GRID: Graph-based Modelling Interface for Domain-Independent Dynamic Programming

Fabio Giordana
Department of Computer Science and Engineering, University of Bologna, Bologna, Italy

Zeynep Kiziltan
Department of Computer Science and Engineering, University of Bologna, Bologna, Italy

Ryo Kuroiwa
Principles of Informatics Division, National Institute of Informatics, Tokyo, Japan
The Graduate University for Advanced Studies, SOKENDAI, Kanagawa, Japan

Abstract

Domain-Independent Dynamic Programming (DIDP) is a general framework for solving combinatorial optimization problems using Dynamic Programming (DP), where search is separated from the problem specification. However, modelling directly in DIDP requires defining state variables, transitions, and dominance relations, which can be complex and error-prone. We introduce GRID, a modelling interface that enables high-level DP-oriented specifications. Users describe entities, relations, and resource attributes over a graph-based structure, from which GRID automatically compiles a DIDP model by deriving the components required by a DIDP solver. Common modelling elements are supported with built-in semantics, while additional constraints can be specified through user-defined variables and expressions. We use vehicle routing problems as a case study to present GRID and evaluate it on three variants: CVRP, PDPTW, and ECVRP. Results show that compilation overhead from GRID to DIDP is small and that generated models remain competitive with manually designed DIDP models while outperforming state-of-the-art CP and mixed-integer programming approaches.

2012 ACM Subject Classification Theory of computation → Dynamic programming; Computing methodologies → Modeling methodologies; Mathematics of computing → Combinatorial optimization

Keywords and phrases Modelling & Modelling Languages, Dynamic Programming

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Supplementary Material *Software*: `GRID and Models (public release at CP 2026)`

Funding *Ryo Kuroiwa*: JSPS KAKENHI grant number JP25K24378

Acknowledgements This work was supported by the NII-UniBo MoU Grant funded by NII.

1 Introduction

Domain-Independent Dynamic Programming (DIDP) [10, 12] solves combinatorial optimization problems via Dynamic Programming (DP) by decoupling the problem specification from the search procedure, so that a generic solver can be reused across domains. Modelling directly in DIDP is difficult: users must manually design state representations, transitions, and dominance relations. The CP community offers high-level languages [7, 15], libraries [8, 13], and problem-specific features such as sequence variables for VRPs [4, 3], but comparable DP tools are lacking. To bridge this gap, we introduce GRID, a graph-based modelling interface that automatically compiles intuitive, DP-oriented specifications into executable DIDP models.

2 GRID: A Graph-based Modelling Interface

GRID allows high-level, DP-oriented specifications of combinatorial optimization problems over a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, on which users describe entities, relations, and resource attributes. From this description, GRID derives the DIDP model automatically.

Because a route is a sequence on a graph, GRID is well suited to VRPs [1] and related sequencing problems such as job scheduling [2], but not to general graph optimization problems like vertex cover, facility location, or graph coloring. In this initial version, we focus on VRPs as a natural and representative use case. Implemented as a Python library, the VRP interface revolves around four classes: `RoutingModel` (managing the graph and entities), `Node` and `Edge` (locations and directed connections), and `VehicleType` (homogeneous fleets).

Native features, predefined constructs with built-in semantics, let users express common VRP variants via object attributes (e.g., demand, capacity) or built-in objectives. *Custom features* handle non-standard variants through user-defined variables at the `VehicleType` scope and expressions attached to nodes and edges. For instance, Electric CVRP [14] needs custom rules for load-dependent battery consumption, which neither native features nor most VRP-specific solvers nor sequence-variable constraints can express, since these assume edge-wise updates by a constant.

3 Compilation to DIDP

A GRID model is compiled into DyPDL [10, 12], the DP modelling language of DIDP, via a single-route representation that concatenates per-vehicle routes. The compiler derives the core state variables, lifts user-defined variables into DyPDL state variables, and generates three transition types (*Visit*, *Change Vehicle*, *Finish*) with their feasibility conditions, cost updates, resource-based dominance, and a default dual bound.

4 Experimental Evaluation and Conclusions

We evaluate GRID on CVRP, PDPTW [6], and ECVRP [14]. We compare GRID against manually designed DIDP models, Gurobi [9] (MIP), OR-Tools CP-SAT [16] (CP) and MaxiCP [17] (CP with sequence variables), OR-Tools VRSolver [5] and PyVRP [18] (VRP-specific), with a 5 minutes time limit and 8 GB memory, on instances of up to a thousand nodes.

Compilation from GRID to DIDP is small and well below the model-building times of MIP and CP-SAT. For quality, we report the primal gap $|\gamma - \gamma^*| / \max\{\gamma, \gamma^*\}$, where γ is the best objective found by a solver and γ^* the best-known value for the instance. GRID and manually designed DIDP models yield nearly indistinguishable primal-gap distributions on all variants, confirming that the high-level abstractions do not penalize the solver. On PDPTW, VRSolver is strongest, while GRID/DIDP closely follow and outperform MIP, and both CP models. On ECVRP, MIP and CP-SAT run out of memory and tested VRP-specific solvers and the CP sequence variables approach cannot express load-dependent battery consumption; a refined GRID model with extra user-defined variables outperforms a simpler one. On CVRP, PyVRP is strongest, while GRID slightly improves over the previously published manual DIDP formulation [11, 12].

No approach dominates: VRP-specific solvers excel on classical variants but lack flexibility, while MIP and CP scale poorly. GRID matches manually designed DIDP and outperforms MIP and CP through a flexible, declarative interface. Future work will focus on extending GRID beyond VRPs, designing generic solving algorithms (e.g., LNS), and compiling GRID models to non-DP back-ends (CP, MIP, VRP solvers).

References

1. C. Archetti, L.C. Coelho, M.G. Speranza, and P. Vansteenwegen. Beyond fifty years of vehicle routing: Insights into the history and the future. *Eur. J. Oper. Res.*, 330(2):355–372, 2026. doi:10.1016/j.ejor.2025.06.014.
2. J. Beck, Patrick Prosser, and E. Selensky. Vehicle routing and job shop scheduling: What's the difference? In *ICAPS*, pages 267–276, 2003.
3. Augustin Delecluse, Pierre Schaus, and Pascal Van Hentenryck. Sequence variables: A constraint programming computational domain for routing and sequencing. *CoRR*, abs/2510.09373, 2025. arXiv:2510.09373, doi:10.48550/arXiv.2510.09373.
4. Augustin Delecluse, Pierre Schaus, and Pascal Van Hentenryck. Sequence variables for routing problems. In *CP*, volume 235, pages 19:1–19:17, 2022. doi:10.4230/LIPIcs.CP.2022.19.
5. Frédéric Didier, Laurent Perron, Sarah Mohajeri, Steven Agostino Gay, Thibaut Cuvelier, and Vincent Furnon. OR-Tools' Vehicle Routing Solver: a generic constraint-programming solver with heuristic search for routing problems. In *ROADEF*, 2023.
6. Yvan Dumas, Jacques Desrosiers, and François Soumis. The pickup and delivery problem with time windows. *Eur. J. Oper. Res.*, 54(1):7–22, 1991. doi:10.1016/0377-2217(91)90319-Q.
7. Alan M. Frisch, Warwick Harvey, Chris Jefferson, Bernadette Martínez-Hernández, and Ian Miguel. Essence: A constraint language for specifying combinatorial problems. *Constraints*, 13(3):268–306, 2008. doi:10.1007/s10601-008-9047-y.
8. Tias Guns. Increasing modeling language convenience with a universal n-dimensional array, CPpy as python-embedded example. In *ModRef*, 2019.
9. Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2026. URL: https://www.gurobi.com.
10. Ryo Kuroiwa and J. Christopher Beck. Domain-independent dynamic programming: Generic state space search for combinatorial optimization. In *ICAPS*, pages 236–244. AAAI Press, 2023. doi:10.1609/icaps.v33i1.27200.
11. Ryo Kuroiwa and J. Christopher Beck. Solving domain-independent dynamic programming problems with anytime heuristic search. In *ICAPS*, pages 245–253, 2023. doi:10.1609/ICAPS.V33I1.27201.
12. Ryo Kuroiwa and J. Christopher Beck. Domain-independent dynamic programming. *Artif. Intell.*, 354:104506, 2026. doi:10.1016/j.artint.2026.104506.
13. Christophe Lecoutre and Nicolas Szczepanski. PyCSP3: Modeling combinatorial constrained problems in Python. *CoRR*, abs/2009.00326, 2020. arXiv:2009.00326.
14. Michalis Mavrovouniotis, Charalambos Menelaou, Stelios Timotheou, Georgios Ellinas, Christoforos Panayiotou, and Marios M. Polycarpou. A benchmark test suite for the electric capacitated vehicle routing problem. In *CEC*, pages 1–8, 2020. doi:10.1109/CEC48606.2020.9185753.
15. Nicholas Nethercote, Peter J. Stuckey, Ralph Becket, Sebastian Brand, Gregory J. Duck, and Guido Tack. MiniZinc: Towards a standard CP modelling language. In *CP*, volume 4741, pages 529–543. Springer, 2007. doi:10.1007/978-3-540-74970-7_38.
16. Laurent Perron, Frédéric Didier, and Steven Gay. The CP-SAT-LP solver. In *CP*, volume 280, pages 3:1–3:2, 2023. doi:10.4230/LIPIcs.CP.2023.3.
17. Pierre Schaus, Guillaume Derval, Augustin Delecluse, Laurent Michel, and Pascal Van Hentenryck. MaxiCP: A constraint programming solver for scheduling and vehicle routing, 2024. URL: https://github.com/aia-uclouvain/maxicp.
18. Niels A. Wouda, Leon Lan, and Wouter Kool. PyVRP: A high-performance VRP solver package. *INFORMS J. Comput.*, 36(4):943–955, 2024. doi:10.1287/ijoc.2023.0055.