CP or DP? Why Not Both: A Case Study in the Partial Shop Scheduling Problem.

Emma Legrand
ICTEAM, UCLouvain, Belgium

Roger Kameugne
ICTEAM, UCLouvain, Belgium

Pierre Schaus
ICTEAM, UCLouvain, Belgium

Abstract

Dynamic Programming (DP) and Constraint Programming (CP) are well-established paradigms for solving combinatorial optimization problems. Usually, these two approaches are used separately. This paper aims to show that the two can be combined effectively and elegantly, providing an alternative way to address the problem, with DP serving as the primary search framework and CP used as a subroutine to leverage global constraint propagation. This paper presents such an approach for the Partial Shop Scheduling Problem (PSSP), for which a pure DP method has previously been proposed, and efficient CP filtering algorithms are available. The PSSP is a general scheduling problem where each job consists of a set of operations with arbitrary precedence constraints. The approach is flexible enough to accommodate anytime DP strategies, such as anytime column search, whereas the original DP algorithm operated in a strictly layer-wise manner. While not competitive with state-of-the-art pure CP solvers for this specific problem, our primary contribution is demonstrating the viability of this hybrid integration.

2012 ACM Subject Classification Mathematics of computing → Combinatorial optimization

Keywords and phrases Partial Shop Scheduling Problem, dynamic programming, decision diagram, constraint programming

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

1 Introduction

This work considers solving the *Partial Shop Scheduling Problem* (PSSP) using a hybrid DP-CP approach. The PSSP is a general scheduling problem where each job consists of a set of operations with arbitrary precedence constraints. A *Partial Shop Scheduling Problem* (PSSP) defined by a finite set of operations $O = \{o_1, \ldots, o_N\}$ to be executed on a finite set of machines $M = \{M_1, \ldots, M_m\}$ with $|M| = m$. This set of operations is partitioned into $n = N/m$ subsets, with each machine assigned to exactly one operation in each partition. Each operation $o \in O$ has a specified processing time $p_o > 0$, requires a specific machine $m(o) \in M$ for its exclusive use without preemption, and is associated with a specific partition $j(o)$. Furthermore, the operations are subject to a set of precedence constraints represented by a directed acyclic graph (DAG) $G = (O, E)$. An edge $(o, o') \in E$ indicates that operation o must be completed before operation o' can start, denoted $o \lessdot o'$. A solution to this problem is an assignment of start times ψ_o to each operation $o \in O$ such that all precedence constraints and machine and partition disjunctive constraints (no two operations on the same machine/partition overlap in time) are satisfied. The objective is to minimize the makespan $C_{\max} = \max_{o \in O}(\psi_o + p_o)$.

In [3], the authors review the integration of CP and operations research (OR) to solve combinatorial optimization problems. A closely related work by Marijnissen et al. [5]

developed in parallel and independently of this one also proposes a generic framework for integrating CP propagation into Domain-Independent Dynamic Programming (DIDP) [4].

A DP formulation for the JSP was originally proposed by Gromicho et al. [2]. In this approach, operations are scheduled by fixing their start times. The state is defined by the set of scheduled operations, the earliest start times for each operation, and the ID of the last machine used. The transition function schedules an operation at its earliest starting time and propagates this information to successor operations. The domain function returns all available operations (i.e., all predecessors already scheduled) such that the current makespan is increased or remains equal, using the id of the last machine as the tiebreaker. To solve instances, they use a breadth-first search approach with dominance.

2 Hybrid DP-CP Model for the PSSP

Our contribution is to employ an anytime column search and full CP propagation. Using an anytime column search allows us to avoid exploring all nodes to find a solution, since the lower bound that guides the search is sufficient. For the first time, the lower bound used is Jackson's preemptive schedule.

We invoke a CP solver implementing all the filtering algorithms from [6] for the NoOverlap constraint on each machine, including *overload checking*, *edge-finding*, *detectable precedences*, and *not-first/not-last* rules. By computing tighter time windows through CP propagation and identifying additional precedence relations, the explored search space can therefore be further reduced.

The state definition from [2] is extended to explicitly represent the set of precedences (both original from the problem and those dynamically discovered through CP time-window filtering).

The transition function is adapted to use a CP model for our scheduling problem, further restricted by the current state. It propagates all constraints until a fixpoint and updates the current state. The domain function is adapted to take into account the precedences. Only operations whose predecessors are all already scheduled can be appended. This restriction reduces the size of the domain function and, consequently, the branching factor and the size of the search space explored by the DP approach.

The lower bound is used in state-space exploration to reduce the search space, thereby speeding up and tightening the problem bounds. We adopt a lower bound similar to that in [1], which uses a dichotomic approach with filtering algorithms. It receives as input a state S and an upper bound ub and computes a lower bound from S to be used in the anytime column search. A first lower bound is computed with the Jackson preemptive scheduling procedure. Then a dichotomic search is performed between this lower bound and the upper bound given as a parameter.

Some preliminary results will be presented during the presentation, showing the promising direction of this hybridization.

References

1 Jacques Carlier and Éric Pinson. An algorithm for solving the job-shop problem. *Management science*, 35(2):164–176, 1989.

2 Joaquim A. S. Gromicho, Jelke J. van Hoorn, Francisco Saldanha-da-Gama, and Gerrit T. Timmer. Solving the job-shop scheduling problem optimally by dynamic programming. *Comput. Oper. Res.*, 39(12):2968–2977, 2012.

88 **3** John N. Hooker and Willem Jan van Hoeve. Constraint programming and operations research.
89 *Constraints An Int. J.*, 23(2):172–195, 2018.
90 **4** Ryo Kuroiwa and J Christopher Beck. Domain-independent dynamic programming: Gen-
91 eric state space search for combinatorial optimization. In *Proceedings of the International*
92 *Conference on Automated Planning and Scheduling*, volume 33, pages 236–244, 2023.
93 **5** Imko Marijnissen, J Christopher Beck, Emir Demirović, and Ryo Kuroiwa. Domain-independent
94 dynamic programming with constraint propagation. *arXiv preprint arXiv:2603.16648*, 2026.
95 **6** Petr Vilím. Global constraints in scheduling. *Univerzita Karlova, Matematicko-fyzikální*
96 *fakulta*, 2007.