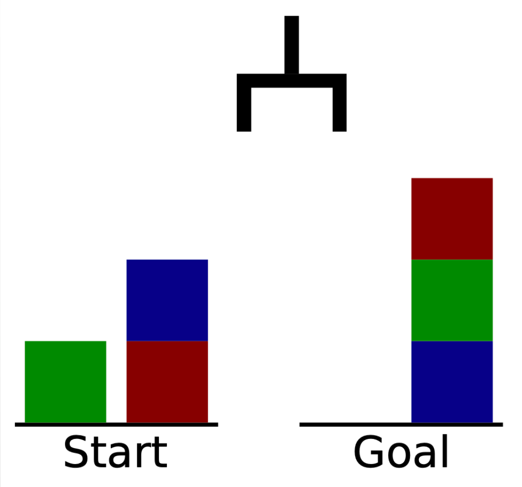
Online Planner Selection with Graph Neural Networks and Adaptive Scheduling

Tengfei Ma,[1] Patrick Ferber,[3,4] Siyu Huo,[1] Jie Chen,[1,2] Michael Katz[1]

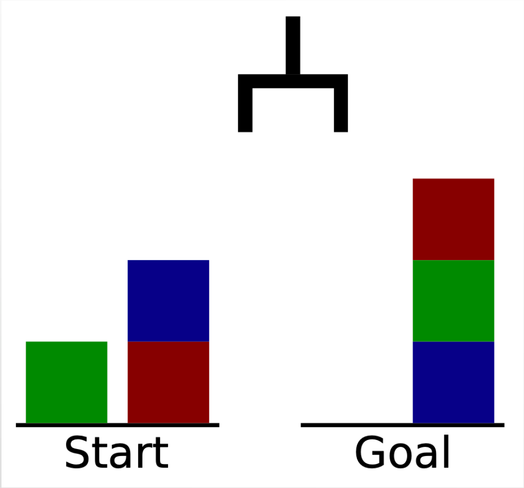
[1]IBM Research, [2]MIT-IBM Watson AI Lab, [3]University of Basel, [4]Saarland University

Background and Context

- Planning is one of the foundational areas of AI
- Planning is intractable in general (e.g., classical planning is PSPACE-complete)
- A single algorithm unlikely works well for all planning tasks and problem domains
- Portfolio-based approach: Build a portfolio of planners and select one on demand
- How?



Machine Learning Can Help

$f\Big($  $\Big) =$

34.03	h2-simpless-dks-celmcut
74.14	h2-simpless-dks-cpdbshc900
time out	simpless-oss-masb50kmiasmdfp
1499.84	h2-simpless-oss-900masb50ksbmiasm
964.99	seq-opt-symba-1
$\vdots$	

Machine Learning Can Help

Input consists of hand-crafted features; f is a classic machine learning model (e.g., SVM)

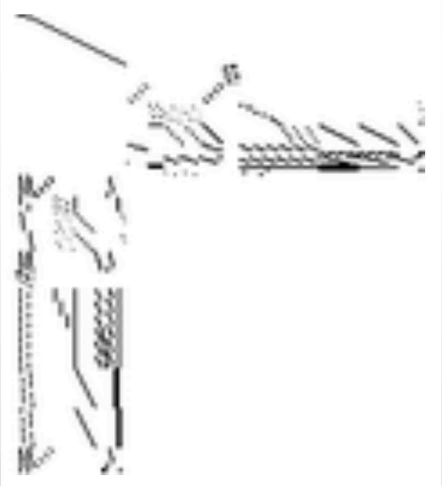
$$f\begin{pmatrix}\text{number of objects}\\ \text{number of axioms}\\ \text{whether action costs are used}\\ \text{number of mutex groups}\\ \text{number of variables of the CG}\\ \text{maximum of accumulated costs}\\ \text{of paths to goal propositions in}\\ \text{the relaxed problem}\end{pmatrix} = \begin{bmatrix}34.03\\ 74.14\\ \text{time out}\\ 1499.84\\ 964.99\\ \vdots\end{bmatrix}$$

h2-simpless-dks-celmcut
h2-simpless-dks-cpdbshc900
simpless-oss-masb50kmiasmdfp
h2-simpless-oss-900masb50ksbmiasm
seq-opt-symba-1

Machine Learning Can Help

Delfi (winner of the Optimal Track of the 2018 International Planning Competition)

Input is an image converted from graph representation of the task; f is convolutional neural network

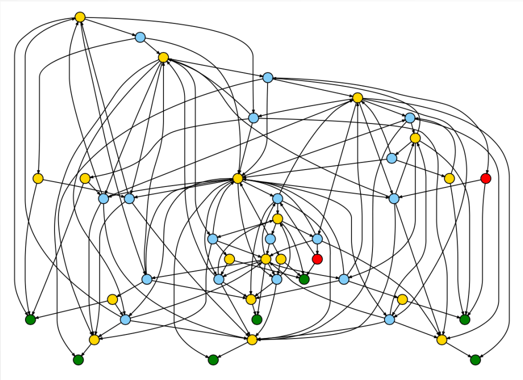
$f\Bigg($  $\Bigg) =$

$$\left[\begin{array}{c} 34.03 \\ 74.14 \\ \text{time out} \\ 1499.84 \\ 964.99 \\ \vdots \end{array}\right] \begin{array}{l} \text{h2-simpless-dks-celmcut} \\ \text{h2-simpless-dks-cpdbshc900} \\ \text{simpless-oss-masb50kmiasmdfp} \\ \text{h2-simpless-oss-900masb50ksbmiasm} \\ \text{seq-opt-symba-1} \\ \end{array}$$

Machine Learning Can Help

This work

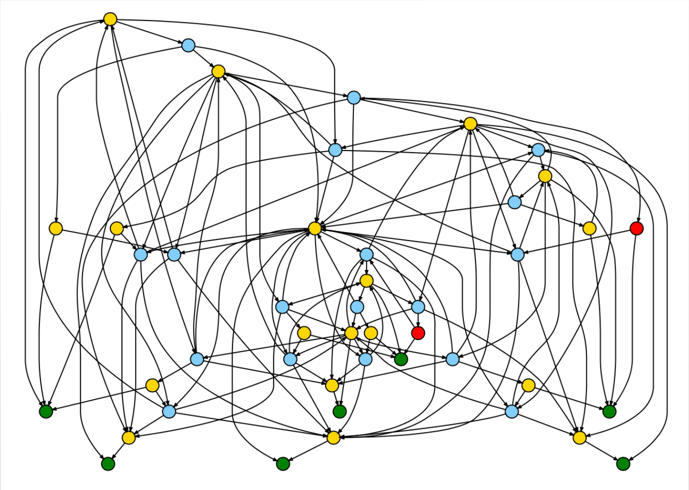
Input is graph representation of the task; f is graph neural network



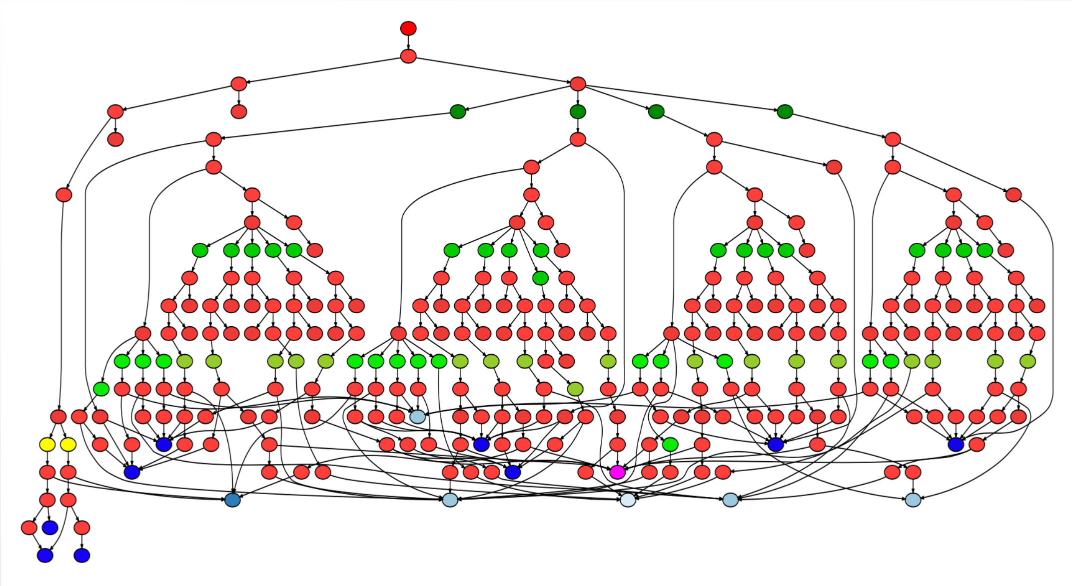
$$f\left(\quad \right) = \begin{array}{c} \begin{bmatrix} 34.03 \\ 74.14 \\ \text{time out} \\ 1499.84 \\ 964.99 \\ \vdots \end{bmatrix} \end{array} \begin{array}{l} \text{h2-simpless-dks-celmcut} \\ \text{h2-simpless-dks-cpdbshc900} \\ \text{simpless-oss-masb50kmiasmdfp} \\ \text{h2-simpless-oss-900masb50ksbmiasm} \\ \text{seq-opt-symba-1} \\ \end{array}$$

Graph Representations of a Planning Task

- State transition graph (memory prohibitive)
- Problem description graph (also called grounded representation) [Pochter et al. 2011]



- Abstract structure graph (also called lifted representation) [Sievers et al. 2019]



Pros and Cons

	Pros	Cons
Hand-crafted feature input	✓ Rich domain knowledge	✗ Engineering good features is a laborious task
Image input	✓ No feature engineering needed ✓ Convolutional neural networks have been constantly improved	✗ Not permutation invariant ✗ Cannot leverage node/edge attributes
Graph input	✓ Permutation invariant ✓ Leverage node/edge attributes ✓ No feature engineering needed ✓ Graph neural networks are emerging (ample opportunity for improvement)	

Planner Selection with Graph Neural Networks

Setting

- Cost-optimal planning (must use optimal planner)
- Given time allowance T, identify a planner that completes within T (no need to be the fastest)
- Online scheduling (selected planner dependent on task)

Problem formulation

- Task G (as a graph)
- D planners
- $y \in \{0,1\}^D$, 0 success, 1 fail
- Find a function $f(G, \theta)$ as close to y as possible
- Selected planner = $\text{argmin}_i\, f(G, \theta)_i$
- Experience favors classification over regression

Planner Selection with Graph Neural Networks

Graph neural network

1. Compute a vector representation h_v for each node v. [See next slide]
2. Compute an attention weight α_v for each node v. [See next slide]
3. Form the graph representation $h_G = \sum \alpha_v h_v$
4. Apply a fully connected layer $f(G, \theta) = \text{sigmoid}(W^T h_G)$. [Note: not softmax]

Planner Selection with Graph Neural Networks

GCN [Kipf and Welling 2017]

- Acts like a convolutional network
- Each node v has an initial feature vector $x_v =: h_v^{(0)}$
- For t = 0, …, T-1

$$h_v^{(t+1)} = \sigma\Big(\sum_{u \sim v} \widehat{a}_{vu}\, {W^{(t)}}^\top h_u^{(t)}\Big)$$

- Attention weight is computed as

$$\alpha_v = \mathrm{sigmoid}\Big(w^\top [h_v^{(T)}; h_v^{(0)}]\Big)$$

GG-NN [Li et al. 2016]

- Acts like a recurrent network
- Treat node representation as system state and recurrently update it
- For t = 0, …, T-1

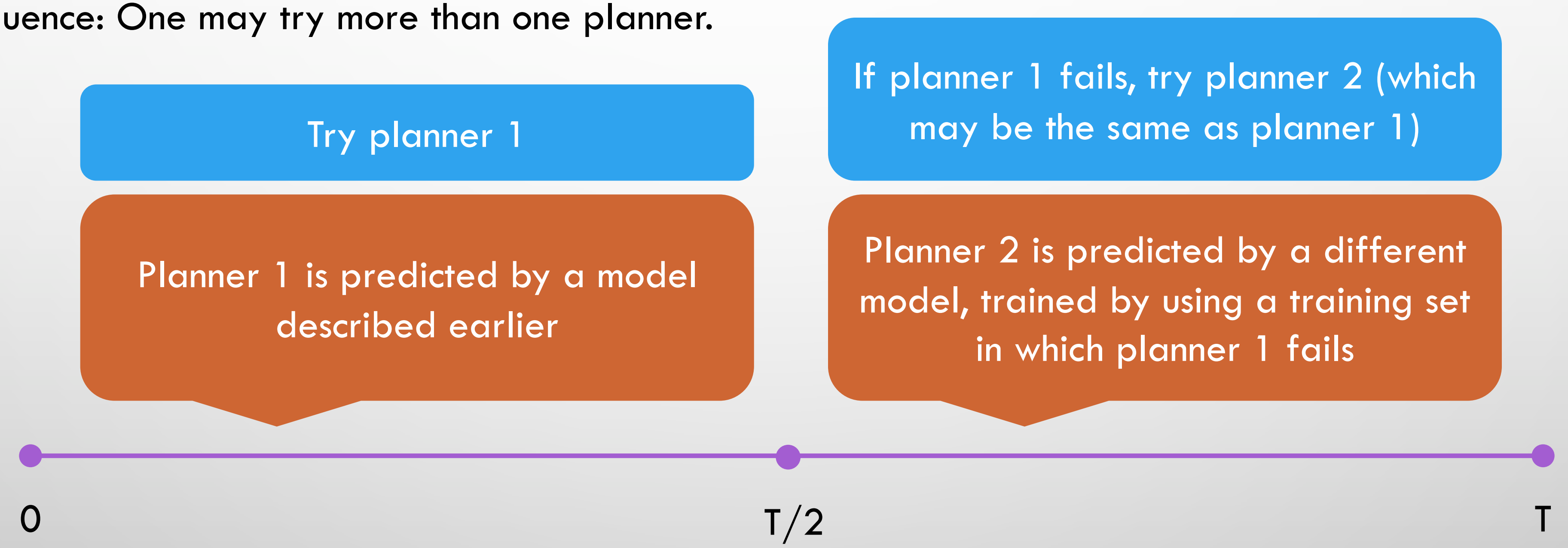
$$h_v^{(t+1)} = \mathrm{GRU}(h_v^{(t)}, m_v^{(t+1)})$$

$$m_v^{(t+1)} = \sum_{u \in \mathrm{in}(v)} W_{\mathrm{in}}^\top h_u^{(t)} + \sum_{u' \in \mathrm{out}(v)} W_{\mathrm{out}}^\top h_{u'}^{(t)}$$

- Computation of α_v is the same as left

Two-Stage Adaptive Scheduling

- Observation: If a planner solves a task in time, often it completes rather quickly.
- Consequence: One may try more than one planner.



Two-Stage Adaptive Scheduling

The second model:

$$f(G, p, \theta) = \text{sigmoid}(W^T h_G + U^T e_p)$$

this part models the condition that planner p fails in the first try

Label =

- 0, if j = p and time of j < T
- 1, if j = p and time of j > T
- 0, if j ≠ p and time of j < T/2
- 1, if j ≠ p and time of j > T/2

Compare with related ideas:

1. Both planners come from the same predictive model
   - If first planner is bad, second can be equally bad
2. More than two planners, same predictive model
   - Same as above
3. More than two planners, each using a different predictive model
   - Training set and label construction become more and more complicated

Data Set

- We compile a data set by using International Planning Competition data
  - Training/validation set: IPC prior to 2018
  - Test set: IPC 2018
- Training/validation split A: fixed; same as Delfi split
- Training/validation split B: random
  - 10 splits preserve domains; 10 splits not
- Each task has a grounded graph and a lifted graph

Portfolio: 17 planners

h2-simpless-dks-celmcut, h2-simpless-dks-cpdbshc900, h2-simpless-dks-900masb50ksccdfp, h2-simpless-oss-900masb50ksbmiasm, h2-simpless-dks-blind, h2-simpless-oss-zopdbsgenetic, h2-simpless-oss-blind, h2-simpless-dks-900masb50ksbmiasm, seq-opt-symba-1, h2-simpless-oss-masginfsccdfp, h2-simpless-dks-900masginfsccdfp, h2-simpless-oss-cpdbshc900, h2-simpless-dks-zopdbsgenetic, simpless-oss-masb50kmiasmdfp, h2-simpless-oss-900masb50ksccdfp, simpless-dks-masb50kmiasmdfp, h2-simpless-oss-celmcut

Results

Delfi split,
single planner

Multiple splits,
single planner &
adaptive scheduling

Table 2: Percentage of solved tasks in the test set and average evaluation time of the method. Delfi split; single planner.

Method	Solved	Eval. Time
Random planner	60.6%	0
Single planner for all tasks	64.8%	0
Complementary2	84.8%	0
Planning-PDBs	82.0%	0
Symbolic-bidirectional	80.0%	0
Enhanced features + random forest	82.1%	0.51s
Image based, CNN, grounded	73.1%	11.00s
Image based, CNN, lifted	86.9%	3.16s
Graph based, GCN, grounded	80.7%	23.15s
Graph based, GCN, lifted	**87.6%**	9.41s
Graph based, GG-NN, grounded	77.9%	14.53s
Graph based, GG-NN, lifted	81.4%	11.44s

Table 3: Percentage of solved tasks in the test set (lifted version). Multiple splits; single planner.

	Domain-preserv.		Random	
	Mean	Std	Mean	Std
Image based, CNN	82.1%	6.6%	86.1%	5.5%
Graph based, GCN	**85.6%**	**5.5%**	**87.2%**	**3.5%**
Graph based, GG-NN	76.6%	5.8%	74.4%	2.7%
Adaptive, GCN	**91.1%**	**3.8%**	**92.1%**	**3.2%**
Adaptive, GG-NN	83.0%	5.8%	86.6%	2.0%

Results

Delfi split,
adaptive scheduling

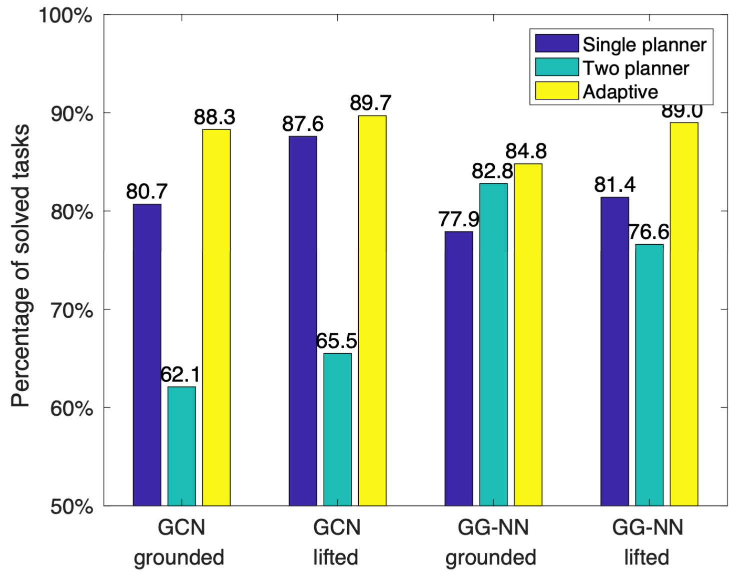


Figure 4: Percentage of solved tasks in the test set. Delfi split; two planners.

Delfi split,
multiple planners

Table 4: Percentage of solved tasks in the test set. Offline method. Delfi split. Compare with performance in Figure 4.

Method	Solved
(Greedy) Best 2 planners from train set	85.5%
(Greedy) Best 3 planners from train set	92.4%
(Greedy) Best 4 planners from train set	89.7%
(Greedy) Best 5 planners from train set	87.6%
(Greedy oracle) Best 2 planners from test set	93.8%
(Greedy oracle) Best 3 planners from test set	93.8%
(Greedy oracle) Best 4 planners from test set	93.1%
(Greedy oracle) Best 5 planners from test set	92.4%
Fast Downward Stone Soup	92.4%

Open Opportunities

- Design graph representations for planning tasks
- Design graph neural networks for special graph structures
- Design scheduling for running more than two planners
- Scale graph neural network training

Paper, Code, and Data Set

- arXiv https://arxiv.org/abs/1811.00210
- Code https://github.com/matenure/GNN_planner
- Data set https://github.com/IBM/IPC-graph-data
- Data set paper https://arxiv.org/abs/1905.06393
    - Patrick Ferber, Tengfei Ma, Siyu Huo, Jie Chen and Michael Katz. IPC: A Benchmark Data Set for Learning with Graph-Structured Data. In ICML 2019 Workshop on Learning and Reasoning with Graph-Structured Data, 2019.