Deep Learning for Cost-Optimal Planning: Task-Dependent Planner Selection

Silvan Sievers[1] Michael Katz[2] Shirin Sohrabi[2]
Horst Samulowitz[2] Patrick Ferber[1]

[1]University of Basel, Switzerland
[2]IBM Research AI, Yorktown Heights, NY, USA

Introduction

Setting

- General purpose of domain-independent planning: solve new planning tasks from unseen domains
- Problem: many domains, many planners – but no single best planner for all domains
- Combine planners in portfolios: parallel (multi-core) or sequential, offline or online schedules, learning setting
  [Gerevini et al. 2011, Helmert et al. 2011, Vallati 2012, Seipp et al. 2012/2015, Seipp et al. 2014, Núñez et al. 2015, Cenamor et al. 2016]
- Most prominent in satisficing planning/learning settings

Motivation

- Can we construct a good portfolio for optimal planning?
- Online portfolios: solve classification task for planner selection
- Good technique for classification tasks: deep learning

Contributions

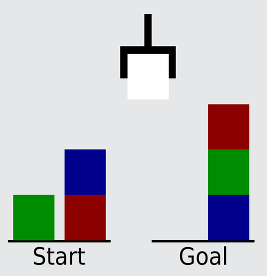
- Deep learning for classification of planning tasks
- Suitable representation of planning tasks
- Proper evaluation of techniques used in Delfi1, winner of last optimal IPC
- Discussion of encountered issues

Planning Task Representation

Example Planning Task

Given in a logic-based description (PDDL):

```
(:action pick-up
 :parameters (?x)
 :precondition
  (and (clear ?x) (ontable ?x) (handempty))
 :effect
  (and (not (ontable ?x))
       (not (clear ?x))
       (not (handempty))
       (holding ?x))
)
```



- Two variables per block: position (4 values) and clearness (binary)
- 729 states

Representing Planning Tasks

Goal:

- Use image convolution for classification
- Requires images serving as planning task representation

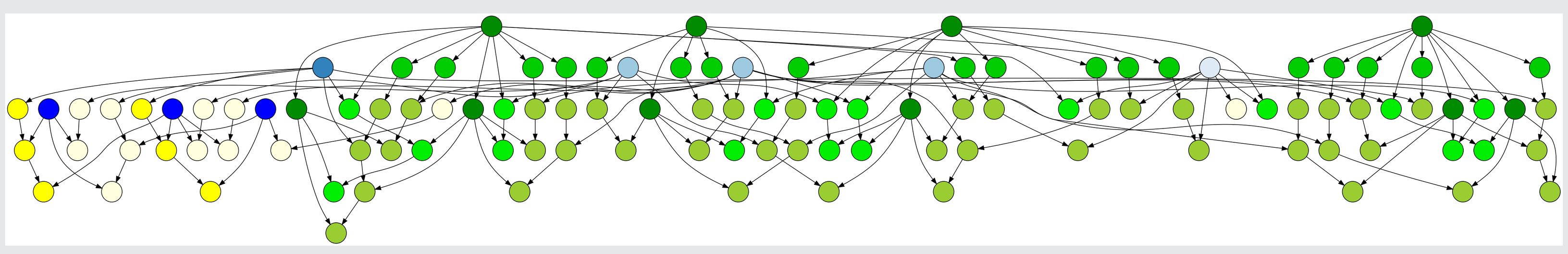
How to obtain representative images?

- SAT/CSP: convert textual problem description into images
- Here: focus on structure of planning tasks

Representative Graphs

Abstract structure graph: compact encoding of the task description

- Nodes for components of the PDDL description (predicates, objects, parameters, etc.)
- Edges to connect components if one is part of another



Representative Images

Conversion of graphs into images:

- Encode adjacency matrix as black&white image
- Turn into grayscale by clustering pixels
- Resize to fixed size

Learning

Performance Representation, CNN Model

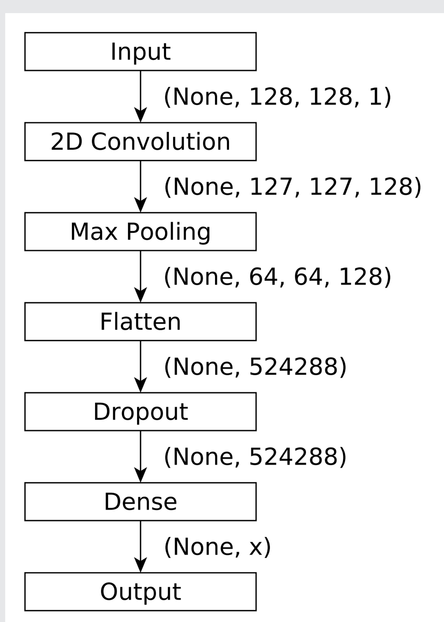
Multilabel classification:

- Binary: predict whether planners solve given task
- Discretized runtime (3 intervals): predict in which interval planners belong

Multilabel regression: predict ...

- Raw runtime
- Normalized runtime

Delfi1: binary



Learning (continued)

Planner Collections

- Fast Downward-based planners from Delfi1
- Those from Delfi1 + additional planners from IPC 2018
- Minimal subset of above to cover training data

Benchmarks

- Training set: domains from IPCs prior 2018
- Test set: domains from IPC 2018

Training Data Separation

- Two training data splits: random vs. domain-preserving random split
- Validation vs. no validation

Choices of Delfi1:

- Hand-crafted domain-preserving split
- No validation for final training (only for hyper parameter optimization)

Total of 48 settings; train 10 models for each setting

Experiments

Results: Comparison of Different Settings

		domain-preserving split				random split			
		validation		no validation		validation		no validation	
		mean	std	mean	std	mean	std	mean	std
time	$\mathcal{C}_D$	50.0	4.4	57.3	1.6	57.5	1.5	57.5	0.0
	$\mathcal{C}_A$	48.7	4.4	49.9	2.7	50.8	3.4	48.8	0.9
	$\mathcal{C}_C$	52.6	3.9	50.5	2.2	50.7	3.9	50.3	2.3
normalized	$\mathcal{C}_D$	50.9	4.4	53.8	2.0	55.4	3.1	54.9	3.1
	$\mathcal{C}_A$	51.8	3.7	50.5	2.6	48.8	1.2	49.3	1.8
	$\mathcal{C}_C$	49.5	5.6	50.2	2.1	50.0	1.3	50.3	1.8
discrete	$\mathcal{C}_D$	49.5	4.0	53.7	5.9	53.9	3.3	54.1	3.0
	$\mathcal{C}_A$	55.4	3.4	52.7	2.2	53.9	3.8	53.7	5.1
	$\mathcal{C}_C$	50.5	1.6	51.6	3.1	58.3	5.2	53.3	1.4
binary	$\mathcal{C}_D$	49.6	4.0	50.2	1.4	52.0	3.3	50.3	1.1
	$\mathcal{C}_A$	50.4	4.7	48.9	1.8	49.9	2.2	49.6	1.5
	$\mathcal{C}_C$	53.4	3.0	49.2	2.2	52.3	2.7	51.7	3.6

	t	*n*	*d*	*b*
time	-	**7**	5	**7**
normalized	4	-	4	**7**
discrete	**7**	**8**	-	**10**
binary	5	5	2	-

	$\mathcal{C}_D$	$\mathcal{C}_A$	$\mathcal{C}_C$
$\mathcal{C}_D$	-	**12**	**10**
$\mathcal{C}_A$	3	-	6
$\mathcal{C}_C$	6	**10**	-

	domain-preserving split		random split	
	validation	no validation	validation	no validation
dom-pres. split & val.	-	5	5	5
dom-pres. split & no val.	**7**	-	2	3
random split & val.	**7**	**10**	-	**8**
random split & no val.	**7**	**9**	3	-

- No domination of any setting over all others
- Delfi1 planner collection significantly better than other two
- Random split somewhat better than domain-preserving split, in particular with validation

Results: Comparison against Baseline

rnd. $\mathcal{C}_D$		rnd. $\mathcal{C}_A$		rnd. $\mathcal{C}_C$		oracle			best		
mean	std	mean	std	mean	std	$\mathcal{C}_D$	$\mathcal{C}_A$	$\mathcal{C}_C$	C2	Sym	Delfi1
42.8	8.3	45.0	8.8	50.3	9.8	67.9	72.1	70.8	58.3	57.1	60.0

- Mostly consistent planner selection within domains
- Mostly better than best individual planners of the collections
- Not as strong as Delfi1 itself

Discussion

Encountered Issues

- Data is not independently identically distributed (i.i.d.)
- Somewhat large variance across different models

Potential Future Work

- More sophisticated networks
- More sophisticated conversion from graphs into images
- Use graphs directly as input to neural networks
- Automatically generate tasks with a certain structure: → i.i.d. distribution of tasks?