European Journal of Operational Research 000 (2015) 1–14

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

Solving stochastic resource-constrained project scheduling problems by closed-loop approximate dynamic programming

Haitao Li[*], Norman K. Womer

College of Business Administration, University of Missouri–St. Louis, One University Blvd, St. Louis, MO 63121, USA

ARTICLE INFO

Article history:
Received 21 May 2014
Accepted 8 April 2015
Available online xxx

Key words:
Resource-constrained project scheduling
Uncertain task durations
Stochastic scheduling
Approximate dynamic programming
Simulation

ABSTRACT

Project scheduling problems with both resource constraints and uncertain task durations have applications in a variety of industries. While the existing research literature has been focusing on finding an *a priori* open-loop task sequence that minimizes the expected makespan, finding a dynamic and adaptive closed-loop policy has been regarded as being computationally intractable. In this research, we develop effective and efficient approximate dynamic programming (ADP) algorithms based on the rollout policy for this category of stochastic scheduling problems. To enhance performance of the rollout algorithm, we employ constraint programming (CP) to improve the performance of base policy offered by a priority-rule heuristic. We further devise a hybrid ADP framework that integrates both the look-back and look-ahead approximation architectures, to simultaneously achieve both the quality of a rollout (look-ahead) policy to sequentially improve a task sequence, and the efficiency of a lookup table (look-back) approach. Computational results on the benchmark instances show that our hybrid ADP algorithm is able to obtain competitive solutions with the state-of-the-art algorithms in reasonable computational time. It performs particularly well for instances with non-symmetric probability distribution of task durations.

1. Introduction

Given a set of tasks and resources, a typical resource-constrained project scheduling problem (RCPSP) involves finding a time- and resource-feasible schedule of tasks, such that the project completion time (makespan) is minimized. It includes various shop scheduling problems such as job shop, flow shop, and open shop as special cases (Brucker, 2002), and has a wide range of applications in construction, manufacturing, R&D, personnel scheduling and military operations. The deterministic RCPSP is well-known to be NP-complete (Bartusch, Mohring, & Randermacher, 1988). Its solution methods have been extensively studied in the operations research literature. See Demeulemeester and Herroelen (2002), Kolisch and Hartmann (2006) and Debels and Vanhoucke (2007) for the state-of-the-art algorithms to solve deterministic RCPSPs.

In the real-world scheduling environment, the exact task duration is often unknown at the point when the scheduling decision is made, giving rise to the stochastic RCPSP (SRCPSP; Demeulemeester & Herroelen, 2002). For instance, construction projects can often be delayed by unexpected weather and/or disruption of logistics; accurate task processing time in the engineering-to-order (ETO) or made-to-order (MTO) settings can be difficult to obtain due to the uniqueness of order and learning; task durations in an R&D project are often uncertain due to the unforeseeable outcome of a predecessor task; planning for a military mission or campaign is often subject to uncertain task durations.

The classical approach to deal with uncertainty of task duration in project management is the well-known PERT analysis (Malcolm, Roseboom, Clark, & Fazar, 1959). PERT estimates the expected project makespan and its variation assuming given probability distribution of task durations, such as the commonly used beta-distribution. A limitation of the PERT method and its variants (cf. Dodin, 2006; Slyke & Richard, 1963) is the lack of decision-support. That is, these methods focus on understanding the statistical properties of project makespan, but they do not provide optimal start times, nor identify which path(s) will likely be critical, or the longest path. Other researchers have attempted to overcome this limitation of the PERT methodology (cf. Dodin, 1984; Elmaghraby, Ferreira, & Tavares, 2000). However, this line of research does not explicitly consider resource constraints: it is assumed that unlimited resources are available for project execution.

A successful approach to address exogenous disruptions on project task duration and resource availability, known as robust project scheduling, aims to construct a robust or stable project schedule in a resource-constrained environment. One may *proactively* construct a robust schedule to minimize the expected deviation from the baseline

[*] Corresponding author. Tel.: +1 314 516 5890.
E-mail address: lihait@umsl.edu, womerk@umsl.edu (H. Li).

http://dx.doi.org/10.1016/j.ejor.2015.04.015
0377-2217/© 2015 Elsevier B.V. All rights reserved.

schedule, which is often achieved by properly inserting time buffers in the schedule (cf. Goldratt, 1997; Herroelen & Leus, 2004); or *reactively* revise/re-optimize the schedule during project execution to obtain a feasible schedule with respect to the newly available information, that minimizes the deviation from the original baseline schedule (Van de Vonder, Ballestin, Demeulemeester, & Herroelen, 2007). Several researchers developed combined proactive–reactive procedures (cf. Demeulemeester, Herroelen, & Leus, 2008; Van de Vonder, Demeulemeester, Herroelen, & Leus, 2006). Recent work by Deblaere, Demeulemeester and Herroelen (2011) developed a dynamic proactive project execution policy to minimize the weighted activity starting time deviations plus the penalty or bonus for late or early project completion. Bruni, Beraldi, Guerrero, and Pinto (2011) proposed a decomposition-based heuristic method with the use of joint probabilistic constraints to obtain a feasible baseline schedule with the ability to hedge against variations of activity duration.

The SRCPSP studied in this paper minimizes the expected project makespan while considering both limited resource availability and uncertain task duration. The goal is to obtain a time- and resource-feasible task sequence that gives minimum expected project makespan. A deterministic schedule, in the form of task start times, is unable to provide an implementable solution to SRCPSP, as it can easily become time- or resource-infeasible due to random task durations. An implementable solution to the addressed problem requires a *policy*-type decision specifying which task(s) to start at each decision point (cf. Igelmund & Radermacher, 1983a; Mohring & Stork, 2000).

The fundamental difference between SRCPSP and the robust scheduling problems is the underlying scheduling setting and environment. In robust scheduling, the decision-maker must obtain a baseline schedule for the entire project upfront, but he/she does not have complete flexibility of revising it during project execution. Therefore, any deviation from the baseline schedule will be penalized, and its goal is to minimize total expected deviation penalty through: either obtaining a proactive baseline schedule to start with, and/or re-optimizing the schedule during execution in a reactive way. In the SRCPSP, we assume that project scheduling decisions are executed sequentially without the need of an *a priori* baseline schedule. The SRCPSP also differs from the other stream of research on resource allocation for projects, which focuses on optimizing the time-cost tradeoffs under uncertainty (cf. Gutjahr, Strauss & Wagner, 2000; Keller & Bayraksan, 2010; Shen, Smith & Ahmad, 2010).

The simplest and most popular policy-type solutions are various priority-based policies, in which all tasks are ranked according to a predefined priority rule, and started in the order specified by the priority. Although easy to implement and fast to execute, they suffer the so-called Graham's anomalies (Graham, 1966), and there are instances for which no priority-based policy generates an optimal schedule (Demeulemeester & Herroelen, 2002). Stochastic branch-and-bound procedures have been proposed by Igelmund and Radermacher (1983b) and Stork (2001). Recent research efforts have been focusing on the integrated simulation-optimization (Sim-Opt) algorithms, in which either a greedy heuristic (Golenko-Ginzburg & Gonik, 1997) or some metaheuristic (Glover & Kochenberger, 2005) is used to search the global solution space; while simulation is employed to evaluate a candidate or neighbor solution. Various metaheuristics have been implemented in such frameworks including a genetic algorithm (GA) by Ballestin (2007), tabu search (TS) by Tsai and Gemmill (1998), and greedy randomized adaptive search procedure (GRASP) by Ballestin and Leus (2009). Notably, Ashtiani, Leus, and Aryanezhad (2011) developed a new pre-processing procedure with a two-phase GA to obtain currently best results for SRCPSP in the literature. These approaches attempt to find a sequence of all tasks at time zero, without observing durations of early tasks. Using the terminology of the optimal control theory, they correspond to an *open-loop policy*. Such solutions are *static* in nature, and are not updated during real-time execution.

An alternative solution approach to SRCPSP is the *closed-loop* policy, in which scheduling decisions are made in a sequential fashion through the methodology of dynamic programming (DP; Bertsekas, 2007). Instead of optimizing the entire task sequence prior to project execution, a closed-loop policy seeks to find an optimal decision rule (policy) for selecting the task(s) to start at each decision-point, given the information a decision-maker knows about the current system. It is dynamic and adaptive in nature, which makes it possible to take advantage of information that becomes available between decision-points. Thus, in principle, a closed-loop policy is more flexible than an open-loop policy. We refer to Dreyfus and Law (1977) and Bertsekas (2007) for a systematic treatment of DP and closed-loop policies.

Although theoretically attractive, optimal closed-loop policies for SRCPSP have generally been perceived as being computationally intractable. There are only a handful of papers that attempt to offer closed-loop policies. Fernandez (1995) describes SRCPSP as a multi-stage sequential decision problem and proposes a decision-tree approach, which is computationally intractable for even a small number of tasks and scenarios. Fernandez and Armacost (1996) and Fernandez, Armacost, and Pet-Edwards (1998) discuss the advantage of modeling SRCPSP as a sequential decision problem and clarifies the importance of an *implementability* or *non-anticipativity* constraint in the formulation, without providing any computational results. Choi, Realff, and Lee (2004) consider a simplified job-shop version of SRCPSP with fixed sequence of tasks in each project, and present a DP algorithm with a heuristically confined state space. They show computational results for an example of project with 17 tasks.

The objective of this paper is to develop computationally tractable near-optimal closed-loop algorithms for solving reasonably large SRCPSPs. To tackle the curse-of-dimensionality, we have devised several schemes to approximately solve the Bellman equation (Bellman, 1957) in the Markov decision process model (MDP; Puterman, 2005) for SRCPSP. Our approximate dynamic programming (ADP) algorithm is built upon three core techniques. First, a sub-problem is constructed in each decision stage through an approximation of the exact recursive cost-to-go function in DP. Second, a forward iteration procedure is employed through sample paths generated by Monte Carlo (MC) simulation, which avoids the need of enumerating all possible states via the transition function in classical DP. Third, some deterministic scheduling methods are employed to handle the sub-problem in each ADP iteration.

Given the combinatorial nature of SRCPSP, we embed some scheduling heuristics in a *rollout* framework (Bertsekas, Tsitsiklis, & Wu, 1997) to sequentially improve a closed-loop solution. Such a rollout policy can be viewed as a *look-ahead* strategy that evaluates the expected cost of a state-action pair using the MC sample paths for all future stages. Our design of the rollout algorithm enhances its basic version in two ways. Only a small subset of random scenarios (features) are generated, using the idea of limited simulation proposed by Bertsekas and Castanon (1999), to reduce the burden of a pure MC simulation. In addition, since the quality of a closed-loop solution depends heavily upon the ability to solve the sub-problem in each iteration, we replace the simple priority-rule based heuristic by the more effective constraint programming (CP; Baptiste, Le Pape, & Nuijten, 2001) techniques to handle the scheduling sub-problem.

We then design a new approximation architecture integrating techniques in the artificial intelligence (AI) area to approximate the cost-go-go function. Rather than directly working on the optimization problem associated with the recursive cost-to-go function, the AI community has developed a suite of reinforcement learning (RL; Sutton & Barto, 1998) techniques to *learn* the value of a state-decision pair through sequential decision-environment interactions. Take the well-known lookup table approach for example: there, a record for the expected value of each state-decision pair is maintained and updated using MC simulation. In contrast to the rollout look-ahead policy, the lookup table approach can be viewed as *look-back* strategy that