Hybrid Teaching-Learning Algorithm for Parallel Machine Scheduling with Tardiness and Energy Consumption Cost Optimization: Postprint

Authors: Wang Yongqi, Wu Fei, Jiang Xiaoxiao, Wang Chunyuan

Date: 2018-05-18T00:00:00+00:00

Abstract

For parallel machine scheduling with controllable processing times, this paper proposes a class of scheduling problems considering the optimization of tardiness and energy consumption costs. First, the scheduling problem is described, and an integer linear programming model is established for solution by CPLEX. To rapidly obtain satisfactory solutions for the problem, a hybrid teaching-learning algorithm is proposed. Based on the properties of the problem, encoding and decoding methods are designed to overcome the disadvantage that the standard teaching-learning algorithm cannot be directly applied to discrete problems. Simultaneously, a local search operator based on variable neighborhood search is constructed to enhance the search performance of the hybrid algorithm. Finally, simulation experiments are conducted on the parallel machine scheduling problem with controllable processing times, and the test results verify the feasibility and effectiveness of the integer linear programming model and hybrid algorithm constructed in this paper.

Full Text

Preamble

Hybrid Teaching-Learning-Based Optimization Algorithm for Optimizing Tardiness and Energy Cost on Parallel Machine Scheduling

Wang Yongqi, Wu Fei, Jiang Xiaoxiao, Wang Chunyuan
(School of Electronic & Electrical Engineering, Shanghai University of Engineering Science, Shanghai 201620, China)

Abstract: This paper addresses a parallel machine scheduling problem with controllable processing times (PMS-CPT) that optimizes tardiness and energy

costs. First, we formulate the scheduling problem and develop an integer linear programming (ILP) model for solution by CPLEX. To rapidly obtain satisfactory solutions, we propose a hybrid teaching-learning-based optimization (HTLBO) algorithm. Leveraging problem characteristics, we design novel encoding and decoding schemes to overcome the limitation of the standard teaching-learning-based optimization algorithm in handling discrete problems. Additionally, we construct a local search operator based on variable neighborhood search (VNS) to enhance the hybrid algorithm' s search capability. Finally, simulation experiments on PMS-CPT problems validate the feasibility and effectiveness of the proposed ILP model and hybrid algorithm.

Keywords: parallel machine scheduling; tardiness; energy consumption; controllable processing times; teaching-learning-based optimization algorithm

0 Introduction

Parallel machine scheduling (PMS) represents a typical class of scheduling problems in modern manufacturing with broad applications in cloud computing, semiconductor fabrication, and automotive production [1]. Traditional PMS research focuses on time-related objectives such as makespan and earliness/tardiness. However, with the rapid development of customized production and rising global energy costs, just-in-time and energy-efficient scheduling have become critical aspects of shop floor management [2-4]. Consequently, research on parallel machine scheduling for tardiness and energy cost optimization holds significant theoretical importance and practical value.

The PMS problem is NP-hard, making the design of fast and efficient algorithms a key research focus. Existing solution methods fall into three categories: exact algorithms, heuristic algorithms, and intelligent algorithms [5]. Exact methods such as dynamic programming and branch-and-bound can obtain optimal solutions but suffer from exponential time complexity as problem size increases. Heuristic algorithms employ scheduling rules to allocate and sequence jobs across machines, yielding near-optimal solutions quickly but with strict limitations on problem settings. Recently, intelligent algorithms—including genetic algorithms, tabu search, particle swarm optimization, and simulated annealing—have been applied to PMS problems, delivering satisfactory solutions within reasonable timeframes [6]. However, these algorithms often suffer from numerous control parameters, slow convergence, and premature convergence to local optima.

The teaching-learning-based optimization (TLBO) algorithm, proposed by Rao et al., is a novel swarm intelligence method that simulates the "teaching" and "learning" processes in classroom instruction [7]. The algorithm comprises a "teacher phase" and a "learner phase," offering advantages such as few control parameters and fast convergence. TLBO has been successfully applied to complex function optimization, neural network training, and resource-constrained

project scheduling [8-11], yet few studies have applied it to parallel machine scheduling problems.

Building upon these foundations, this paper proposes a scheduling problem for parallel machines with controllable processing times that considers tardiness and energy cost optimization. We develop an integer linear programming model for CPLEX solution and design a hybrid teaching-learning algorithm to rapidly obtain satisfactory solutions for medium- and large-scale problems. Simulation tests validate the feasibility and effectiveness of both the ILP model and the hybrid optimization algorithm.

1.1 Problem Description

Consider a system with m machines processing n orders. Each order has a specific due date and unit-time tardiness penalty cost. Each machine offers multiple processing speeds with different energy consumption costs per unit time. The optimization objective is to minimize the total cost comprising tardiness penalties and energy consumption. The decision variables involve three aspects: (1) assignment of orders to machines, (2) processing sequence of orders on each machine, and (3) processing speed selection for orders on each machine.

1.2 Mathematical Model

Based on the problem description, we define the following notation for the integer linear programming model of the PMS-CPT problem:

Sets and Parameters: - m: Total number of machines, indexed by i ($1 \leq i \leq m$) - n: Total number of orders, indexed by j ($1 \leq j \leq n$) - l: Processing position index on machines ($1 \leq l \leq n$) - u_i: Number of processing speeds for machine i, indexed by s ($1 \leq s \leq u_i$) - P_j: Workload of order j - d_j: Due date of order j - C_j: Completion time of order j - T_j: Tardiness of order j, i.e., $T_j = \max\{0, C_j - d_j\}$ - ω_j: Unit-time tardiness cost for order j - S_{is}: Value of the s-th processing speed of machine i - e_{is}: Energy cost per unit time when machine i operates at speed s - t_{is}: Processing time of order j on machine i at speed s, i.e., $t_{is} = P_j/S_{is}$ - b_{il}: Start time of the order at position l on machine i

Decision Variables: - x_{jils}: Binary variable that equals 1 if order j is processed at position l on machine i at speed s, and 0 otherwise - st_{il}: Start time of the order at position l on machine i

Mathematical Formulation:

Objective Function:

$$\min F = \sum_{j=1}^{n} \omega_j T_j + \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{l=1}^{n} \sum_{s=1}^{u_i} e_{is} \cdot \frac{P_j}{S_{is}} \cdot x_{jils} \tag{1}$$

Constraints:

$$\sum_{i=1}^{m} \sum_{l=1}^{n} \sum_{s=1}^{u_i} x_{jils} = 1, \quad \forall j \in \{1, 2, \ldots, n\} \tag{2}$$

$$\sum_{j=1}^{n} x_{jils} \leq 1, \quad \forall i \in \{1, 2, \ldots, m\}, l \in \{1, 2, \ldots, n\} \tag{3}$$

$$b_{i,l+1} = \sum_{j=1}^{n} \sum_{s=1}^{u_i} (b_{il} + \frac{P_j}{S_{is}}) \cdot x_{jils}, \quad \forall i \in \{1, 2, \ldots, m\}, l \in \{1, 2, \ldots, n-1\} \tag{4}$$

$$T_j \geq b_{il} + \frac{P_j}{S_{is}} \cdot x_{jils} - d_j - M \cdot (1 - x_{jils}), \quad \forall i, j, l, s \tag{5}$$

$$T_j \geq 0, \quad \forall j \in \{1, 2, \ldots, n\} \tag{6}$$

$$x_{jils} \in \{0, 1\}, \quad \forall i, j, l, s \tag{7}$$

Equation (1) defines the objective function comprising two cost components: total tardiness penalty costs and total processing energy costs. Constraint (2) ensures each order is assigned to exactly one position on one machine at a specific speed. Constraint (3) limits each position on a machine to at most one order. Constraint (4) calculates the start time of the order at position $l + 1$ on machine i. Constraints (5) and (6) compute the tardiness T_j for order j. Equation (7) specifies the binary domain of the decision variables.

---

2 Standard Teaching-Learning Optimization Algorithm

The TLBO algorithm uses a population to represent solutions to the problem, where the best individual in the current population is called the teacher and the remaining individuals are students. The algorithm' s neighborhood search comprises two phases: a teacher phase and a learner phase. In the teacher phase, the algorithm uses the teacher individual and the mean of the population to instruct student individuals, thereby evolving the population. In the learner phase, the algorithm randomly selects two individuals from the current population and employs a learning strategy where the current individual learns from the better one. The standard TLBO algorithm proceeds as follows:

a) Initialization: Set TLBO control parameters and problem parameters, including population size (number of students P_n), problem dimension D, and variable bounds $[x_l, x_u]$ for each dimension.

b) Population Construction: Generate the initial population randomly and evaluate each individual' s fitness. The initialization formula is:

$$x = x_l + \text{rand}(0,1) \cdot (x_u - x_l)$$

where $\text{rand}(0,1)$ is a random number uniformly distributed in $[0,1]$.

c) Teacher Phase: First, compute the mean of all individuals across each dimension to obtain the mean individual $\bar{x}$. Then, for each individual x_i in the current population, generate a new individual x_i' using the update formula. Finally, compare the fitness of the original and new individuals, retaining the better one via a greedy selection strategy. The update formula is:

$$x_i' = x_i + \text{rand}(0,1) \cdot (x_{\text{teacher}} - T_F \cdot \bar{x})$$

where x_{teacher} is the best solution in the current population (the teacher) and T_F is a teaching factor that randomly takes integer values from the set $\{1,2\}$.

d) Learner Phase: Randomly select two individuals x_1 and x_2 from the population after the teacher phase, and have them learn from the better individual. Let $f(\cdot)$ denote the fitness function (assuming a minimization problem). The update formula is:

$$x_i' = \begin{cases} x_i + \text{rand}(0,1) \cdot (x_1 - x_2) & \text{if } f(x_1) < f(x_2) \\ x_i + \text{rand}(0,1) \cdot (x_2 - x_1) & \text{otherwise} \end{cases}$$

e) Termination Check: If the termination condition is met (e.g., maximum iterations or computational time limit), stop TLBO and output the current best solution; otherwise, return to step c).

3.1 Encoding and Decoding

The standard TLBO algorithm employs real-valued encoding, which cannot be directly applied to discrete problems. Therefore, we redesign the encoding and decoding schemes based on the characteristics of the PMS-CPT problem.

Given problem parameters—machine count m, order count n, and processing speed count u_i for each machine—the parallel machine scheduling problem can be encoded using an n-dimensional array $x = \{x_1, x_2, \ldots, x_n\}$, where each dimension' s value ranges in $[1, m+1)$. The j-th dimension corresponds to order j. The integer part of the j-th dimension encodes the assigned machine number, while the decimal part determines the processing speed. The first decimal digit selects the speed, and the remaining digits determine the processing sequence on the machine.

Decoding Process: 1. Take the integer part of each dimension to determine machine assignments. 2. Extract the first decimal digit and construct a roulette

wheel probability model for each machine to select processing speeds. For machine i with u_i speed types, if the encoded value falls in interval $[(s-1)/u_i, s/u_i)$, the processing speed is set to S_{is} $(1 \leq s \leq u_i)$. 3. Sort orders assigned to the same machine in ascending order based on the remaining decimal digits to determine the processing sequence.

[Figure 1: see original paper] illustrates the encoding and decoding scheme. For the example with $m = 2$, $n = 5$, and $u_i = 3$, the encoding range is $[1, 3)$. The integer part represents order-to-machine assignment, the first decimal digit selects processing speed, and the remaining digits determine processing order. Given the initial encoding $\{2.629, 2.812, 1.254, 2.827, 2.265, 1.195\}$, the decoding yields: - Order 1: Machine 2, Speed 3 - Order 2: Machine 2, Speed 3 - Order 3: Machine 1, Speed 1 - Order 4: Machine 2, Speed 3 - Order 5: Machine 1, Speed 2 - Machine 1 processing sequence: $\{3, 5\}$ - Machine 2 processing sequence: $\{2, 4, 1\}$

3.2 Local Search Operator Based on VNS

Similar to other swarm intelligence algorithms, the standard TLBO algorithm tends to fall into local optima in later iterations, limiting its search performance. To address this, we construct a local search operator based on Variable Neighborhood Search (VNS) to enhance the algorithm' s exploration capability.

VNS systematically changes a set of neighborhood structures to expand the search scope and obtain local optima. From these local optima, it again systematically varies the neighborhood structures to explore further and identify potentially better local optima [12]. The core of the VNS algorithm lies in defining the neighborhood structure and search procedure.

Given a maximum iteration count K, the VNS-based local search operator is defined as follows:

a) Initialization: Set $k = 1$.

b) Shaking: Generate a random solution x' from the k-th neighborhood of the current solution x.

c) Local Search: Apply a local search method to x' to obtain a local optimum x''.

d) Update: If $f(x'') \leq f(x)$, set $x \leftarrow x''$ and $k \leftarrow 1$; otherwise, set $k \leftarrow k + 1$.

e) Termination Check: If $k \leq K$, return to step b); otherwise, terminate the VNS search.

The neighborhood transformation formula in step d) is defined as:

$$x' = x + \text{sign} \cdot \text{rand} \cdot \beta \cdot d$$

where sign is a random integer from $\{-1, 1\}$, β is a neighborhood control parameter, and d is a model parameter (i.e., the number of machines).

3.3 Hybrid Teaching-Learning Algorithm Flow

Integrating the above components, the HTLBO algorithm for solving the PMS-CPT problem proceeds as follows:

a) Initialization: Set algorithm parameters including population size and termination criteria.

b) Initial Population: Generate the initial population randomly and evaluate fitness values.

c) Teacher Phase: Compute the mean individual of the current population and update the population using the teacher phase update formula.

d) Learner Phase: Update the population using the learner phase update formula.

e) VNS Local Search: Apply the VNS-based local search operator to individuals that failed to improve in the teacher and learner phases.

f) Termination: If the termination condition is met, stop and output the best solution; otherwise, return to step c).

4.1 Example Design

As no existing benchmark datasets match our proposed problem, we generate test instances following Hall and Posner' s methodology for machine scheduling problems [13]. The test suite comprises 21 instances categorized as small-scale and medium-to-large-scale. Small-scale instances can be solved exactly by CPLEX to evaluate the HTLBO algorithm' s performance, while medium-to-large-scale instances require comparison with other intelligent algorithms.

Small-scale instances: $m \in \{3, 4, 5\}$, $n \in \{5, 10, 15\}$
Medium-to-large-scale instances: $m \in \{3, 4, 5\}$, $n \in \{30, 50, 80, 100\}$

For each instance (m, n), parameters are generated as follows:

a) Order workload $P_j \sim U(1, 100)$; unit-time tardiness cost coefficient $\omega_j \sim U(0, 1)$

b) Number of processing speed types $u_i = 5$; speed values $S_{is} \sim U(5, 10)$

c) Energy cost per unit time $e_{is} \sim U(0, 1)$, where lower speeds correspond to higher e_{is} values

d) Due date $d_j \sim U(0.5D, 2D)$, where D is calculated as:

$$D = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{s=1}^{u_i} P_j}{m \cdot \sum_{i=1}^{m} u_i}$$

All algorithms were implemented in MATLAB 2016a on a computer with an Intel Core i5-8250U CPU (1.6 GHz) and 8 GB RAM. The ILP model was solved via the CPLEX-MATLAB interface. For each test instance, the HTLBO algorithm performed 30 independent runs. For small-scale instances, population size was set to 30, maximum computation time to 10s, $K = 5$, and $\beta = 0.2$. For medium-to-large-scale instances, population size was 80, maximum computation time 150s, $K = 15$, and $\beta = 0.1$.

4.2 Test Results

Small-scale Instances: Table 1 compares CPLEX and HTLBO results. F^* denotes the exact solution from CPLEX with its computation time. F_b and F_w represent the best and worst solutions from 30 HTLBO runs, while P_b indicates the percentage of runs achieving the optimal solution.

The results show that CPLEX obtains exact solutions using our ILP model, validating its effectiveness. However, CPLEX computation time grows dramatically, reaching 2815s for instance (5,15). HTLBO achieves optimal solutions ($F_b = F^*$) for all small-scale instances, with success rates $P_b \geq 83.74\%$. The worst solutions (F_w) remain close to optimal. Given HTLBO' s computation time of only 10s—far less than CPLEX—these results confirm HTLBO' s effectiveness for small-scale problems.

Medium-to-large-scale Instances: Since CPLEX cannot obtain exact solutions within reasonable time, we compare HTLBO with standard TLBO and particle swarm optimization (PSO) [14]. Each algorithm ran 30 times per instance. Table 2 reports mean values ($\bar{F}$), standard deviations (σ), and relative mean deviation percentage (κ). For each instance, let F' be the best solution among all 90 runs; κ is calculated as:

$$\kappa = \frac{\bar{F} - F'}{F'} \times 100\%$$

The results demonstrate that HTLBO, enhanced with VNS-based local search, outperforms both TLBO and PSO in terms of mean solution quality ($\bar{F}$) and relative deviation (κ). Moreover, HTLBO's smaller standard deviations indicate stronger robustness and more stable performance.

---

5 Conclusion

This paper investigates a parallel machine scheduling problem with controllable processing times, optimizing tardiness and energy costs. We develop an integer linear programming model for exact solution of small-scale problems via CPLEX and design a hybrid teaching-learning algorithm for rapid solution of medium- and large-scale problems. Simulation tests on 21 instances demonstrate that both CPLEX and HTLBO achieve optimal solutions for small-scale problems, with HTLBO requiring significantly less computation time. For medium-to-large-scale problems, HTLBO exhibits superior performance and stronger robustness compared to standard TLBO and PSO algorithms.

References

[1] Cheng T C E, Sin C C S. A state-of-the-art review of parallel-machine scheduling research [J]. European Journal of Operational Research, 1990, 47(3): 271-292.

[2] Adamu M O, Abass O. Parallel machine scheduling to maximize the weighted number of just-in-time jobs [J]. Journal of Applied Science and Technology, 2010, 15(1-2).

[3] Yin Y, Cheng S R, Cheng T C E, et al. Just-in-time scheduling with two competing agents on unrelated parallel machines [J]. Omega, 2016, 63: 41-47.

[4] Li K, Zhang X, Leung J Y T, et al. Parallel machine scheduling problems in green manufacturing industry [J]. Journal of Manufacturing Systems, 2016, 38: 98-106.

[5] Lin Y K, Pfund M E, Fowler J W. Heuristics for minimizing regular performance measures in unrelated parallel machine scheduling problems [J]. Computers & Operations Research, 2011, 38(6): 901-916.

[6] Balin S. Non-identical parallel machine scheduling using genetic algorithm [J]. Expert Systems with Applications, 2011, 38(6): 6814-6821.

[7] Rao R V, Savsani V J, Vakharia D P. Teaching–learning-based optimization: an optimization method for continuous non-linear large scale problems [J]. Information Sciences, 2012, 183(1): 1-15.

[8] Rao R V, Patel V. An improved teaching-learning-based optimization algorithm for solving unconstrained optimization problems [J]. Scientia Iranica, 2013, 20(3): 710-720.

[9] Gao Liqun, Ouyang Haibin, Kong Xiangyong, et al. Teaching-learning-based optimization algorithm with crossover operation [J]. Journal of Northeastern University: Natural Science, 2014, 35(3): 323-327.

[10] Yu Kunjie, Wang Xin, Wang Zhenlei. Elite teaching-learning-based optimization algorithm based on feedback [J]. Acta Automatica Sinica, 2014, 40(9): 1976-1983.

[11] Tuo Shouheng, Yong Longquan, Deng Fang'an. Survey on teaching-learning-based optimization algorithm [J]. Application Research of Computers, 2013, 30(7): 1933-1938.

[12] Hemmelmayr V C, Doerner K F, Hartl R F. A variable neighborhood search heuristic for periodic routing problems [J]. European Journal of Operational Research, 2009, 195(3): 791-802.

[13] Hall N G, Posner M E. Generating experimental data for computational testing with machine scheduling applications [J]. Operations Research, 2001, 49(6): 854-865.

[14] AlRashidi M R, El-Hawary M E. A survey of particle swarm optimization applications in electric power systems [J]. IEEE Trans on Evolutionary Computation, 2009, 13(4): 913-918.

Note: Figure translations are in progress. See original paper for figures.