm is able to reduce the average error, to increase the number of optimal solutions, and to determine a greater number of best solutions with respect to the Longest Processing Time algorithm. Furthermore, the Iterated Local Search algorithm is shown to be effective in reducing the average error significantly and yielding optimal solutions in over 80% of the tested instances.**

***Index Terms*—scheduling, uniform parallel machines, longest processing time algorithm, iterated local search algorithm**

I. Introduction

This paper considers the uniform machine scheduling problem, denoted by *Q*||*Cmax* [1]. Given *n* independent jobs $J = \{1, ..., j, ..., n\}$ with processing times p_j such that $p_1 \geq ... \geq p_j \geq ... \geq p_n$ and *m* uniform machines $M = \{1, ..., i, ..., m\}$ with processing speed s_i such that $s_1 \geq ... \geq s_i \geq ... \geq s_m = 1$, the objective of *Q*||*Cmax* is to assign the *n* jobs to the *m* machines in such a way to minimize the makespan. Without loss of generality, we assume that the processing times are integers. Also, all jobs and machines are available at time zero, and no job preemption is allowed. A schedule is represented by an *m*-partition $S = \{S_1, ..., S_i, ..., S_m\}$ of the set *J*, where S_i is the subset of jobs assigned to machine *i*, $i = 1, ..., m$. Thus, the total processing time of job subset S_i, $T(S_i)$, is $\sum_{j \in S_i} p_i$ and the completion time for S_i is $C(S_i) = T(S_i)/s_i$. The makespan for schedule *S*, *Cmax*(*S*), is equal to $\max_{i=1,...,m}\{C(S_i)\}$. An optimal schedule is defined as one where its makespan is at the minimum. *Q*||*Cmax* is known to be strongly NP-Hard for $m \geq 2$, see [2].

The classical Longest Processing Time (*LPT*) heuristic was developed by Graham [3] initially for solving the scheduling problem of independent jobs on equivalent machines, denoted by *P*||*Cmax*, see [1]. Gonzalez, Ibarra, and Sahni [4] adapted the *LPT* heuristic for solving *Q*||*Cmax*; their *LPT* heuristic considers the jobs sorted in non-increasing order with respect to the processing times, and iteratively assigns each job to the machine on which it will complete first. Gonzalez, Ibarra, and Sahni [4], Dobson [5], and Friesen [6] discussed the worst-case bounds of the *LPT* heuristic on multiple uniform machines. Mireault, Orlin, and Vohra [7] found the worst-case bound of *LPT* as an explicit function of the ratio of machine speeds for two uniform machines. Massabó, Paletta, and Ruiz-Torres [8] developed a posterior worst-case bound of the *LPT* algorithm for two uniform machines. Friesen and Langston [9] and Chen [10] analyzed the worst-case bounds of *MULTIFIT* scheduling on uniform processors. Koulamas and Kyparisis [11] studied the *LPT* algorithm for *Q*2||*Cmax* when the *k* longest jobs are first scheduled optimally, and then the remaining jobs are scheduled according to the *LPT* algorithm.

Horowitz and Sahni [12] developed a dynamic programming algorithm to derive the optimal solution; however, it has an exponential time complexity in the worst case. Hochbaum and Shmoys [13] presented a polynomial approximation scheme for the problem. The papers by Lin and Liao [14] and Liao and Lin [15] proposed optimal solution algorithms, based on lexicographic search, to solve the two-machine and the multiple machines cases, respectively.

We propose a Modified Longest Processing Time (*MLPT*) algorithm and an Iterated Local Search (*ILS*) algorithm for solving uniform parallel machine scheduling problem with minimum makespan objective. *MLPT* modifies the Longest Processing Time (*LPT*) algorithm of Gonzalez, Ibarra, and Sahni [4]. The main difference is in the rule used to assign the job subsets S_i, $i = 1, ..., m$, to the machines. In our algorithm the assignment of job subsets S_i, $i = 1, ..., m$, can change if it involves an improvement of the makespan; whereas in Gonzalez, Ibarra, and Sahni's *LPT* algorithm each S_i is always assigned to the machine M_i. *ILS* is an improvement procedure, using *MLPT* as the seed solution. We evaluate the effectiveness of the proposed procedures through a large-scale computational study. The computational results show that the *MLPT* algorithm is able to reduce the average error, to increase the number of optimal solutions, and to find a larger number of best solutions with respect to the *LPT* algorithm. Moreover, the *ILS* algorithm significantly reduces the average error and provides optimal solutions in over 80% of the tested instances.

The rest of this paper is organized as follows. A new Modified Longest Processing Time (*MLPT*) algorithm is proposed in Section 2 and an Iterated Local Search (*ILS*)

Manuscript received September 29, 2017; revised December 10, 2017.

Domenico De Giovanni is with Dipartimento di Economia e Statistica, Università della Calabria, 87036 Arcavacata di Rende (CS), Italy (e-mail: domenico.degiovanni@unical.it).

Johnny C. Ho is with Department of Management and Marketing, Turner College of Business, Columbus State University, Columbus, GA, 31907, USA (e-mail: ho_johnny@columbusstate.edu).

Giuseppe Paletta is with Dipartimento di Economia e Statistica, Università della Calabria, 87036 Arcavacata di Rende (CS), Italy (e-mail: giuseppe.paletta@unical.it).

Alex J. Ruiz-Torres is with Departamento de Gerencia, Facultad de Administración de Empresas, Universidad de Puerto Rico–Rio Piedras, San Juan, PR, 00931-3332, USA (e-mail: alex.ruiztorres@uprrp.edu).

algorithm is proposed in Section 3. The results of a computational experimentation are reported in Section 4.

II. A Modified LPT Algorithm

The basic idea of this procedure is that when a job is assigned to the machine on which it will complete first, an improvement can be obtained by exchanging the job sets between two machines. *MLPT*(*S*) considers the jobs sorted in non-increasing order with respect to the processing times, and iteratively it:

- assigns each job to one of the *m* job subsets, and
- reassigns the job subsets to the machines if necessary

with the objective of minimizing the makespan.

This procedure is a modified version of the Gonzalez, Ibarra, and Sahni's *LPT* algorithm. The main difference is in the rule used to assign the job subsets S_i, $i = 1, ..., m$, to the machines. In our algorithm the assignment of job subsets S_i, $i = 1, ..., m$, can change if it involves an improvement of the makespan; whereas in Gonzalez, Ibarra, and Sahni's *LPT* algorithm each S_i is always assigned to the machine M_i. Formally, the procedure can be described as follows.

MLPT(*S*) Procedure

- Sort the jobs in non-increasing order with respect to p_j, $j = 1, ..., n$. Set $S_i = \emptyset$ for each $i = 1, ..., m$. Initialize $C_{max}(S) = 0$.
- For $j = 1$ to n:
  - Find $i^* = arg\ min_{i=1,...,m}\{\frac{T(S_i)+p_j}{s_i}\}$. If assigning job j to machine i^* does not change the current makespan, $C_{max}(S)$, then set $S_{i^*} = S_{i^*} \cup \{j\}$; otherwise,
  - Compute $(i^*, l^*) = arg\ min_{i=1,...,m}\{max_{l=1,...,m}\{\frac{T(S_i)+p_j}{s_l}, \frac{T(S_l)}{s_i}\}\}$ and assign the job j to the machine i^* by setting $S_{i^*} = S_{i^*} \cup \{j\}$. If $i^* \neq l^*$, then exchange S_{i^*} with S_{l^*}. Update $C_{max}(S)$.

  End for.
- Return *S*.

III. Iterated Local Search Algorithm

An Iterated Local Search algorithm, see Lourenço, Martin, and Stützle [16, 17], is based on building a sequence of locally optimal solutions by using the following framework:

a. Generate an initial solution.
b. Use a local search procedure to improve the initial solution.
c. Repeat
   - Perturb the local optimum solution to obtain a new starting solution;
   - Use a local search procedure to improve the starting solution;
   - Use an acceptance criterion to determine from which solution the search of new starting point should continue;

   Until termination condition is met.

Following these ideas, our algorithm constructs the initial solution by the *MLPT*(*S*) procedure and improves by applying the local search procedure *LS*(*S*). Then, it iteratively perturbs the current local optimum solution *S* in order to obtain a new starting solution $\bar{S}$, and generates a new local optimum solution $\bar{S}$ by using *LS*(*S*). The best of *S* and $\bar{S}$ is chosen as the current local optimum solution *S*. Thus, the perturbation is always performed on the best current solution *S* of a given iteration (acceptance criterion). Two perturbation procedures are considered with the aim of leading to the exploration of regions of the solution space not previously visited. These procedures, referred to as *First Diversification Procedure* (*FDP*(*S*,$\bar{S}$,*h*)) and *Second Diversification Procedure* (*SDP*(*S*,$\bar{S}$,*h*)), perturb the current local optimum solution *S* by obtaining a new starting solution $\bar{S}$ in function of the parameter *h*, $1 \leq h \leq m$. The algorithm terminates when no improvement is realized after *m* iterations, or when a lower bound is achieved. The lower bound $LB_{[P-W]}$ determined by Lin and Liao [14] is used in the termination test of the algorithm. For the sake of completeness the lower bound $LB_{[P-W]}$ is described as follows. Let's consider $P = \sum_{j=1,n} p_j$, $S = \sum_{i=1,m} s_i$, $w_i = \lfloor s_i P/S \rfloor$, $W = \sum_{i=1,m} w_i$, and $K = (P - W)$. Compute $LB_{[0]} = P/S$, and $LB_{i,k} = (w_i + k)/s_i$ for $i = 1, ..., m$ and $k = 1, ..., K$. Sort the $LB_{i,k}$, $i = 1, ..., m$ and $k = 1, ..., K$ and denote the sorted sequence by $LB_{[0]} \leq LB_{[1]} \leq LB_{[2]} \leq ... \leq LB_{[K]} \leq ... \leq LB_{[mK]}$. $LB_{[P-W]}$ is a lower bound on makespan for $Q||C_{max}$. The following is a general schema of the proposed *ILS*(*S*) algorithm.

ILS(*S*) Algorithm

Step 0. Perform *MLPT*(*S*) to obtain a feasible solutions $S = \{S_1, ..., S_i, ..., S_m\}$. If $C_{max}(S) = LB_{[P-W]}$, then go to Step 6.
Step 1. Perform *LS*(*S*). If $C_{max}(S) = LB_{[P-W]}$, then go to Step 6.
Step 2. Set $h = 1$.
Step 3. Repeat
  a. Perform *FDP*(*S*,$\bar{S}$,*h*);
  b. Perform *LS*($\bar{S}$):
    If $C_{max}(S) > C_{max}(\bar{S})$ then set $S = \bar{S}$ and if $C_{max}(S) = LB_{[P-W]}$ then go to Step 6; otherwise, return to Step 2;
    If $C_{max}(S) \leq C_{max}(\bar{S})$ then set $h = h + 1$;

  Until $h \leq m$.
Step 4. Set $h = 1$.
Step 5. Repeat
  a. Perform (*SDP*(*S*,$\bar{S}$,*h*));
  b. Perform *LS*($\bar{S}$):
    If $C_{max}(S) > C_{max}(\bar{S})$ then set $S = \bar{S}$ and if $C_{max}(S) = LB_{[P-W]}$ then go to Step 6; otherwise, return Step 4;
    If $C_{max}(S) \leq C_{max}(\bar{S})$ then set $h = h + 1$;

  Until $h \leq m$.
Step 6. Return $S = \{S_1, ..., S_m\}$ and $C_{max}(S)$.

Details of the *LS*(*S*), *FDP*(*S*,$\bar{S}$,*h*), and *SDP*(*S*,$\bar{S}$,*h*) procedures are available from the authors upon request.

IV. Computational Results

The proposed algorithms were implemented in Fortran and all experiments conducted on a personal computer with an Intel Core i5 processor. A large number of experiments are conducted to test the effectiveness of the proposed heuristics. Four experimental parameters, consistent with previous experiments by Lin and Liao [14], are considered:

ISBN: 978-988-14048-8-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

(i) the ratio of the number of jobs to the number of machines, n/m;
(ii) number of machines m;
(iii) job processing requirements p_j; and
(iv) machine speeds s_i.

Four levels are analyzed for the number of machines factor: 3, 4, 5, and 10. Ten levels are analyzed for the n/m ratio factor: 2, 3, 4, 5, 10, 20, 30, 40, 50, and 100. The integer processing requirements of the jobs were randomly generated from a discrete uniform distribution $U(1, p_{max})$ and we consider four levels for the p_{max} factor: 25, 50, 100, and 200. Finally, machine speeds were randomly generated from a uniform distribution $U(1, s_{max})$ and we consider three values for the s_{max} factor: 3, 5, and 7. It is noted that for each problem instance the generated speed values were rounded off to three decimal places. There are a total of 480 experimental combinations, and for each combination 100 replications were made. Hence, we reported the results of the total 48,000 instances. Specifically, there are 4,800 instances for each value of n/m, 12,000 instances for each value of m and for each value of p_{max}, and 16,000 instances for each value of s_{max}.

The performances of the heuristics have been evaluated with respect to Lin and Liao's lower bound $LB_{[P\ -W]}$. We define the following measures:

e%: the average percentage relative error with respect to the lower bound and averaged on the number of instances for the corresponding parameter.
o%: the percentage of the number of instances (out of all the instances of the corresponding parameter) that have been solved to optimality.
b%: the percentage of the number of times (out of the number of instances of the corresponding parameter) that the heuristic gives a solution better than the other.

Our computational results show that heuristics *LPT* and *MLPT* determine about the same percentage of optimal solutions for the complete problem set: *MLPT* finds the optimal solution in at least the 36.302% of the instances (corresponding to 17,425 of the 48,000 instances); whereas *LPT* finds the optimum in at least the 35.521% of the instances (corresponding to 17,050 of the 48,000 instances). The overall average relative error of *MLPT* is 1.162%, while for *LPT* is 1.540%. The percentage of time that *MLPT* gives a solution better than the *LPT* heuristic is the 35.250% of the instances (corresponding to 16,920 of the 48,000 instances); whereas the number of times that *LPT* gives a better solution than *MLPT* is 23.058% of the instances (corresponding to 11,068 of the 48,000 instances). In the remaining instances, *MLPT* and *LPT* give the same solution. These results demonstrate that the *MLPT* heuristic in general outperforms the *LPT* approach. As can be noted, computational times are insignificant as solving all 48,000 instances only required 15.7 seconds for *MLPT* (and less for *LPT*).

The results indicate that heuristic performance was sensitive to three of the experimental factors: p_{max}, n/m, and m. The experimental factor s_{max} does not seem to have an effect on heuristic performance for any of the three relevant measures of performance (e%, o%, and b%).

For both *LPT* and *MLPT*, as p_{max} increased the error e% increased, the percentage of optimal solutions o% decreased, and the percentage of times a heuristic dominated the other b% increased. These results seem to indicate that as p_{max} increased the problems became more difficult (higher error and less optimal solutions) and each approach generated different solutions (one heuristic dominated the other). As the n/m ratio increased, the error percentage e% for both heuristics decreased, with the relative difference being notable at low levels of the n/m ratio.

Regarding the performance of the *LS* and *ILS* algorithms, the computational results demonstrate that *LS* finds the optimal solution in at least 75.82% of the instances with an overall average relative error of 0.44%, greatly improving the performance of the algorithm *MLPT*. Furthermore, *ILS* finds the optimal solution in at least 83.96% of the instances with an overall average relative error of 0.336%. The time required to solve all 48,000 instances is still relatively insignificant for both *LS* and *ILS* at less 69.9 and 132.3 seconds, respectively.

REFERENCES

[1] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan, "Optimization and approximation in deterministic sequencing and scheduling: A survey," *Annals of Discrete Mathematics*, vol. 5, pp. 287–326, 1979.
[2] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: W. H. Freeman, 1979.
[3] R. L. Graham, "Bounds on multiprocessing timing anomalies," *SIAM Journal on Applied Mathematics*, vol. 17, pp. 416–429, 1969.
[4] T. Gonzalez, O. H. Ibarra, and S. Sahni, "Bounds for LPT schedules on uniform processors," *SIAM Journal on Computing*, vol. 6, pp. 155–166, 1977.
[5] G. Dobson, "Scheduling independent tasks on uniform processors," *SIAM Journal on Computing*, vol. 13, pp. 705–716, 1984.
[6] D. K. Friesen, "Tighter bounds for LPT scheduling on uniform processors," *SIAM Journal on Computing*, vol. 16, pp. 554–560, 1987.
[7] P. Mireault, J. B. Orlin, and R. V. Vohra, "A parametric worst case analysis of the LPT heuristic for two uniform machines," *Operations Research*, vol. 45, pp. 116–125, 1997.
[8] I. Massabó, G, Paletta, and A. J. Ruiz-Torres, "A note on longest processing time algorithms for the two uniform parallel machine makespan minimization problem," *Journal of Scheduling*, vol. 19, pp. 207–211, 2016.
[9] D. K. Friesen, and M. A. Langston, "Bounds for MULTIFIT scheduling on uniform processors," *SIAM Journal on Computing*, vol.12, pp. 60–70, 1983.
[10] B. Chen, "Tighter bound for MULTIFIT scheduling on uniform processors," *Discrete Applied Mathematics*, vol. 31, pp. 227–260, 1991.
[11] C. Koulamas and G. J. Kyparisis, "A modified LPT algorithm for the two uniform parallel machine makespan minimization problem," *European Journal Operational Research*, vol. 196, pp. 61–68, 2009.
[12] E. Horowitz and S. Sahni, "Exact and approximate algorithms for scheduling non-identical processors," *Journal of the ACM*, vol. 23, pp. 317–327, 1976.
[13] D. S. Hochbaum and D. B. Shmoys, "A polynomial approximation scheme for scheduling on uniform processors: using the dual approximation approach," *SIAM Journal on Computing*, vol. 17, pp. 539–551, 1988.
[14] C. H. Lin and C. J. Liao, "Makespan minimization for multiple uniform machines," *Computing and Industrial Engineering*, vol. 54, pp. 983–992, 2008.
[15] C. J. Liao and C. H. Lin, "Makespan minimization for two uniform parallel machines," *International Journal of Production Economics*, vol. 84, pp. 205–213, 2003.
[16] H. R. Lourenço, O. Martin, and T. Stützle, "Iterated Local Search: Framework and Applications," *Handbook of Metaheuristics, 2nd. Edition*, International Series in Operations Research & Management Science, Kluwer Academic Publishers, vol. 146, pp. 363–397, 2010.
[17] H. R. Lourenço, O. Martin, and T. Stützle, "Iterated Local Search. Handbook of Metaheuristics," *Handbook of Metaheuristics*, International Series in Operations Research & Management Science, Kluwer Academic Publishers, vol. 57, pp. 321–353, 2003.

ISBN: 978-988-14048-8-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)