Operations Research Letters 45 (2017) 442–447

Contents lists available at ScienceDirect

Operations Research Letters

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

Approximate policy iteration for dynamic resource-constrained project scheduling

Mahshid Salemi Parizi [a], Yasin Gocgun [b], Archis Ghate [a,*]

[a] Industrial & Systems Engineering, University of Washington, Seattle, USA
[b] Department of Industrial Engineering, Altinbas University, Istanbul, Turkey

ARTICLE INFO

Article history:
Received 15 January 2017
Received in revised form 14 June 2017
Accepted 14 June 2017
Available online 11 July 2017

Keywords:
Markov decision processes
Approximate dynamic programming
Queueing

ABSTRACT

We study non-preemptive scheduling problems where heterogeneous projects stochastically arrive over time. The projects include precedence-constrained tasks that require multiple resources. Incomplete projects are held in queues. When a queue is full, an arriving project must be rejected. The goal is to choose which tasks to start in each time-slot to maximize the infinite-horizon discounted expected profit. We provide a weakly coupled Markov decision process (MDP) formulation and apply a simulation-based approximate policy iteration method. Extensive numerical results are presented.

1. Introduction and literature review

Mathematical studies on project scheduling date at least as far back as the Critical Path Method (CPM) and the Program Evaluation and Review Technique (PERT) [14]. Both CPM and PERT assume unlimited resource availability. Early work on resource-constrained project scheduling problems (RCPSPs) can perhaps be traced to the zero–one programming model in [16]. That paper studied a discrete-time problem and incorporated multiple projects with precedence-constrained tasks and multiple resources. Task durations were deterministic and new project arrivals were not allowed. The binary decision variables corresponded to whether or not a task is completed in a certain period. Such static-deterministic RCPSPs started receiving more attention in the 1980s. For instance, Blazewicz et al. [4] showed in 1983 that such RCPSPs are NP-hard. More recent surveys of static-deterministic RCPSPs are available in Herroelen et al. [10], and in Hartmann and Briskorn [9]. Hartmann and Briskorn stated that most of the literature available at the time focused on rather simplistic models and ignored two features that are important in practice: stochastic task durations and dynamic arrivals of new projects. One of the appropriate models for incorporating such stochastic and dynamic components is Markov decision processes (MDPs). We therefore review literature that uses MDPs to model project scheduling problems next.

Choi et al. [5] formulated an infinite-horizon, discrete-time MDP model for RCPSPs with stochastic task durations, uncertain task outcomes (success or failure), and uncertain costs. In their problem setting, only one resource was needed to perform one task. They focused on linear activity-on-node (that is, precedence constrained) networks and did not model dynamic arrivals of new projects. The state in their MDP included the status of each project, that is, which tasks are complete and which are ongoing; information about whether or not the latest task in each project was a success was also stored in the state; in addition, the state included the number of time-slots for which a resource has been used for the currently ongoing task. Since their activity-on-node network was linear, at most one task in a project can be started in one time-period. The decisions in their MDP were therefore binary, representing whether or not to start a particular task in a particular project. The resulting MDP still suffered from the curse of dimensionality. A simulation-based approximate dynamic programming (ADP) algorithm was applied.

Choi et al. [6] extended the above MDP by allowing new project arrivals from a pre-determined group of projects. Consequently, their state included an additional variable to represent the realized arrival time of each new project. The decisions were identical to those in Choi et al. [5]. They implemented a variation of the Q-learning algorithm [17] on the resulting large-scale MDP.

Melchiors [11] formulated an infinite-horizon, continuous-time MDP model for scheduling dynamically arriving projects with stochastic task durations. Project arrivals followed a Poisson process. Their model was not restricted to linear activity-on-node networks. However, as in the above two papers by Choi et al., Melchiors also made the simplifying assumption that each task needed only one resource. The state in this MDP included

* Correspondence to: Industrial & Systems Engineering, BOX 352650, University of Washington, Seattle, WA, 98195, USA.
E-mail address: archis@uw.edu (A. Ghate).

http://dx.doi.org/10.1016/j.orl.2017.06.002
0167-6377/

information about waiting and ongoing tasks in each project. The decision-maker chose the tasks to work on in each period. A simulation-based ADP algorithm that employed value-function approximation was implemented.

We study infinite-horizon, discrete-time RCPSPs with dynamic arrivals of new heterogeneous projects. We allow for arbitrary arrival distributions and any (not necessarily linear) activity-on-node networks. One task may simultaneously require multiple resources. These features generalize the corresponding components of the problems studied in Choi et al. [5,6] and in Melchiors [11]. We focus on deterministic task durations and provide an MDP formulation of such dynamic resource constrained project scheduling problems (DRCPSPs). It turns out that this MDP is weakly coupled [1]. That is, the immediate expected profits are additively separable over project-types and transition probabilities are multiplicatively separable. Decisions about distinct project-types are only linked by resource-availability constraints. Exact solution of this MDP is intractable owing to the curse of dimensionality. Standard approaches for approximate solution of weakly coupled MDPs include Lagrangian relaxation and approximate linear programming [1,8,12]. Unfortunately, owing to the complicated state-evolution process in our MDP, such mathematical programming-based methods are computationally difficult to implement. We therefore apply a simulation-based approximate policy iteration algorithm [3,15] to this MDP. This method employs a value-function approximation whose parameters are tuned via simulation using least-squares fitting.

2. Problem statement

Consider a projection scheduling problem over time-stages $t = 1, 2, \ldots$ with the following notation.

1. $\mathcal{I} = \{1, 2, \ldots, I\}$ is the index set of project types.
2. Up to D^i new projects of type $i \in \mathcal{I}$ may arrive during one time-period. Let $p^i(m)$ denote the probability that $m \in \{0, 1, \ldots, D^i\}$ new projects of type $i \in \mathcal{I}$ arrive during one time-period.
3. For each project of type $i \in \mathcal{I}$, $\mathcal{N}^i = \{1, \ldots, N^i\}$ denotes the finite set of tasks that need to be accomplished in order to complete the project. Tasks are performed in a non-preemptive manner; i.e., once started, a task cannot be interrupted until it is complete.
4. For each task $n \in \mathcal{N}^i$, $\mathcal{M}^i_n \subset \mathcal{N}^i$ denotes the set of tasks that must be completed before starting task n. This set is called the set of predecessors of task n in project-type i.
5. Task durations are deterministic. Task $n \in \mathcal{N}^i$ takes Δ^i_n time-periods to complete.
6. $\mathcal{J} = \{1, \ldots, J\}$ is the set of resources. B^j is the integer quantity of resource $j \in \mathcal{J}$ available in each time-period. Task $n \in \mathcal{N}^i$ requires an integer amount $b^{ij}_n \geq 0$ of resource $j \in \mathcal{J}$.
7. By the beginning of any time-period, a project that arrived earlier may be completed, may be incomplete or may be waiting inception. The projects that have not been completed, i.e., the ones that are incomplete or waiting inception form a "queue". $W^i < \infty$ denotes the queue capacity for incomplete and waiting projects of type i.
8. The following rewards and costs are obtained or incurred.
   - Projects of type $i \in \mathcal{I}$ that arrive when W^i projects of that type are in the queue are rejected incurring a penalty cost G^i at the end of the time-period.
   - A reward R^i is received on completing a project of type $i \in \mathcal{I}$ at the end of a time-period.
   - A cost c^i_n per time-period is charged at the end of that time period for performing task n in project-type $i \in \mathcal{I}$.
   - An incomplete stalled project (no ongoing tasks) of type $i \in \mathcal{I}$ incurs a cost Q^i per period. This cost is charged at the end of the time-period.
   - A holding cost H^i per project of type $i \in \mathcal{I}$ is incurred in each time-period where such a project is waiting inception. This cost is incurred at the beginning of the time-period.
9. The goal is to maximize the total discounted expected profit (rewards minus costs) over an infinite-horizon where the per-period discount factor is $0 < \alpha < 1$.

An MDP model for this class of DRCPSPs is developed next.

3. A Markov decision process model

The MDP **states** are given by $X = (X^1, \ldots, X^I)$, where matrix X^i stores information about all incomplete and waiting type-i projects (i.e., projects in queue). The number of rows in X^i equals the number of type-i projects in queue and hence cannot exceed W^i. Let rows(X^i) denote the set of rows in X^i. Recall that $\mathcal{N}^i = \{1, \ldots, N^i\}$ is the set of tasks in type-i projects. Each row in X^i is of length N^i. The entries X^i_{lk} in the lth row and kth column of matrix X^i are defined as

$X^i_{lk} = -1$ if task $k \in \mathcal{N}^i$
in the lth type i project in queue has not yet started;
$X^i_{lk} = \Delta^i_k$ if task $k \in \mathcal{N}^i$
in the lth type-i project in queue has been completed;
$0 < X^i_{lk} < \Delta^i_k$ if task $k \in \mathcal{N}^i$
in the lth type-i project has started but not complete.

When $0 < X^i_{lk} < \Delta^i_k$, X^i_{lk} denotes the number of time-periods since the inception of task k. Let $\mathcal{X}^i$ denote the set of all state matrices X^i that are feasible with respect to precedence constraints for type-i projects. Also let $\mathcal{X} = \mathcal{X}^1 \times \mathcal{X}^2 \times \cdots \times \mathcal{X}^I$. We define the set $\mathcal{C}^i_l(X^i) = \{k : 0 < X^i_{lk} < \Delta^i_k\}$ of tasks that have started but are incomplete. Owing to our non-preemptive setting, tasks in $\mathcal{C}^i_l(X^i)$ must be processed in the current period. The set of all feasible states is then given by

$$\bar{\mathcal{X}} = \left\{ X \in \mathcal{X} : \sum_{i=1}^{I} \sum_{l \in \text{rows}(X^i)} \sum_{k \in \mathcal{C}^i_l(X^i)} b^{ij}_k \leq B^j, j \in \mathcal{J} \right\}. \tag{1}$$

The vector A of **action** matrices in our MDP is written as $A = (A^1, \ldots, A^I)$. Here, A^i is a matrix of zeros and ones, equal in size to X^i, and indicates which tasks will be started next. $A^i_{lk} = 1$ implies that we choose to begin task k in the lth type-i project in queue; A^i_{lk} is zero if we do not begin task k in the lth type-i project in queue. A feasible action must satisfy the logical restrictions

$$A^i_{lk} = 1 \text{ only if } X^i_{lk} = -1 \text{ and } X^i_{ln} = \Delta^i_n, \ \forall n \in \mathcal{M}^i_k. \tag{2}$$

This ensures that task k can be started only if it had not begun or completed earlier and if all of its predecessors have been completed. The set of all such action matrices for state matrix X^i is denoted by $\mathcal{U}^i(X^i)$. Also let $\mathcal{U}(X) = \mathcal{U}^1(X^1) \times \mathcal{U}^2(X^2) \times \cdots \times \mathcal{U}^I(X^I)$. Given the state–action pair (X^i, A^i), the matrix $X^i + A^i$ provides valuable information. In particular, we define the set of ongoing tasks in the lth type-i project as $\mathcal{V}^i_l(X^i, A^i) = \{k : 0 \leq X^i_{lk} + A^i_{lk} < \Delta^i_k\}$. Note that $\mathcal{C}^i_l(X^i) \subseteq \mathcal{V}^i_l(X^i, A^i)$ for every $A^i \in \mathcal{U}^i(X^i)$. The amount of resource $j \in \mathcal{J}$ consumed by all ongoing tasks from type-i projects equals $\sum_{l \in \text{rows}(X^i)} \sum_{k \in \mathcal{V}^i_l(X^i, A^i)} b^{ij}_k$ and is denoted by $B^{ij}\left(X^i, A^i\right)$. The set of all vectors of action matrices that are feasible