Minimizing Total Tardiness on Identical Parallel Machines Using VNS with Learning Memory

Eduardo Lalla-Ruiz[1(✉)] and Stefan Voß[2]

[1] Department of Computer and Systems Engineering, University of La Laguna, San Cristóbal de La Laguna, Spain
elalla@ull.es

[2] Institute of Information Systems, University of Hamburg, Hamburg, Germany
stefan.voss@uni-hamburg.de

Abstract. Minimizing total tardiness on identical parallel machines is an $\mathcal{NP}$-hard parallel machine scheduling problem that has received much attention in literature due to its direct application to real-world applications. For solving this problem, we present a variable neighbourhood search that incorporates a learning mechanism for guiding the search. Computational results comparing with the best approaches for this problem reveals that our algorithm is a suitable alternative to efficiently solve this problem.

1 Introduction

Minimizing total tardiness on identical parallel machines (referred to as $P||\sum T_j$ in standard machine scheduling terminology) assumes a set of n jobs $J = \{1, ..., n\}$ to be processed on m identical parallel machines $M_1, ..., M_m$. Each job j has a processing time $p_j > 0$ and a due date d_j. All jobs are available at time zero, and no job pre-emption is allowed. The tardiness T_j of job j is given by $T_j = max(C_j - d_j, 0)$, where C_j is the completion time of j. The objective of this problem is to find a schedule that minimizes the total tardiness $\sum_{j=1}^{n} T_j$. Recent references include [1,2,6].

The $P||\sum T_j$ has been widely studied in the literature. Recent studies are the following. Biskup *et al.* [1] present a comparison of the existing heuristic algorithms for solving this problem. Moreover, they propose a new heuristic approach to solve the problem. This heuristic provides better results in terms of solution quality than the other heuristics. Tanaka and Araki [6] propose a new branch and bound algorithm with a lagrangian relaxation for computing the lower bounds. They propose a set of problem instances to assess their approach. Niu *et al.* [4] propose a Clonal Selection Particle Swarm Optimization (CSPSO) for this problem. The authors evaluate their proposal over the instances proposed in [6]. Deng *et al.* [2] propose a Hybrid Differential Evolution algorithm (HDDE) and also used the problem instances presented in [6]. The computational experience reported in their work shows that HDDE outperforms CSPSO for the

C. Dhaenens et al. (Eds.): LION 9 2015, LNCS 8994, pp. 119–124, 2015.
DOI: 10.1007/978-3-319-19084-6_10

small-sized problem instances. Moreover, they compare HDDE with branch and bound (B&B) for the large-sized instances and indicate that HDDE improves the performance of B&B in terms of computational time.

In this work, we present a Variable Neighbourhood Search with a learning mechanism (VNS-L) for solving $P||\sum T_j$. Our approach combines a variable neighbourhood search with restarting strategy which exploits the use of a memory for learning from past solutions. The goal of this work is to assess the performance of this idea as well as to provide high-quality solutions in short computational times. In doing so, the performance of the VNS-L is evaluated using the well-known benchmark suite proposed by Tanaka and Araki [6] and comparing its results with those given by some arguably best algorithms reported in the literature.

The remainder of this paper is organized as follows. Section 2 describes the VNS-L proposed to address $P||\sum T_j$. Afterwards, the performance of our algorithm is analyzed in realistic scenarios proposed in literature; see Sect. 3. Finally, Sect. 4 provides the main conclusions extracted from the work and suggests several directions for further research.

2 Variable Neighbourhood Search with Learning

Variable Neighbourhood Search (VNS) is a well-established meta-heuristic that systematically exploits the idea of changing neighbourhoods during the search [3]. VNS relies only on the best solutions currently known to center the search. In this regard, the information collected during the search relative to previous good solutions or their characteristics is forgotten. In order to address this deficiency and take advantage of that information, a Variable Neighbourhood Search with a Learning Mechanism is proposed. It is a hybridization within a Multi-Start strategy (MS) embedded with a learning mechanism for taking advantage of the information obtained during the VNS process using a memory structure. Within the VNS-L template we use the exploitation capabilities of the VNS and the exploration capabilities provided by MS as it gives the ability of re-starting the search. As noted by [5], incorporating memory structures into re-starting processes improves their performance.

The learning mechanism within VNS-L is based on (i) a frequency based memory structure, termed as M, with the aim of collecting promising solution features found during the search and (ii) a solution generation procedure. The memory structure, M, is composed of solution features as follows. Consider a set of solutions Λ. Each solution $x \in \Lambda$ has associated a set $C(x) = \{(i, j)\}$ of features. That is, a job j is the i-th job served in the whole schedule. The memory has a matrix structure with dimension $n \times n$, where the rows represent the jobs and the columns the service order. Each time the VNS improves the best solution known within the search, the information related to the solution features is updated, for keeping track of these matches. For example, in case the solution structure considered for updating the memory includes the feature (i, j), i.e., job j is the i-th job served in the schedule, then its corresponding memory position M_{ij} is updated.

Algorithm 1. VNS-L pseudocode

```
1  iter = 1
2  Initialize M
3  while iter ≠ iter_max do
4      x ← Solution Generation Procedure using M
5      k = 1
6      while k ≠ k_max do
7          Shaking:
8          Choose a random neighbour x' ∈ N_1(x, k)
9          Improvement phase:
10         while stopping criterion is not met do
11             a) Reinsertion move over x' → x''
12             b) Interchange move over x'' → x'''
13             if x''' is better than x' then
14                 x' ← x'''
15         Solution assessment:
16         if x''' is better than x then
17             x ← x'''
18         if x''' is better than x_best then
19             Update memory M using β parameter
20             x_best ← x'''
21             k = 1
22         else
23             Update memory M using γ parameter
24             k = k + 1
25     iter = iter + 1
26 return x_best
```

The way a memory position M_{ij} is updated is as follows: $M_{ij} = (M_{ij} + 1) \cdot \beta$. The parameter $\beta \geq 1$ is used so that when the memory is updated, those solution features that have been part of the best known solutions more often have greater significance. On the other hand, in VNS-L we keep track of the worst solution obtained during the local search process. In case we are not able to improve the disturbed solution, we update the memory according to $M_{ij} = (M_{ij} + 1) \cdot \gamma$. The parameter $\gamma < 1$ is used so that when the memory is updated, those solution features affected will have less significance.

VNS-L as shown in Algorithm 1 uses a finite set of neighbourhoods based on (a) reinsertion-move, $N_1(x, k)$, namely k jobs are removed from a machine m and reinserted in another machine m', where $m \neq m'$, and (b) interchange-move $N_2(x)$, that consists of exchanging a job j assigned to machine m with a job j' assigned to machine m', where $m \neq m'$. For any given k the application of these neighbourhood structures is performed sequentially, i.e., firstly the reinsertion-move is applied and thereafter the interchange-move. The shaking process allows

Table 1. Computational results for the 2250 instances proposed by [6]. Note that only the computational times (measured in seconds) are reported since all the approaches reach the optimal solutions

Instance		HDDE		B&B		VNS-L	
n	m	Avg. t(s.)	Max. t(s.)	Avg. t(s.)	Max. t(s.)	Avg. t(s.)	Max. t(s.)
20	2	0.01	0.30	0.43	1.41	0.06	0.12
	3	0.02	0.41	0.30	4.00	0.07	0.13
	4	0.03	1.14	0.15	4.03	0.08	0.13
	5	0.02	0.17	0.08	0.47	0.08	0.13
	6	0.02	0.38	0.05	0.36	0.09	0.16
	7	0.02	0.27	0.04	0.36	0.09	0.17
	8	0.02	0.63	0.03	0.28	0.10	0.18
	9	0.01	0.52	0.11	8.81	0.10	0.19
	10	0.01	0.88	0.03	0.33	0.12	0.18
25	2	0.02	0.45	1.16	13.47	0.12	0.22
	3	0.06	1.92	19.85	757.28	0.15	0.32
	4	0.08	1.22	47.47	4148.98	0.17	0.28
	5	0.13	5.97	14.15	1534.72	0.17	0.43
	6	0.12	4.00	0.37	27.22	0.24	0.45
	7	0.10	1.52	0.16	2.84	0.28	0.58
	8	0.18	13.78	0.11	0.89	0.30	0.54
	9	0.06	0.72	0.18	12.78	0.32	0.57
	10	0.03	0.28	0.07	2.81	0.33	0.59
		0.05	1.92	4.71	362.28	0.16	0.30

to escape from those local optima found along the search by using the reinsertion-move. Once the search process in the VNS-L is over, the information stored in M is used by the solution generation procedure for re-starting the VNS-L. To do so, a roulette wheel mechanism using the information stored in M is applied to generate a job order sequence. Then, the first job in that sequence will be assigned to the machine which adds the minimum tardiness completion time to the solution.

3 Computational Results

This section is devoted to present the computational experiments carried out in order to assess the performance of the proposed algorithm. For this purpose, we use a set of 2250 instances provided by [6]. All the computational experiments reported in this work are conducted on a computer equipped with an Intel 3.16 GHz and 4 GB of RAM. We run 20 executions of VNS-L with the following parameter values: $iter_{max} = 15$, $k_{max}= 3$, $\beta = 1.2$, $\gamma = 0.95$, and M initialized as the one-matrix.

Table 1 shows the average computational results provided by (i) the best approximate approach reported in the literature based on a Hybrid Discrete Differential Evolution Algorithm, HDDE [2], (ii) the best exact approach based on a Branch and Bound, B&B [6], and (iii) our VNS-L. HDDE and B&B were executed on an Intel 3.2 GHz with 512 MB of RAM by [2]. The first columns correspond to the sizes of the instance sets. Since all the sets are composed of 125 instances each and the optimal solution values are known, in the tables we only report the average computational time values since the three methods always obtain the optimal solution values.

As can be seen in the table, VNS-L maintains a consistent temporal performance during the search. VNS-L reduces the maximum required time in 85 % and 99.92 % of the cases in comparison to HDDE and B&B, respectively. In this regard, it should be noted that, on average, there is not much difference between the average and maximum running performance of VNS-L (about 0.15 s). This gives a sense of the temporal performance of VNS-L, which makes it suitable when addressing related problems, solving larger instances or tackling integrated problem schemes where this problem is involved.

4 Conclusions and Further Research

In this work, the problem of minimizing total tardiness on identical parallel machines ($P||\sum T_j$) has been addressed. In order to solve it, a Variable Neighbourhood Search with a Learning mechanism (VNS-L) is proposed. According to our computational experience over a well-known set of instances proposed in the literature, our algorithm shows a competitive performance in terms of solution quality and computational time. In this regard, it exhibits a similar performance by means of average and maximum computational time, which makes it suitable for those environments where the expected computational time may not vary from one instance to another within the same scenario dimensions.

As further research, we are going to assess the contribution of VNS-L for different configurations of its learning parameters as well as to determine an adaptive method to parameterize it according to given problem instances.

Acknowledgements. This work has been partially funded by the European Regional Development Fund, the Spanish Ministry of Economy and Competitiveness (project TIN2012-32608). Eduardo Lalla-Ruiz thanks the Canary Government for the financial support he receives through his doctoral grant.

References

1. Biskup, D., Herrmann, J., Gupta, J.N.D.: Scheduling identical parallel machines to minimize total tardiness. Int. J. Prod. Econ. **115**(1), 134–142 (2008)
2. Deng, G., Zhang, K., Gu, X.: A hybrid discrete differential evolution algorithm to minimise total tardiness on identical parallel machines. Int. J. Comput. Integr. Manuf. **26**(6), 504–512 (2013)

3. Hansen, P., Mladenovic, N., Moreno Pérez, J.A.: Variable neighbourhood search. Ann. Oper. Res. **175**, 367–407 (2010)
4. Niu, Q., Zhou, T., Wang, L.: A hybrid particle swarm optimization for parallel machine total tardiness scheduling. Int. J. Adv. Manuf. Technol. **49**(5–8), 723–739 (2010)
5. Stützle, T.: Local search algorithms for combinatorial problems. Darmstadt University of Technology. Ph.D. thesis (1998)
6. Tanaka, S., Araki, M.: A branch-and-bound algorithm with lagrangian relaxation to minimize total tardiness on identical parallel machines. Int. J. Prod. Econ. **113**(1), 446–458 (2008)