Computers & Operations Research 39 (2012) 1315–1324

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

Minimizing mean weighted tardiness in unrelated parallel machine scheduling with reinforcement learning

Zhicong Zhang [a,1], Li Zheng [b,*], Na Li [c,2], Weiping Wang [d,3], Shouyan Zhong [d,4], Kaishun Hu [a,5]

[a] Department of Industrial Engineering, School of Mechanical Engineering, Dongguan University of Technology, Songshan Lake District, Dongguan 523808, Guangdong Province, China
[b] Department of Industrial Engineering, Tsinghua University, Beijing 100084, China
[c] Department of Industrial Engineering, Shanghai Jiao Tong University, Shanghai 200240, China
[d] School of Mechanical Engineering, Dongguan University of Technology, China

ARTICLE INFO

Available online 2 August 2011

Keywords:
Scheduling
Unrelated parallel machines
Reinforcement learning
Tardiness

ABSTRACT

We address an unrelated parallel machine scheduling problem with R-learning, an average-reward reinforcement learning (RL) method. Different types of jobs dynamically arrive in independent Poisson processes. Thus the arrival time and the due date of each job are stochastic. We convert the scheduling problems into RL problems by constructing elaborate state features, actions, and the reward function. The state features and actions are defined fully utilizing prior domain knowledge. Minimizing the reward per decision time step is equivalent to minimizing the schedule objective, i.e. mean weighted tardiness. We apply an on-line R-learning algorithm with function approximation to solve the RL problems. Computational experiments demonstrate that R-learning learns an optimal or near-optimal policy in a dynamic environment from experience and outperforms four effective heuristic priority rules (i.e. WSPT, WMDD, ATC and WCOVERT) in all test problems.

1. Introduction

Study on parallel machine scheduling mainly uses two sorts of criteria, one is completion time related criteria, e.g. Weng et al. [39]; the other is tardiness or earliness related criteria. Most parallel machine scheduling problems with tardiness criteria are known to be NP hard. Identical parallel machines are the simplest form of parallel machines. Methodology applied in identical parallel machine scheduling with tardiness criterion includes approximation algorithms [22,13,40], dynamic programming [32], branch and bound algorithms [41], heuristic methods, etc. Heuristic methods include the modified due date (MDD) rule [1], minimum penalty increase assignment (MPA) rule, start-time decision (SD) rule [35] and local search methods, such as genetic algorithm [8], simulated annealing [35] and Tabu Search [2]. Setup time is considered in some research. For example, Sivrikaya-Serifoglu and Ulusoy [31] and Kim et al. [21] considered sequence-dependent setup time, Yi and Wang [42] considered batch setup times, and Eom et al. [15] considered family setup times.

Unrelated parallel machines are a more complicated form of machines than identical parallel machines. Bank and Werner [5] addressed the problem of minimizing the weighted sum of linear earliness and tardiness penalties in unrelated parallel machine scheduling. They derived some structural properties useful to searching for an approximate solution and proposed various constructive and iterative heuristic algorithms. Liaw et al. [23] found the efficient lower and upper bounds of minimizing the total weighted tardiness by a two-phase heuristic and presented a branch-and-bound algorithm incorporating various dominance rules. Kim et al. [20] studied batch scheduling of unrelated parallel machines with a total weighted tardiness objective considering setup times. They examined four search heuristics for this problem: the earliest weighted due date rule, the weighted shortest processing time (WSPT) rule, the two-level batch scheduling heuristic, and the simulated annealing method.

Priority rules are widely applied to production scheduling with tardiness criteria. earliest due date (EDD), shortest processing time (SPT), critical ratio (CR) and minimal slack (MS) are four simple rules for this area. WSPT rule usually performs reasonably with heavy load and tight due date allowance. Rachamadugu and Morton [27] developed a look-ahead rule for the single-machine weighted-tardiness problem called the Apparent Tardiness Cost

* Corresponding author. Tel.: +86 10 62785584; fax: +86 10 62794399.
E-mail addresses: stephen1998@gmail.com (Z. Zhang), lzheng@mail.tsinghua.edu.cn (L. Zheng), na-li03@sjtu.edu.cn (N. Li), wangwp@dgut.edu.cn (W. Wang), zhongsy@dgut.edu.cn (S. Zhong), huks@dgut.edu.cn (K. Hu).
[1] Tel.: +86 769 22256633; fax: +86 769 22861122.
[2] Tel.: +86 21 34206740; fax: +86 21 34206477.
[3] Tel./fax: +86 769 22861188.
[4] Tel.: +86 769 22861280; fax: +86 769 22861122.
[5] Tel./fax: +86 769 22861122.

0305-0548/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2011.07.019

(ATC) rule. Volgenant and Teerhuis [37] showed that the ATC rule outperformed EDD and WSPT in single-machine weighted tardiness problems. Carroll [7] proposed the original COVERT rule for minimizing average tardiness and Vepsalainen and Morton [36] extended it to the weighted COVERT rule for coping with mean weighted tardiness criterion, as we call WCOVERT in this paper. Russell et al. [28] conducted comparative analysis of the COVERT rule within the context of machine-constrained job shop and varying degree of due-date tightness. The results showed that COVERT outperforms the truncated SPT rule, the dynamic slack rule and MDD rule in most instances. ATC and WCOVERT integrate many existing rules and extend them with 'looking ahead' information. Vepsalainen and Morton [36] tested several dispatching rules, e.g. First Come First Served (FCFS), EDD, Slack per Remaining Processing Time (S/RPT), WSPT, ATC and WCOVERT, with job shop problems with weighted tardiness criteria. The results indicated that ATC and WCOVERT are superior to the other rules. Baker and Bertrand [4] developed the MDD rule for minimizing unweighted tardiness. Kanet and Li [19] generalized MDD to Weighted Modified Due Date (WMDD) to deal with weighted tardiness criteria and showed that WMDD, ATC and WCOVERT give similar results in many test cases.

In most of the previous studies, the release dates of all jobs are known and the assumption of a common due date is made. In this paper, we address a dynamic unrelated parallel machine scheduling problem with a mean weighted tardiness objective. There are n types of jobs and they dynamically arrive in independent Poisson processes. Therefore, the arrival time and the due date of each job are stochastic. We apply a reinforcement learning (RL) method to solve this problem. We introduce basic concepts of reinforcement learning and its application in Section 2, formulate the unrelated parallel machine scheduling problem in Section 3, present the detailed R-learning algorithm in Section 4, conduct computational experiments in Section 5 and draw conclusions in Section 6.

2. Reinforcement learning

Reinforcement learning is a machine learning method which is widely used in the area of artificial intelligence. Reinforcement learning is a model in which an agent learns to select optimal or near-optimal actions for achieving its long-term goals through trial-and-error interactions with dynamic environment. In this paper we address RL problems of undiscounted continuing tasks. Sutton and Barto [33] defined four key elements of RL methods: a policy, a reward function, a value function and a model of the environment. A policy specifies the agent's action in each state. A reward function specifies the payment on transition from one state to another. A value function specifies the value of a state or a state-action pair indicating its desirability in the long run. It is the expected average reward for each decision time step. Given the current state and a specific action, a model of the environment predicts the next state and the next reward with probability. RL methods aim to find the optimal or a near-optimal policy that maximizes the expected value of every state or state-action pair by learning from interactions between the agent and its environment.

Specifically, the agent and the environment interact at each decision time step. The agent perceives the current state s_q of the environment at decision time step q and then selects an action $a_t \in A(s_q)$ to perform according to the value of s_q, $V(s_q)$, or the value of state-action pair (s_q,a_q), $Q(s_q,a_q)$, following a specific policy π_q, where $A(s_q)$ denotes the set of available actions at state s_q. The action causes a change in the environment. In the next decision time step, the environment transfers into a new state s_{q+1} and the agent receives an immediate reward r_{q+1} that is used to pay for or penalize the selected action. Then by iterations state values or state-action values are updated towards the optimal ones and the policy is improved using the rewards. These interactions between the agent and its environment continue until the agent learns a policy that maximizes the average reward R, defined as follows:

$$R = \lim_{u\to\infty} \sum_{q=1}^{u} r_q/u, \tag{1}$$

where u is the number of decision time steps.

Schwartz [29] proposed the original R-learning algorithm, an average-reward RL algorithm. Tabular R-learning updates action values using the following formula:

$$Q(s,a) = Q(s,a) + \alpha[r - \rho + \max_{a'} Q(s',a') - Q(s,a)], \tag{2}$$

where α $(0 < \alpha \le 1)$ is the learning rate, r is the immediate reward and ρ is an approximation of R^π. R^π is the value function of policy π, i.e. the expected reward per decision time step under policy π.

Mahadevan [24] carried out detailed sensitivity analysis of R-learning and suggested that R-learning is quite sensitive to exploration strategies. Das et al. [12] presented another average-reward RL algorithm called Semi-Markov Average Reward Technique (SMART) and applied it to inventory control. Gosavi [17] extended the SMART to relaxed-SMART and proved its convergence. All the above algorithms are based on value iteration. Gosavi [16] proposed a policy iteration based RL algorithm for average reward, which was proved to converge and obtain optimal solutions under specific conditions. So far, most of convergence analysis assumes that the average-reward RL algorithms are in a tabular form. The above methods are model free. Tadepalli and Ok [34] proposed a model-based average-reward RL algorithm, H-learning and demonstrated that it converged quickly and robustly. They employed local linear regression for value function approximation and applied it to AGV scheduling. Paternina-Arboleda and Das [26] used an average-reward RL algorithm to develop a production control policy. They compared it with the existing policies on the basis of total average Work In Process (WIP) and average cost of WIP.

In recent years, RL algorithms have been applied to production scheduling. Their application covers scheduling in some basic forms of production systems, such as flow shop [6,9], job shop [3,11], JIT manufacturing systems [18] and Flexible Manufacturing Systems(FMS). Q-learning and TD(λ) are the two primary RL methods used in production scheduling. Wang and Usher [38] applied Q-Learning to a single machine dispatching rule selection problem to learn the commonly accepted dispatching rules. Singh et al. [30] proposed a policy gradient method for SMDP with application to call admission control. Csáji et al. [10] presented an adaptive iterative distributed scheduling algorithm that operated in a market-based production control system, where each machine and each job was associated with its own software agent. So far, average-reward RL algorithms are seldom applied to production scheduling in literature.

Reinforcement learning makes a basic assumption that the decision-making problem has Markovian or semi-Markovian nature. Therefore, the decision and the state or state-action value are assumed to be a function only of the current state. Moreover, given the current state and the chosen action, the next state and the reward can be predicted. Under this assumption, the interaction of an agent and the environment is a Markov Decision Process (MDP) or a Semi-Markov Decision Process (SMDP). Fortunately, reinforcement learning is still applicable by constructing approximate Markov states even when the decision-making problem is non-Markov and the information about the environment is incomplete.