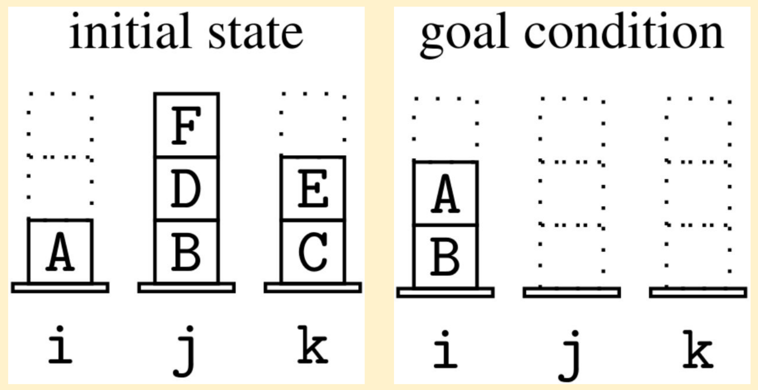
Graph Learning for Numeric Planning

Dillon Z. Chen[1,2] Sylvie Thiébaux[1,2]
[1]LAAS-CNRS, Université de Toulouse, [2]The Australian National University
{dillon.chen, sylvie.thiebaux}@laas.fr

Numeric Planning

- factored representations of states
  - **predicates** for Boolean values:
    `on(A, i), (F, D), on(D, B), on(B, j), on(E, C), on(C, k)`
  - **functions** for numeric values
    `capacity(i) = 2, capacity(j) = 0, capacity(k) = 1`
- transition system and goals specified
  - by Boolean assignments
    `on(A, B), on(B, i)`
  - numeric conditions
    `capacity(i) >= 1`
- solution:
  - goal-reaching sequence of actions



Planning and RL: What's the Difference?

Both Solve MDPs!

Planning	Reinforcement Learning
model-known	model-free or model-based
goal-conditioned, minimise cost	maximise reward
transitions modelled symbolically	transitions modelled as distributions
search algorithms: A*, GBFS, iLAO*, LRTDP	search algorithms: MTCS, UCT, TD(λ)
heuristic functions (cost-to-go estimator)	value functions (expected reward)
algorithms guided by models	algorithms guided by rewards

Graph Learning

- in the top 3 keywords of the past 3 ICLR conferences
- handles relational data and arbitrarily sized inputs = planning
- deep learning
  - Graph Neural Networks (GNNs)
- classical ML
  - graph kernels

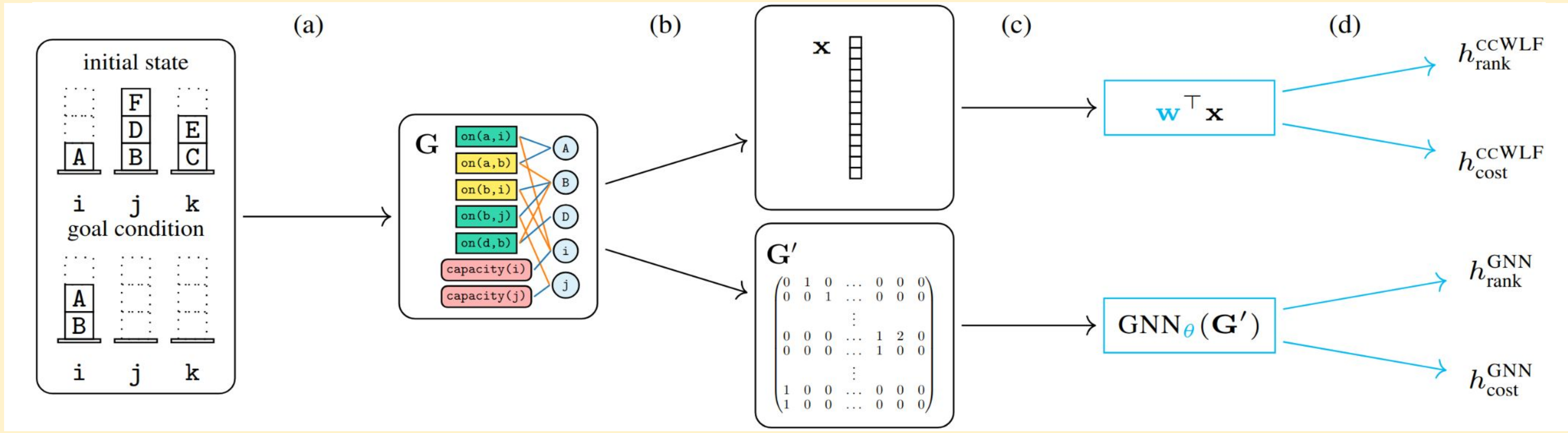
Learning Heuristic Functions

(a) numeric planning tasks are **transformed into graphs** with edge labels, and both categorical and continuous node features

(b) graphs are fed into graph kernels (top) or transformed into matrix representations (bottom)

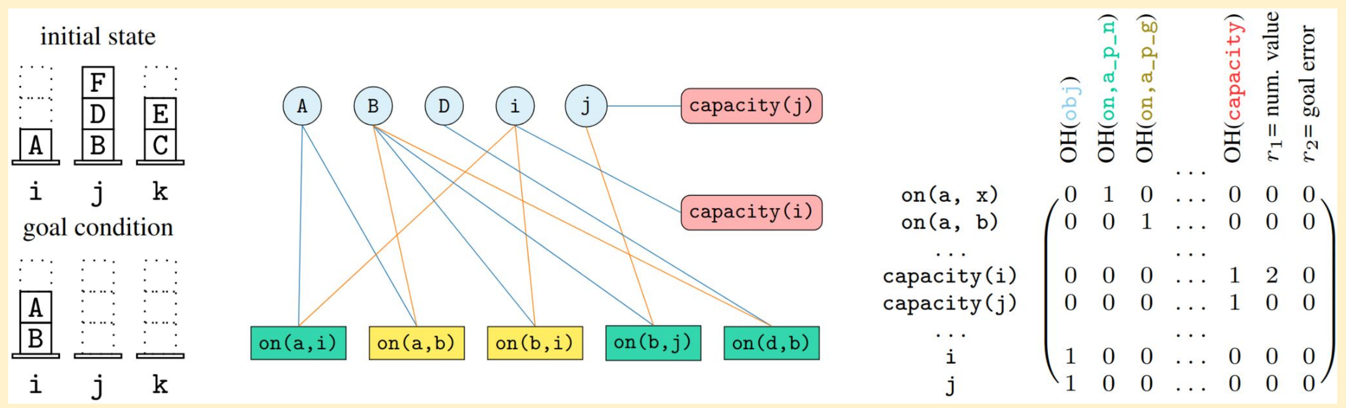
(c) graph kernel features use a linear kernel; matrix representations are fed into GNNs

(d) models **learn a heuristic function** as either cost-to-go estimators or ranking functions used for search



New Contributions

- graph representation for numeric planning



- graph kernel for categorical-continuous attributes

Algorithm 1: CCWL algorithm

Data: A graph $\mathbf{G} = \langle \mathbf{V}, \mathbf{E}, \mathbf{F}_{\text{cat}}, \mathbf{F}_{\text{con}}, \mathbf{L}\rangle$ with continuous and categorical attributes, a deterministic HASH function, allowed colours $\mathbf{C} = [|\mathbf{C}|]$, a pooling function POOL, and number of CCWL iterations L.

Result: Feature vector of size $\mathbb{R}^{(1+d)|\mathbf{C}|}$.

$\kappa^0(v) \leftarrow \mathbf{F}_{\text{cat}}(v), \forall v \in \mathbf{V}$

for $j \in [L]$ **do for** $v \in \mathbf{V}$ **do**

$\kappa^j(v) \leftarrow \text{HASH}\left(\kappa^{j-1}(v), \bigcup_{\iota\in\Sigma_E}\{\{(\kappa^{j-1}(u),\iota) \mid u \in \mathbf{N}_\iota(v)\}\}\right)$

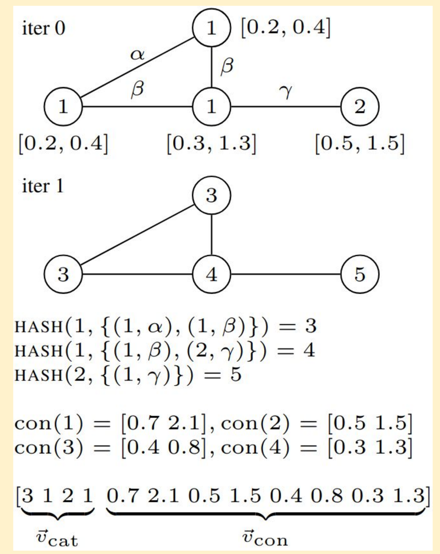
$\mathbf{M} \leftarrow \bigcup_{j=0,\dots,L}\{\{\kappa^j(v) \mid v \in \mathbf{V}\}\}$

$\vec{v}_{\text{cat}} \leftarrow [\text{COUNT}(\mathbf{M}, c_1), \dots, \text{COUNT}(\mathbf{M}, c_{|\mathbf{C}|})]$

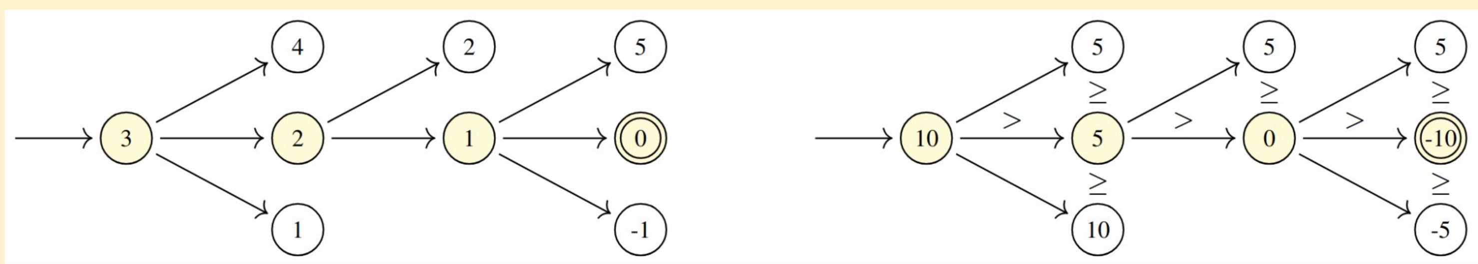
$\mathbf{S}_i = \{v \mid v \in \mathbf{V} \text{ s.t. } \exists j \in \{0\} \cup [L], \kappa^j(v) = c_i\}, \forall i \in \mathbf{C}$

$\vec{v}_{\text{con}} \leftarrow [\text{con}(1) \| \dots \| \text{con}(|\mathbf{C}|)]$, $\text{con}(i) = \text{POOL}_{v\in\mathbf{S}_i}(\mathbf{F}_{\text{con}}(v))$

return $\vec{v}_{\text{cat}} \| \vec{v}_{\text{con}}$

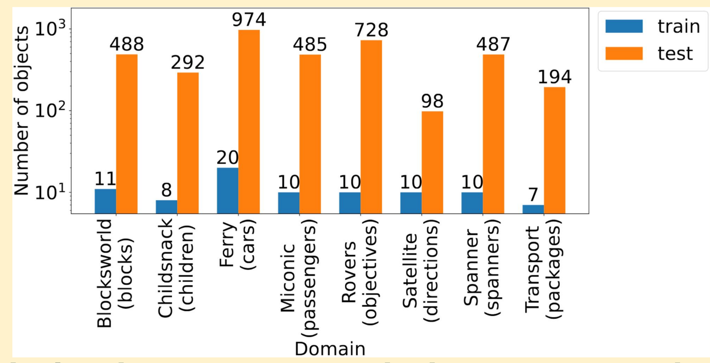


- LP ranking formulation for learning heuristic

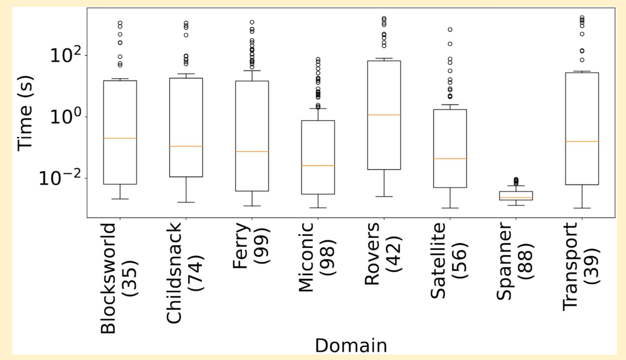


Experiments

- Generalise across object sizes → out-of-distribution learning



- access to model gives us quick generation of training data



- competitive with domain-independent numeric planners
  - purple = numeric planners
  - orange = learning planners

higher values better (↑)