(Near)-optimal policies for Probabilistic IPC 2018 domains

Brikena Çelaj

Department of Mathematics and Computer Science
University of Basel

June 2020

Introduction

- The International Planning Competition (IPC) is a competition of state-of-the-art planning systems.
- Quality of the planners is measured in terms of IPC Score.
- Evaluation metric is flawed without optimal upper bound.
- **Thesis aim and motivation** - Contribute to the IPC evaluation metric by finding near-optimal solution of two domains:
  - Academic Advising
  - Chromatic Dice

Academic Advising

- Academic Advising Domain
- Relevance Analysis
- Mapping to Classical Planning
- Results

Academic Advising Domain

Semester	No.	Title	Lecturers	CP
fs	15731-01	Multimedia Retrieval	Roger Weber	6
ss	13548-01	Foundation of Artificial Intelligence	Malte Helmert Thomas Keller	8
fs	45400-01	**Planning and Optimization**	Thomas Keller Gabriele Röger	8
fs	45401-01	Bioinformatics Algorithms	Volker Roth	4
ss	17165-01	**Machine Learning**	Volker Roth	8
ss	10948-01	Theory of Computer Science	Gabriele Röger	8

Academic Advising Domain

Semester	No.	Title	Lecturers	CP
fs	15731-01	Multimedia Retrieval	Roger Weber	6
ss	13548-01	Foundation of Artificial Intelligence	Malte Helmert Thomas Keller	8
fs	45400-01	**Planning and Optimization**	Thomas Keller Gabriele Röger	8
fs	45401-01	Bioinformatics Algorithms	Volker Roth	4
ss	17165-01	**Machine Learning**	Volker Roth	8
ss	10948-01	Theory of Computer Science	Gabriele Röger	8

Prerequisite
Theory of Computer Science
Foundation of Artificial Intelligence

Academic Advising Domain

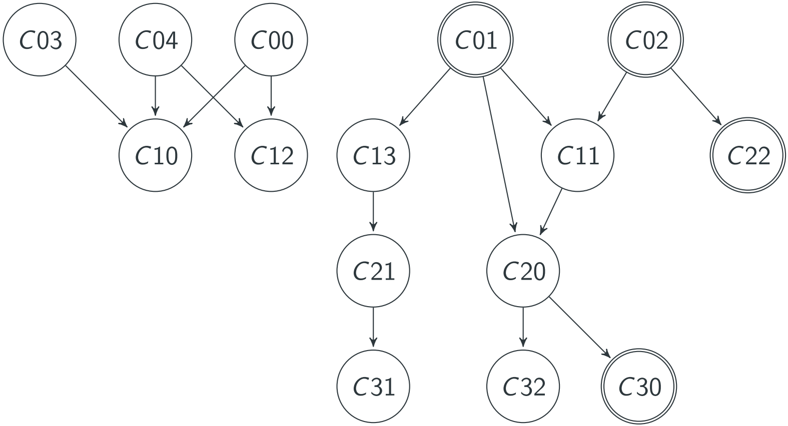
- The smallest instances has more than a trillion states.
- The hardest instance has around 10^{167} states and
- The hardest instance has around 10^{12} actions.
- First step toward solution - **Relevance Analysis**!

Relevance Analysis

1. An instance is represented by directed acyclic graph (DAG)
   - Nodes $\rightarrow$ courses
   - Edges $\rightarrow$ connect course to its prerequisites

Relevance Analysis

Example: Academic Advising Instance



Relevance Analysis

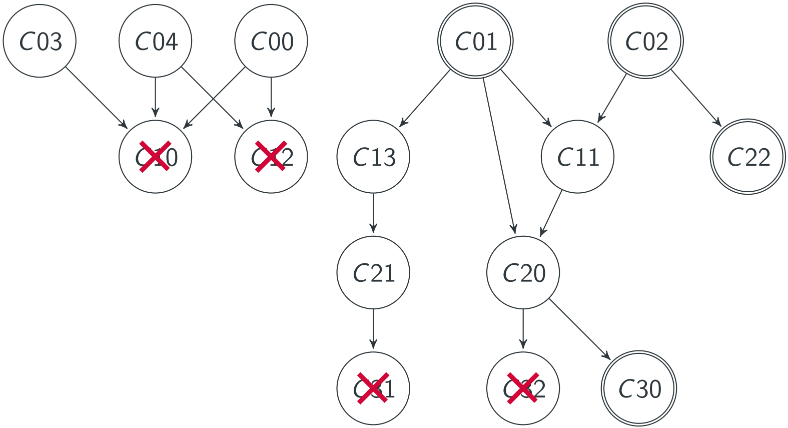
1. An instance is represented by directed acyclic graph (DAG)
   - Nodes → courses
   - Edges → connect course to its prerequisites
2. In each iteration find the leaves of the graph

Relevance Analysis

1. An instance is represented by directed acyclic graph (DAG)
   - Nodes → courses
   - Edges → connect course to its prerequisites
2. In each iteration find the leaves of the graph
3. Prune any leaf that it not in program required courses

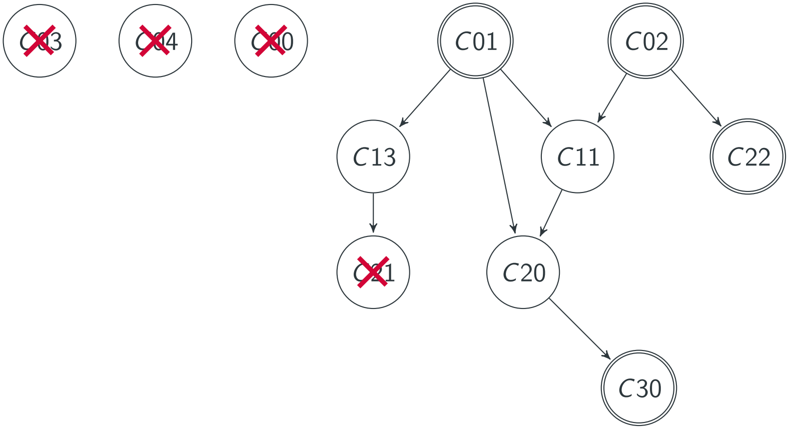
Relevance Analysis

First iteration



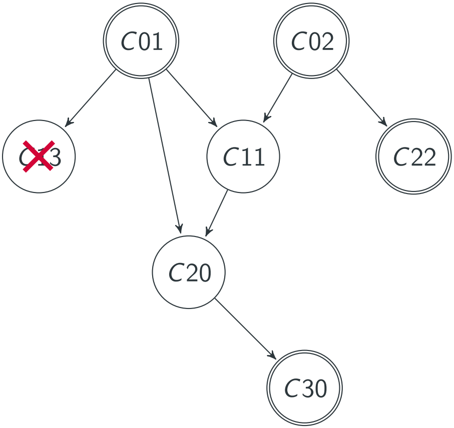
Relevance Analysis

Second iteration

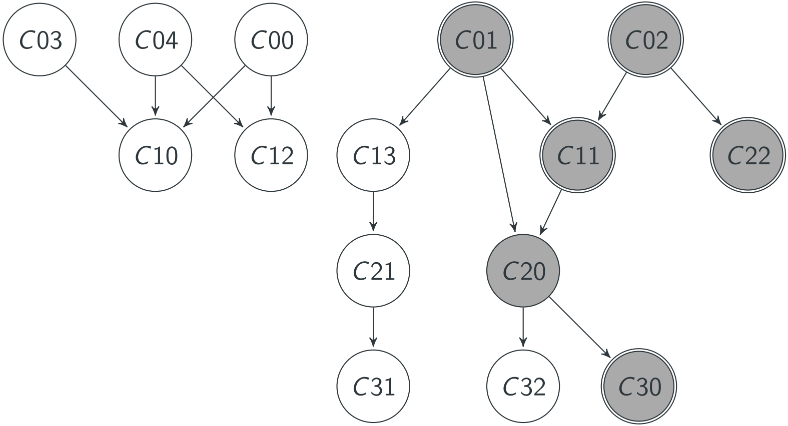


Relevance Analysis

Third iteration



Relevance Analysis



Relevance Analysis

- After shrinking, in average, we have half the number of courses.
- The hardest instance now has around 10^{46} states and 10^{9} actions.
- Still too large to find an optimal solution!
- Next step: **Mapping to Classical Planning**!

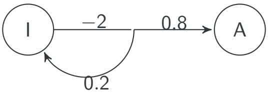
Mapping to Classical Planning

In Academic Advising domain:

- There are no dead ends.
- If horizon h is infinite, any optimal policy will try to reach a state where the program requirement is complete.
- If concurrency σ is one, we have two outcomes for each action (succeed or fail).

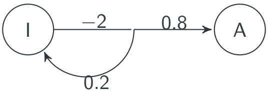
Mapping to Classical Planning

Assumption: $h = \infty$, $\sigma = 1$.

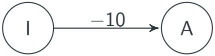


Mapping to Classical Planning

Assumption: $h = \infty$, $\sigma = 1$.

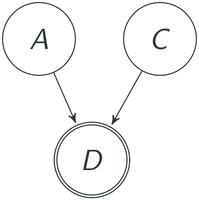


$$\Downarrow$$



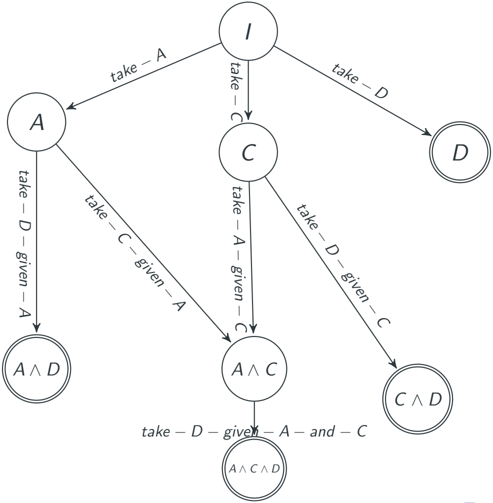
Mapping to Classical Planning

Academic Advising domain example



Mapping to Classical Planning

Academic Advising domain converted into a classical domain



Mapping to Classical Planning

Theorem

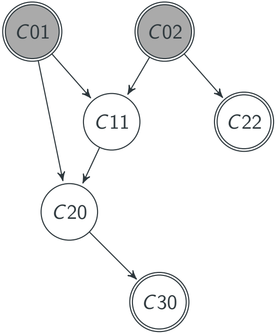
For all Academic Advising instances, where $\sigma = 1$ and $h = \infty$, and π, an optimal plan for the induced Classical Planning Task, we have

$$V_*(s_0, \infty) = -cost(\pi)$$

Mapping to Classical Planning

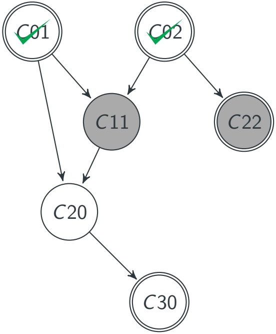
- In most of the instances, $\sigma > 1$!
- **Question:** Why it is not simple to map to Classical Planning when $\sigma > 1$?
- **Answer:** We no longer have only two outcomes (succeed or fail)!
- **Solution:** Ignore that courses can be taken in parallel, and divide cost of the plan by σ.

Example: $\sigma = 2$



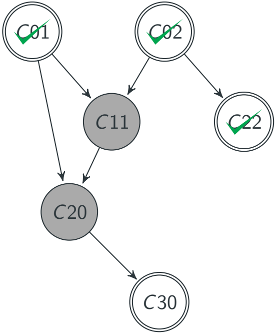
- Assume we always perform as many actions as concurrency,
- Assume we take the courses where all the prerequisites are already passed.

Example: $\sigma = 2$



- Assume we always perform as many actions as concurrency,
- Assume we take the courses where all the prerequisites are already passed.

Example: $\sigma = 2$



- Assume we always perform as many actions as concurrency,
- Assume we take the courses where all the prerequisites are already passed.

Mapping to Classical Planning

Theorem

For all Academic Advising instances, where $\sigma > 1$ and $h = \infty$, and π, an optimal plan for the induced Classical Planning Task, we have

$$V_*(s_0, \infty) \geq -\frac{cost(\pi)}{\sigma}$$

Mapping to Classical Planning

- In practice, the horizon is finite!
- If we don't expect to achieve the goal in time, it is better to do nothing instead of applying an operator.
- Applying an operator incurs cost.

Mapping to Classical Planning

- **Question:** Can we deal with cases where $h \neq \infty$?
- **Answer:** No, but we can come up with good estimates!
- **Solution:** Comparison of the optimal policy with noop policy!

Mapping to Classical Planning

Result

For all Academic Advising instances, where $h \neq \infty$, and π, an optimal plan for the induced Classical Planning Task, we have

$$V_*(s_0, h) \approx \max\left(-\frac{cost(\pi)}{\sigma}, h \cdot penalty\right)$$

Results

Instance	Concurrency	Horizon	Our Results	SOGBOFA	PROST-DD
01	1	20	**-25**	-48.4	-47.13
02	2	20	-15	-63.13	-49.93
03	1	20	**-20**	-35.2	-37.8
04	1	20	**-21.87**	-79.18	-39.48
05	2	20	-26.63	-100.0	-90.12
06	1	30	**-55**	-82.86	-83.46
07	2	30	-40.98	-150.0	-188.96
08	2	30	-30.41	-150.0	-182.84
09	1	30	**-25**	-66.53	-86.33
10	2	30	-42	-150.0	-200.24
11	3	40	-34.09	-200.0	
12	2	40	-36.51	-200.0	-215.2
13	2	40	-42.57	-200.0	-282.48
14	3	40	-44.24	-200.0	
15	2	40	-53.09	-200.0	
16	3	50	-52.79	-250.0	
17	4	50	-41.8	-250.0	
18	3	50	-44.74	-250.0	
19	4	50	-45.59	-250.0	
20	5	50	-35.35	-250.0	

Chromatic Dice

- Chromatic Dice Domain
- Implementation Strategy
- Chromatic Dice Structure
- Near-optimal Strategy
- Results

Yahtzee

Chromatic Dice Domain

Chromatic Dice is similar to Yahtzee with some differences:

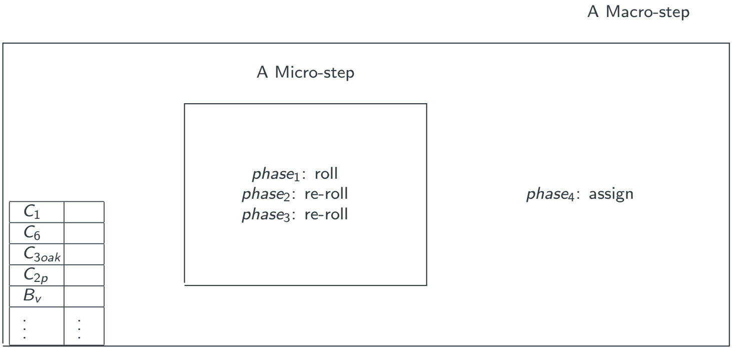
1. Dices are two-dimensional(values and colors).
2. There are more categories.
3. There are two different type of bonuses.

Chromatic Dice Structure

Chromatic Dice structure looks as follow:

- The state space can be structured into rounds
- Each round consists of 3 roll operators and 1 assign operator
- Roll operators roll (a subset of) the dice
- Assign operators select an unassigned category and yield a reward
- The number of rounds is equal to the number of categories

Chromatic Dice Architecture



Macro-steps

- The dice remain the same while we perform Macro-steps.

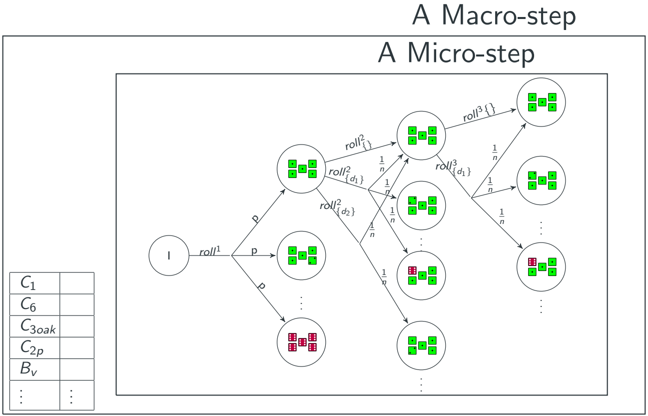
Macro-step State Space

- A naive representation considers all information of the scorecard:
  Yahtzee: 2^{37} states **Chromatic Dice:** 2^{74} states

- Computation of an optimal strategy possible with much more compact representation based on
  1. which category is still available (is the category taken or not)
  2. which is the bonus level of the upper and middle section.

  Yahtzee: 2^{19} states **Chromatic Dice:** 2^{29} states

Micro-steps



- The categories and level of the bonuses remain the same while we perform Micro-steps.

Micro-steps

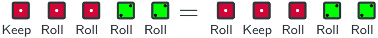
We can reduce the problem by:

- Shrinking the Micro-step state space.

  $\rightarrow$ 3 $\times$ , 2 $\times$ 

- Shrinking the Micro-step edges.

  $\rightarrow$ Roll 0, 1, 2 or 3  and 0, 1 or 2 

Micro-step State Space and Edges

- Naive representation:

	Yahtzee	Chromatic Dice
States	2^{12}	2^{24}
Edges	2^{13}	2^{24}

- Compact representation:

	Yahtzee	Chromatic Dice
States	2^{10}	2^{18}
Edges	2^{12}	2^{23}

Backtracking Method

- Our state space is a DAG.
- **Benefit of DAG:** We compute the policy by using backtracking method.
- **How?** - Initialize the state values on the last layer with 0 and go backward up to the initial state by replacing all value states.

Optimal Results

Instances	Macro-steps	Instance edges	Optimal Results	SOGBOFA	PROST-DD
01	$\approx 2^{11}$	$\approx 2^{14}$	**72.51**	48.89	37.87
02	$\approx 2^{15}$	$\approx 2^{14}$	160.03	**182.17**	71.69
03	$\approx 2^{16}$	$\approx 2^{14}$	**216.88**	142.21	108.61
04	$\approx 2^{19}$	$\approx 2^{14}$	**279.42**	247.52	119.45
05	$\approx 2^{13}$	$\approx 2^{18}$	**154.40**	118.31	108.0
06	$\approx 2^{21}$	$\approx 2^{18}$	-	325.84	150.47
07	$\approx 2^{25}$	$\approx 2^{18}$	-	402.53	203.36
08	$\approx 2^{14}$	$\approx 2^{21}$	**147.17**	120.29	94.53
09	$\approx 2^{22}$	$\approx 2^{21}$	-	313.79	144.49
10	$\approx 2^{26}$	$\approx 2^{21}$	-	370.13	193.27
11	$\approx 2^{15}$	$\approx 2^{23}$	**155.48**	108.43	86.93
12	$\approx 2^{27}$	$\approx 2^{23}$	-	355.01	162.13
13	$\approx 2^{16}$	$\approx 2^{24}$	-	115.63	86.2
14	$\approx 2^{21}$	$\approx 2^{24}$	-	204.92	74.63
15	$\approx 2^{29}$	$\approx 2^{25}$	-	402.25	159.48
16	$\approx 2^{29}$	$\approx 2^{14}$	-	441.05	227.68
17	$\approx 2^{29}$	$\approx 2^{14}$	-	414.27	236.28
18	$\approx 2^{29}$	$\approx 2^{14}$	-	450.97	211.73
19	$\approx 2^{29}$	$\approx 2^{14}$	-	423.39	205.24
20	$\approx 2^{29}$	$\approx 2^{14}$	-	452.44	208.96

Near-optimal Strategy

- We can find the optimal solution only for small instances because the state space is large.
- For harder instances, finding an optimal solution is intractable in practice.
- **Solution**: Heuristic Strategy

Near-optimal Strategy

We generalize the idea of *cost partitioning* and apply it for FH-MDPs, called *reward partitioning*, as follow:

- We divide an instance into any number of sub-instances.
- Each category yield the reward in only one of the sub-instances, while in all others the reward is 0.
- The sum of solution rewards of each sub-instances is an *admissible* expected reward.

Near-optimal Strategy in Practice

- **Drawback**: The horizon is still the same!
- **Drawback**: The size of MDP is almost the same, therefore,it is hard to compute in practice!
- **Solution**: Near optimal solution without the guarantee of admissibility!
- **Solution**: Decrease the horizon to the number of categories that are considered in the sub-instance

Heuristic Results

Instances	|Sub-instances|	Our Results	SOGBOFA	PROST-DD
06	3	**389 .95**	325.84	150.47
07	3	**496.29**	402.53	203.36
09	3	**395.49**	313.79	144.49
10	3	**489 .76**	370.13	193.27
12	3	**480.70**	355.01	162.13
13	3	**225.70**	115.63	86.2
14	3	**297.05**	204.92	74.63
15	3	**500.50**	402.25	159.48
16	2	406.43	**441.05**	227.68
17	2	409.32	**414.27**	236.28
18	2	381.33	**450.97**	211.73
19	2	401.63	**423.39**	205.24
20	2	430.44	**452.44**	208.96

Thank you!