Domain-Independent Dynamic Programming (Student Abstract)

Ryo Kuroiwa*

Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, Canada, ON M5S 3G8
ryo.kuroiwa@mail.utoronto.ca

Abstract

In my dissertation, I will propose Domain-Independent Dynamic Programming (DIDP), a novel model-based paradigm for combinatorial optimization (CO) based on dynamic programming (DP). In DIDP, a problem is first formulated as a declarative DP model and then solved by a general-purpose solver. The goal of my dissertation is to develop an algorithm-independent modeling formalism to define a DP model and general-purpose solvers for it and demonstrate that DIDP is promising for CO in practice. In particular, I will propose a modeling formalism based on a state transition system and heuristic search solvers for it.

Introduction

Combinatorial optimization (CO) is to make a set of discrete decisions from a finite set to optimize an objective function. Model-based approaches such as mixed-integer programming (MIP) and constraint programming (CP) are widely used for CO. In such a approach, a user formulates a problem as a mathematical model and then solves it using a general-purpose solver. Model-based approaches are considered the 'holy grail' of declarative problem-solving, where a user just needs to define a problem to solve it (Freuder 1997).

In dynamic programming (DP), a problem is modeled with recursive equations. For CO, DP has been typically implemented as customized problem-specific algorithms, which exploit problem-specific information to reduce the solving effort. With the success of these methods in multiple CO problems, I want to develop a model-based DP paradigm that can exploit problem-specific information while being generic and problem-independent.

I propose Domain-Independent Dynamic Programming (DIDP), a model-based paradigm for CO based on DP. In DIDP, a user defines a problem as a DP model using a solver-independent formalism. The formalism also allows (but does not require) a user to declaratively incorporate redundant information, which is implied by the problem definition but can be exploited by a solver. For general-purpose solvers, I will use heuristic search algorithms, which can easily and efficiently exploit the redundant information. The goal of my dissertation is to demonstrate that DIDP is a promising model-based approach for CO in practice.

Research Progress

I have proposed a modeling formalism and general-purpose solvers for it. The developed software is open-source.[1]

DyPDL: a Modeling Formalism

I developed Dynamic Programming Description Language (DyPDL), a modeling formalism for DIDP (Kuroiwa and Beck 2023a). In DyPDL, a problem is formulated in a state-based representation. For example, in the traveling salesperson problem with time windows (TSPTW), a set of customers $N = \{0, ..., n\}$ are given, where the depot is 0, and visiting customer j from i incurs travel time c_{ij}. We assume that $c_{ik}+c_{kj} \geq c_{ij}$. A salesperson must visit each customer i in $N \setminus \{0\}$ within time window $[a_i, b_i]$ and then return to the depot while minimizing the total travel time. Let U be the set of unvisited customers, i be the current location, and t be the current time. These variables define a state of the problem: they are *state variables* in DyPDL. Let V be a function mapping a state to its optimal objective value.

$$\text{compute } V(N \setminus \{0\}, 0, 0) \tag{1}$$

$$V(U, i, t) = \begin{cases} \min_{j \in U : t + c_{ij} \leq b_j} c_{ij} + V(U \setminus \{j\}, j, t'_j) & \text{if } U \neq \emptyset \\ c_{i0} & \text{if } U = \emptyset \end{cases} \tag{2}$$

$$V(U, i, t) = \infty \qquad \text{if } \exists j \in U, t + c_{ij} > b_j \tag{3}$$

$$V(U, i, t) \leq V(U, i, t') \qquad \text{if } t \leq t' \tag{4}$$

$$V(U, i, t) \geq 0 \tag{5}$$

where $t'_j = \max\{t + c_{ij}, a_j\}$. Objective (1) declares that we want to compute the optimal objective value of the state where all customers except for the depot are unvisited, the current location is the depot, and $t = 0$. Equation (2) is the main body of the DP model: a state is transformed to another by visiting a customer, and when all customers are visited, the objective value is c_{i0}, the cost to return to the depot. In DyPDL, these dynamics are described as *transitions*. A transition is defined by applicability conditions, the changes in the objective, and the changes in the state. In addition, a user can model redundant information to achieve better performance. Equation (3) defines a *state constraint*, which states

*Supervised by Prof. J. Christopher Beck.

[1] https://didp.ai

	MIP			CP			CAASDy			CABS		
	#	gap	p.i.	#	gap	p.i.	#	gap	p.i.	#	gap	p.i.
TSPTW (340)	227	0.227	484.1	47	0.026	49.0	257	0.244	458.6	**259**	**0.003**	**9.00**
CVRP (207)	**26**	0.585	1157.4	0	0.317	601.2	5	0.976	1757.1	6	**0.185**	**351.2**
m-PDTSP (1178)	945	0.078	180.0	**1049**	0.013	26.0	947	0.196	357.8	1035	**0.002**	**5.3**
SALBP-1 (2100)	1357	0.345	634.6	1584	0.005	28.5	1653	0.213	387.5	**1801**	**0.000**	**1.9**
Bin Packing (1615)	1157	0.039	86.2	**1234**	**0.002**	8.0	922	0.429	779.4	1163	**0.002**	**5.3**
MOSP (570)	224	0.039	100.4	437	0.004	13.0	483	0.153	275.5	**527**	**0.000**	**0.4**
Graph-Clear (135)	24	0.110	311.8	1	0.015	44.3	76	0.437	789.9	**103**	**0.000**	**0.5**
Talent Scheduling (1000)	0	0.051	142.9	0	**0.002**	18.1	207	0.793	1435.5	**237**	0.011	26.4
$1\|\|\sum w_i T_i$ (375)	109	0.018	74.6	150	**0.000**	2.3	270	0.280	513.7	**284**	0.034	73.6
Average ratio	0.424			0.416			0.587			**0.659**		

Table 1: Number of optimally solved instances ('#'), the average primal gap ('gap'), and the average primal integral ('p.i.') with 30 minutes and 8GB memory. 'Average ratio' is the mean of the per-problem-class proportion of solved instances.

that a state is a dead-end if any customer cannot be visited by its deadline. Inequality (4) defines the dominance relation based on t, and t is called a *resource variable*. Inequality (5) is a *dual bound*, a lower bound on the objective value.

CAASDy: a Prototype Solver

The DP model for TSPTW can be solved as a shortest path problem in a state space graph: nodes are states, edges are transitions, the weight of each edge is the travel time, and the shortest path cost from a node is the optimal objective value of the corresponding state. In general, if a DyPDL model satisfies certain conditions, cost algebraic heuristic search (Edelkamp, Jabbar, and Lafuente 2005), generalized versions of shortest path algorithms, can solve it. I developed Cost Algebraic A* Solver for DyPDL (CAASDy), a prototype solver using the cost algebraic version of A* (Kuroiwa and Beck 2023a). CAASDy exploits redundant information in a given model; it uses the dual bound (Inequality (5)) as a heuristic function, which guides the search. In addition, CAASDy prunes a state if it does not satisfy a state constraint (Equation (3)) or a better state according to a dominance relation (Inequality (4)) has been already explored. I evaluated CAASDy using six problems, TSPTW, capacitated vehicle routing problems (CVRP), single assembly line balancing problems (SALBP-1), bin packing, the minimization of open stacks problem (MOSP), and graph-clear as benchmarks. I experimentally compared the DP models with CAASDy against MIP and CP models with commercial solvers. Table 1 shows that CAASDy solves more instances in four problems and a larger portion of instances on average than MIP and CP within the time and memory limits.

Anytime Solvers

CAASDy does not find any solution until it finds the optimal solution. In contrast, general-purpose MIP and CP solvers are usually anytime solvers, which often quickly find a feasible solution and continuously improve it until the optimality is proved. I developed anytime solvers for DyPDL using six anytime heuristic search algorithms (Kuroiwa and Beck 2023b) and evaluated them using the benchmark problems extended with multi-commodity pick-and-delivery traveling salesperson problem (m-PDTSP), talent scheduling, and the single machine total weighted tardiness ($1||\sum w_i T_i$). The primal gap and primal integral (Berthold 2013) are used as performance measures. The primal gap is the relative gap between a solution cost to the optimal (or best-known) solution cost scaled from 0 to 1, and smaller is better. The primal integral considers the change of the primal gap over time, and smaller is better. Overall, the best solver is Complete Anytime Beam Search (CABS) (Zhang 1998), which iteratively performs beam search with an exponentially increasing beam width, in all the metrics. We show the result of CABS in Table 1. It consistently outperforms CAASDy in all problems and MIP and CP in the majority of problems.

Future Work

For the rest of my thesis, I will focus on improving DIDP solvers by leveraging techniques developed in heuristic search. First, I will develop parallel solvers based on parallel heuristic search algorithms such as HDA* (Kishimoto, Fukunaga, and Botea 2013). In addition, I am considering developing methods to automatically extract a heuristic function from a model. Such methods are actively studied in planning (e.g., merge-and-shrink (Helmert et al. 2014)) and are useful when a user does not provide a good dual bound.

References

Berthold, T. 2013. Measuring the Impact of Primal Heuristics. *Oper. Res. Lett.*, 41: 611–614.

Edelkamp, S.; Jabbar, S.; and Lafuente, A. L. 2005. Cost-Algebraic Heuristic Search. In *Proc. AAAI*, 1362–1367.

Freuder, E. 1997. In Pursuit of the Holy Grail. *Constraints*, 2: 57–61.

Helmert, M.; Haslum, P.; Hoffmann, J.; and Nissim, R. 2014. Merge-and-shrink abstraction. *J. ACM*, 61(3): 1–63.

Kishimoto, A.; Fukunaga, A.; and Botea, A. 2013. Evaluation of a simple, scalable, parallel best-first search strategy. *Artif. Intell.*, 195: 222–248.

Kuroiwa, R.; and Beck, J. C. 2023a. Domain-Independent Dynamic Programming: Generic State Space Search for Combinatorial Optimization. In *Proc. ICAPS*.

Kuroiwa, R.; and Beck, J. C. 2023b. Solving Domain-Independent Dynamic Programming Problems with Anytime Heuristic Search. In *Proc. ICAPS*.

Zhang, W. 1998. Complete Anytime Beam Search. In *Proc. AAAI*, 425–430.