Refinement Strategies for Counterexample-Guided Cartesian Abstraction Refinement

Martin Zumsteg <mar.zumsteg@stud.unibas.ch>

Faculty of Science at the University of Basel

July 26, 2019

Contents

- Background
  - Definitions
  - SCP
  - CEGAR
- Methods
  - Refinement Strategies
  - Experiments
  - Evaluation
- Results
- Conclusion

Transition Systems

Definition

A **transition system** is a 6-tuple

$$\mathcal{T} = \langle S, \mathcal{O}, \text{cost}, T, s_0, S_* \rangle$$

With associated **variables** $\mathcal{V} = \{v_1, ..., v_n\}$ generating **states**

$$S = \{\{v_1 \rightarrow d_1, ..., v_n \rightarrow d_n\} \mid d_i \in \text{dom}(v_i)\}$$

Abstractions

Abstractions are themselves transition systems
Each abstract state contains **at least one** concrete state
which is itself contained in **exactly one** abstract state:

$$s \in S \Rightarrow s \in [s] \quad \text{where } [s] \text{ is the abstract lookup}$$

For **cartesian abstractions** the states become:

$$S \subset \{\{v_1 \rightarrow d_1, ..., v_n \rightarrow d_n\} \mid d_i \in \mathcal{P}(\mathrm{dom}(v_i))\}$$

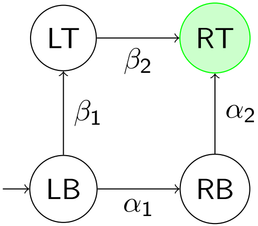
Counterexample-guided abstraction refinement

Method of **incrementally** refining abstractions

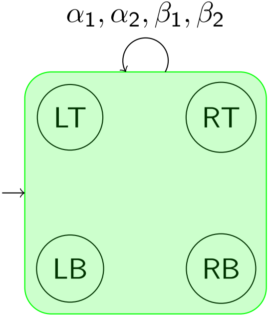
1. Initialize with trivial abstraction
2. Find a solution for the current abstraction
   If there is none, end refinement (unsolvable)
3. Apply solution to concrete planning problem
   If a goal is reached, end refinement (trivially solvable)
4. Otherwise, find a flaw in the abstraction
5. Apply **some** split to avoid this flaw
6. Optionally continue refinement with step 2

Counterexample-guided abstraction refinement

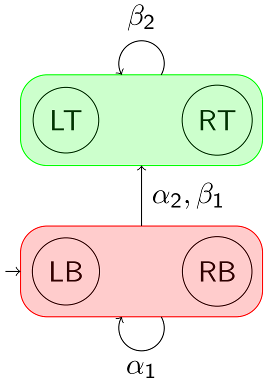
$$\mathcal{V} = \{X, Y\}$$



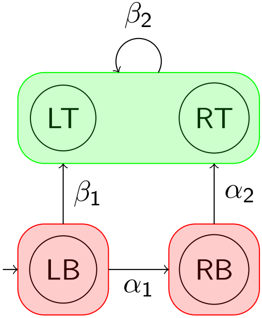
Counterexample-guided abstraction refinement



Counterexample-guided abstraction refinement



Counterexample-guided abstraction refinement



Subtasks and SCP

Generate multiple abstractions using subtasks

- One subtask per **goal fact**
- One subtask per **fact landmark**

Each abstraction only solves one subtask

Subtasks and SCP

Generate multiple abstractions using subtasks

- One subtask per **goal fact**
- One subtask per **fact landmark**

Each abstraction only solves one subtask

Compute a saturated cost partitioning over abstractions

- Distributes operator costs to enforce additivity
- Diverse abstractions result in better heuristic estimates

Refinement Strategies

Extending Fast Downward with **new refinement strategies**:
For each base strategy: `MAX` and `MIN` variants

Refinement Strategies

Extending Fast Downward with **new refinement strategies**:
For each base strategy: `MAX` and `MIN` variants

- `CG`: Based on the causal graph index computed by the planner.

Refinement Strategies

Extending Fast Downward with **new refinement strategies**:
For each base strategy: `MAX` and `MIN` variants

- `CG`: Based on the causal graph index computed by the planner.
- `GOAL_DIST`: Computes the average goal distance over all states where the goal can be reached.
  Aims for abstractions with few heuristic collisions.

Refinement Strategies

Extending Fast Downward with **new refinement strategies**:
For each base strategy: `MAX` and `MIN` variants

- `CG`: Based on the causal graph index computed by the planner.
- `GOAL_DIST`: Computes the average goal distance over all states where the goal can be reached.
  Aims for abstractions with few heuristic collisions.
- `HIGHER_DIST`: Counts the number of abstract states which have an increased goal distance.
  Highly similar to `GOAL_DIST`

Refinement Strategies

Extending Fast Downward with **new refinement strategies**:
For each base strategy: `MAX` and `MIN` variants

- `CG`: Based on the causal graph index computed by the planner.
- `GOAL_DIST`: Computes the average goal distance over all states where the goal can be reached.
  Aims for abstractions with few heuristic collisions.
- `HIGHER_DIST`: Counts the number of abstract states which have an increased goal distance.
  Highly similar to `GOAL_DIST`
- `ACTIVE_OPS`: Counts the number of operators which induce non-looping transitions.

Experiments

Evaluate strategies in different contexts:

- Based on the original planning task
    - One abstraction, used as heuristic

Experiments

Evaluate strategies in different contexts:

- Based on the original planning task
  - One abstraction, used as heuristic
- Using subtasks (single-order SCP)
  - Combination of abstractions (one per subtask)

Experiments

Evaluate strategies in different contexts:

- Based on the original planning task
  One abstraction, used as heuristic
- Using subtasks (single-order SCP)
  Combination of abstractions (one per subtask)
- Saturated cost partitioning
  Multiple orders over subtasks

Evaluation

Measure performance of each strategy via

- Time to construct abstraction
- Number of expansions during planning

Evaluation

Measure performance of each strategy via

- Time to construct abstraction
- Number of expansions during planning

Estimate of the freedom of the strategy

- Average number of splits to pick from
- Average number of distinctly valued splits

Evaluation

Measure performance of each strategy via

- Time to construct abstraction
- Number of expansions during planning

Estimate of the freedom of the strategy

- Average number of splits to pick from
- Average number of distinctly valued splits

Progress of average goal distance (`GOAL_DIST`)

Evaluation

Measure performance of each strategy via

- Time to construct abstraction
- Number of expansions during planning

Estimate of the freedom of the strategy

- Average number of splits to pick from
- Average number of distinctly valued splits

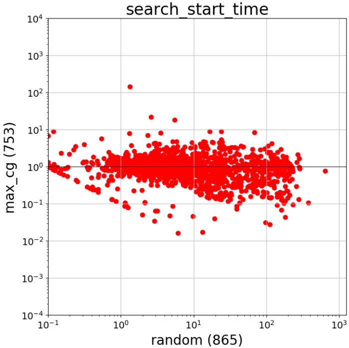
Progress of average goal distance (`GOAL_DIST`)

Attempt to predict the best strategy by problem

- Based on attributes of the SAS+ problem
- Using linear algebra or Gaussian estimators

Results (Time)

Simple strategies: time requirement similar to `RANDOM`

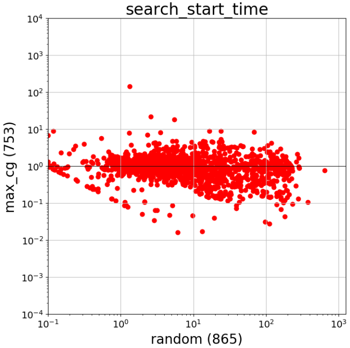


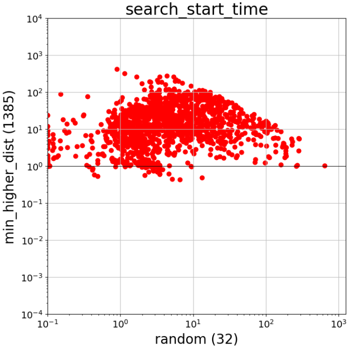
Results (Time)

Simple strategies: time requirement similar to `RANDOM`

Complex strategies: increasingly slower than `RANDOM`

Cutoff at 1800 seconds (overall time limit)

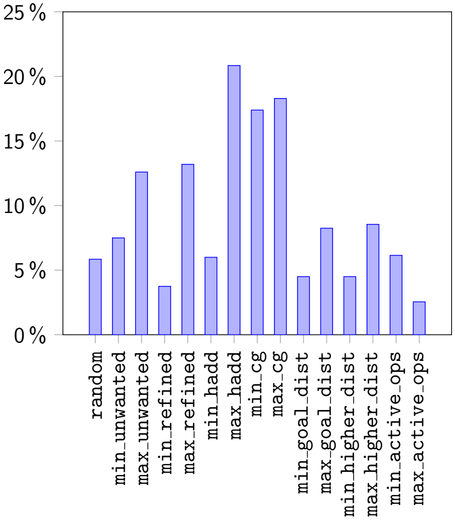




Results (Expansions)

Original Task

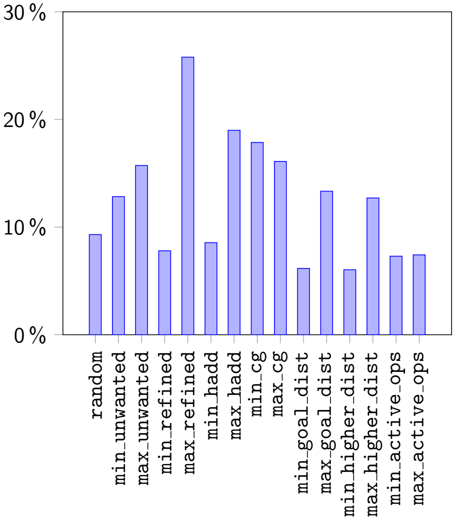
- Generally MAX is better, except for ACTIVE_OPS
- Best strategy is MAX_HADD, followed closely by CG



Results (Expansions)

Using subtasks

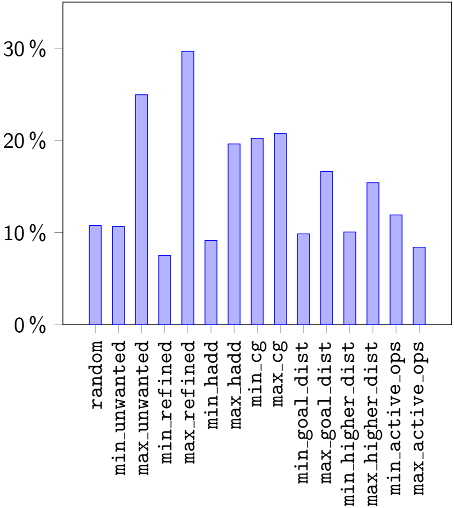
- Medium improvements to most strategies
- New best strategy is MAX_REFINED
- Previous best strategies become worse



Results (Expansions)

Using SCP

- Small improvements overall
- Large improvement of
  MAX_UNWANTED



Results (Strategy Freedom)

- **options** is the upper bound for **distinct**
- Ratings for **options** only vary slightly overall
- No apparent correlation to performance
- MAX_HADD has lower **distinct** rating than CG
- Low **distinct** rating indicates use of the tie-breaker

	$\mu_{\textbf{distinct}}$	$\sigma_{\textbf{distinct}}$	$\mu_{\textbf{options}}$	$\sigma_{\textbf{options}}$
RANDOM	1.18	0.167	1.18	0.167
MAX_HADD	1.14	0.150	1.22	0.215
MAX_CG	1.24	0.256	1.24	0.256
MAX_GOAL_DIST	1.06	0.058	1.18	0.160
MAX_HIGHER_DIST	1.02	0.021	1.18	0.163
MIN_ACTIVE_OPS	1.00	0.022	1.18	0.168
MAX_ACTIVE_OPS	1.01	0.069	1.18	0.169

Results (Strategy Freedom)

- **options** is the upper bound for **distinct**
- Ratings for **options** only vary slightly overall
- No apparent correlation to performance
- `MAX_HADD` has lower **distinct** rating than `CG`
- Low **distinct** rating indicates use of the tie-breaker

	$\mu_{\textbf{distinct}}$	$\sigma_{\textbf{distinct}}$	$\mu_{\textbf{options}}$	$\sigma_{\textbf{options}}$
RANDOM	1.18	0.167	1.18	0.167
MAX_HADD	1.14	0.150	1.22	0.215
MAX_CG	1.24	0.256	1.24	0.256
MAX_GOAL_DIST	1.06	0.058	1.18	0.160
MAX_HIGHER_DIST	1.02	0.021	1.18	0.163
MIN_ACTIVE_OPS	1.00	0.022	1.18	0.168
MAX_ACTIVE_OPS	1.01	0.069	1.18	0.169

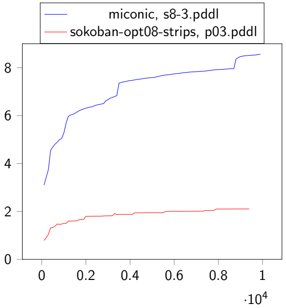
Results (Strategy Freedom)

- **options** is the upper bound for **distinct**
- Ratings for **options** only vary slightly overall
- No apparent correlation to performance
- `MAX_HADD` has lower **distinct** rating than `CG`
- Low **distinct** rating indicates use of the **tie-breaker**

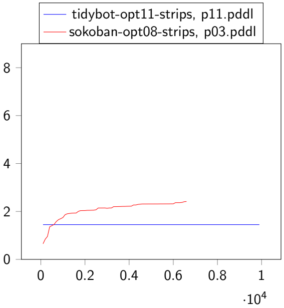
	$\mu_{\textbf{distinct}}$	$\sigma_{\textbf{distinct}}$	$\mu_{\textbf{options}}$	$\sigma_{\textbf{options}}$
RANDOM	1.18	0.167	1.18	0.167
MAX_HADD	1.14	0.150	1.22	0.215
MAX_CG	1.24	0.256	1.24	0.256
MAX_GOAL_DIST	1.06	0.058	1.18	0.160
MAX_HIGHER_DIST	1.02	0.021	1.18	0.163
MIN_ACTIVE_OPS	1.00	0.022	1.18	0.168
MAX_ACTIVE_OPS	1.01	0.069	1.18	0.169

Results (Goal Distance)

Follows a logarithmic curve, stagnating near the end
Strategies perform best if they miss their goal



MIN_GOAL_DIST (best)



MAX_GOAL_DIST (best)

Results (Prediction)

- Compare expansions caused by a prediction method
  - Influenced by selection of test set
- Neither prediction method consistently beats the best strategy
- Gaussian is almost always better than linear

	least exp.	most exp.	best strategy	MVND	linear
original task	46038868	131558133	64292474	70275225	116803152
	1	2.858	1.396	1.526	2.537
			1	1.093	1.817
subtasks	30570863	64358053	36005074	60324594	61799436
	1	2.105	1.178	1.973	2.022
			1	1.675	1.716
SCP	67968544	85726053	71164310	77393698	74462557
	1	1.261	1.047	1.139	1.096
			1	1.088	1.046

Conclusion

- No new heuristic significantly better
  - Effective only in specific domains
  - Best heuristic strongly depends on experiment
- Freedom during refinement does not predict performance
- Prediction approaches picking best strategy overall

Future work

- Make use of tie-breaking strategy
  - Combining multiple refinement strategies
- Use multiple refinement strategies for SCP
- More parameters to predict the best strategy

Questions?