Iterated Local Search Based Heuristic for Scheduling Jobs on Unrelated Parallel Machines with Machine Deterioration Effect

Vívian L. Aguiar Santos
Dep. of Computer Science
U. Federal de Viçosa, Brazil
vivian.santos@ufv.br

José Elias C. Arroyo
Dep. of Computer Science
U. Federal de Viçosa, Brazil
jarroyo@dpi.ufv.br

Thales F. M. Carvalho
Dep. of Computer Science
U. Federal de Viçosa, Brazil
thales.carvalho@ufv.br

ABSTRACT

In this research, we study an unrelated parallel machine scheduling problem in which the jobs cause deterioration of the machines. This deterioration decreases the performance of the machines, therefore the processing times of the jobs are increased over time. The problem is to find the processing sequence of jobs on each machine in order to reduce the deterioration of the machines and consequently minimize the makespan. Given that the problem is NP-hard, we propose an Iterated Local Search (ILS) heuristic to obtain near-optimal solutions. In this work we combine ILS meta-heuristic with Random Variable Neighborhood Descent (RVND) local search. The performance of our heuristic is compared with the state-of-the-art meta-heuristic algorithm proposed in the literature for the problem under study. The results show that the our heuristic outperform the existing algorithm by a significant margin.

Keywords

Scheduling; deterioration effect; meta-heuristics.

1. INTRODUCTION

In deterministic scheduling problems, the job processing times are generally known in advance and remain constant during the scheduling horizon. However, there are many practical situations in which the processing time of a job may increase over time, such that the later a job starts, the longer it takes to be processed. This phenomenon is known as the job processing time deterioration effect [4].

Ruiz-Torres et al. [2] presented a job deterioration model that considers the case where the deterioration of the processing time for a job depends on the specific jobs that have been previously processed by the machine. The deterioration of a machine produces deterioration (increasing) of job processing times. These authors addressed an unrelated parallel machine scheduling problem with the makespan minimization.

In this paper we address the same problem formulated by Ruiz-Torres et al. [2]. There is a set of n jobs, $N = \{1, \ldots, n\}$, to be processed on a set of m unrelated parallel machines, $M = \{1, \ldots, m\}$. All jobs are available for processing at time zero and job preemption is not allowed. Each machine can process at most one job at a time and can not stand idle until the last job assigned to it has been finished. The normal processing time of job j on machine k is defined by $p_{j,k}$ and the deteriorating effect of job j on machine k is given by $d_{j,k}$, where, $0 \leq d_{j,k} \leq 1$, $\forall j \in N$ and $k \in M$. The performance level of the machine k to processes the job in position h ($h > 1$) is defined by: $q_{k,[h]} = (1 - d_{[h-1],k}) \times q_{k,[h-1]}$, where, $d_{[h-1],k}$ is the deteriorating effect of job in position $h-1$. It is assumed that the machines start with no deterioration, thus the performance level to processes the job in position 1 is 100%, i.e. $q_{k,[1]} = 1$ for all $k \in M$. The actual processing of a job j processed on machine k in position h is determined by: $p'_{j,k} = \frac{p_{j,k}}{q_{k,[h]}}$. The objective of the problem is to sequence the jobs on the machines in order to minimize the makespan (C_{max}).

In [2] is proved that, for a subset J of jobs assigned to a machine k, the minimum completion time of the machine is found by sequencing the jobs in decreasing order of the ratio: $r_{j,k} = p_{j,k}(1 - d_{j,k})/d_{j,k}, \forall j \in J$. Therefore, the objective of the problem is to determine the subset of jobs for each machine. The problem is NP-hard for $m > 1$ [2].

2. PROPOSED HEURISTIC ALGORITHM

In this work we propose a heuristic, called ILS-RVND, which is based on ILS meta-heuristic [1] and RVND local search [3]. The heuristic uses a dynamic perturbation, i.e., the value of perturbation length is modified during the search process. A general description of the pseudo-code of the ILS-RVND heuristic is shown in Algorithm 1. In Steps 2 and 3, an initial solution is constructed and improved by local search (LS) procedure. In Step 4, the length of the perturbation t is initialized with the minimum value t_{min}. The main iterations of the algorithm are computed in Steps 5 to 15 until the stop criterion is satisfied. During each iteration, the current solution S is perturbed and improved by the LS procedure obtaining a new solution S_2. If the solution S_2 improves the best solution obtained so far, the perturbation length is setted to its lowest level, $t = t_{min}$ (Steps 8-9). If the best solution is not improved, the perturbation length

GECCO '16 July 20-24, 2016, Denver, CO, USA

ACM ISBN 978-1-4503-4323-7/16/07.

DOI: http://dx.doi.org/10.1145/2908961.2909053

is incremented (Step 11). The maximum value for the perturbation length is t_{max}. If the perturbation length exceeds this limit, its value is reset to the minimum value, $t = t_{min}$ (Steps 12-13). The algorithm only accepts better solutions (Steps 14-15).

Algorithm 1: ILS-RVND ($stopC$, t_{min}, t_{max}, β)

```
1  begin
2      S ← INITIAL_SOLUTION();
3      S ← RVND_LOCAL_SEARCH(S);
4      t ← t_min; S* ← S; //best solution
5      while not stopC do
6          S_1 ← PERTURBATION(S, t);
7          S_2 ← RVND_LOCAL_SEARCH(S_1);
8          if f(S_2) < f(S*) then
9              S* ← S_2; t ← t_min;
10         else
11             t ← t + 1;
12             if t > t_max then
13                 t ← t_min;
14         if f(S_2) < f(S) then
15             S ← S_2;
16     return S*
```

To construct an initial solution S, all jobs j are initially arranged in decreasing order of the following rule: $r_j^{average} = \frac{\sum_{k \in M} r_{j,k}}{m}$. Following this ordering, each job is designed to the machine that produces the shortest completion time. The jobs on the machines are arranged by the ratio $r_{j,k}$.

The perturbation procedure is done as follows. First, a set of t machines are randomly chosen: $\{M_1, ..., M_t\}$. This set must contain the machine with a maximum completion time (makespan machine). Next, a job is selected at random in each machine. The job of machine M_1 is inserted in machine M_2, the job of machine M_2 is inserted in machine M_3 and so on until, the job of machine M_t is inserted in machine M_1. The parameter $t \in [t_{min}, t_{max}]$ defines the perturbation length.

The RVND local search is performed by a VND heuristic which utilizes a random neighborhood ordering, that is the move to generate neighbor solutions is randomly selected. Two moves are used: *pairwise exchange* (PI) and *single insertion* (SI). The PI move consists in exchange two jobs i and j from to two different machines M_c and M_k, respectively, where M_c must to be the makespan machine. The SI move consists in remove a job j from the makespan machine M_c and inserting it in other machine M_k. The moves PI and SI define the neighborhoods, N_1 and N_2, respectively. Randomly a neighborhood N_i is selected from $L = \{N_1, N_2\}$ and the best solution S_{Best} in this neighborhood is determined. If S_{Best} improves the current solution, it is replaced by S_{Best} and the list L is reset with the two neighborhoods. Otherwise, N_i is removed from L. The local search finishes when L is empty.

3. RESULTS AND CONCLUSIONS

The heuristic ILS-RVND was tested on 900 medium-size instances ($n \in \{20, 35, 50\}$ and $m \in \{4, 7, 10\}$) provided by Ruiz-Torres et al. [2], and on 900 large-size instances ($n \in \{80, 100, 150\}$ and $m \in \{5, 10, 20\}$) randomly generated. The stopping criterion ($stopC$) was set at a predefined elapsed CPU time defined by $\frac{n}{m}$ seconds. t_{min} was set at 2. The following values were tested for t_{max}: $\lceil 0.7m \rceil$, $\lceil 0.8m \rceil$, $\lceil 0.9m \rceil$, $\lceil 1.0m \rceil$. The best value was $t_{max} = \lceil 0.9m \rceil$.

We compare the ILS-RVD with the Simulated Annealing algorithm SA* proposed in [2]. For the medium-size instances, the best results of SA* were made avalable by [2]. Following the original paper, we re-implemented the SA* algorithm. We name SA*2. In SA*2 we use the same stop condition, n/m seconds. We compare SA*2 with the original algorithm SA* on the 900 medium-size instances. The results show that SA*2 is significantly better than SA*. Since SA*2 and ILS-RVND use the same stop condition, in this paper we compare the performance of these algorithms. The algorithms were coded in C++ and all instances were solved 16 times by the algorithms. The performance of the algorithms are measured by computing the Relative Percentage Deviation (RPD) defined by: $RPD = \frac{f_{\text{Average}} - f_{\text{Best}}}{f_{\text{Best}}} \times 100\%$, where $f_{Average}$ is the average makespan value (among the 16 runs) obtained by a given algorithm and f_{Best} is the best known makespan. Table 1 reports the average RPDs of the algorithms, for medium and large instances. Bold values represent best RPDs. We can see that ILS-RVND produces the best average results in all instance groups. For medium-size instances, SA*2 and ILS-RVND present overall RPD means of 2.3% and 0.1%, respectively. For large-size instances, the overall RPD means of the algorithms SA*2 and ILS-RVND are 3.2% and 1.3%, respectively. The effectiveness of ILS-RVND was statistically validated.

Table 1: Average RPDs for the compared algorithms

$n \times m$	SA*2	ILS-RVND	$n \times m$	SA*2	ILS-RVND
20×4	1.3	**0.0**	80×5	1.0	**0.2**
20× 7	3.6	**0.0**	80×10	3.3	**1.1**
20×10	3.8	**0.0**	80×20	5.7	**2.2**
35×4	0.6	**0.0**	100×5	1.1	**0.2**
35×7	2.1	**0.1**	100×10	3.2	**1.2**
35×10	4.3	**0.3**	100×20	5.4	**2.4**
50×4	0.4	**0.0**	150×5	1.2	**0.3**
50×7	1.7	**0.2**	150×10	3.5	**1.1**
50×10	3.2	**0.2**	150×20	4.5	**2.6**
Mean	2.3	**0.1**	Mean	3.2	**1.3**

It can be concluded that the proposed algorithm can be considered the state-of-the-art method for solving the scheduling problem under study, as seen through a comparison of the obtained results with the best available meta-heuristic on a wide range of benchmark instances.

Acknowledgments

This work was supported by CAPES, CNPq and FAPEMIG.

4. REFERENCES

[1] H. R. Lourenço, O. C. Martin, and T. Stützle. *Iterated local search*. Springer, 2003.

[2] A. J. Ruiz-Torres, G. Paletta, and E. Pérez. Parallel machine scheduling to minimize the makespan with sequence dependent deteriorating effects. *Computers & Operations Research*, 40(8):2051–2061, 2013.

[3] A. Subramanian, E. Uchoa, and L. S. Ochi. A hybrid algorithm for a class of vehicle routing problems. *Computers & Operations Research*, 40(10):2519–2531, 2013.

[4] D.-L. Yang, T. Cheng, S.-J. Yang, and C.-J. Hsu. Unrelated parallel-machine scheduling with aging effects and multi-maintenance activities. *Computers & Operations Research*, 39(7):1458–1464, 2012.