Shoma Endo, Masataro Asai, Alex Fukunaga

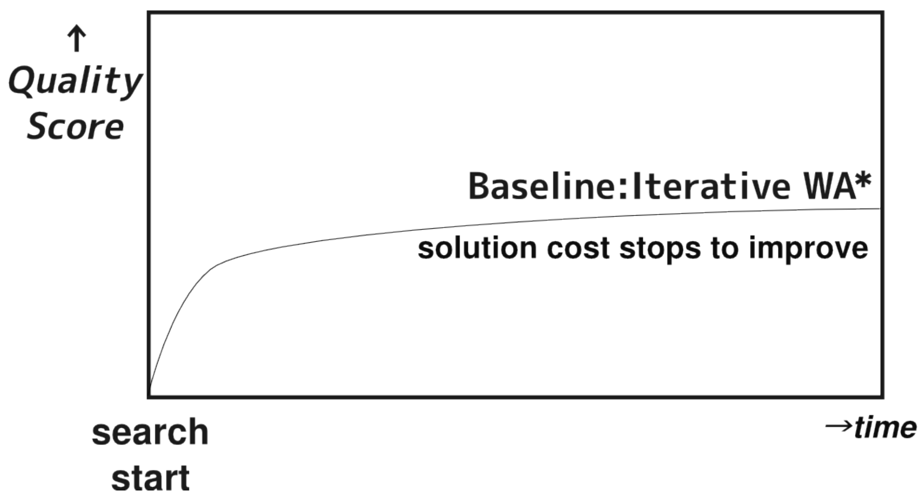
University of Tokyo

Presentation available on

https://guicho271828.github.io/2016-6-13-hsdip/

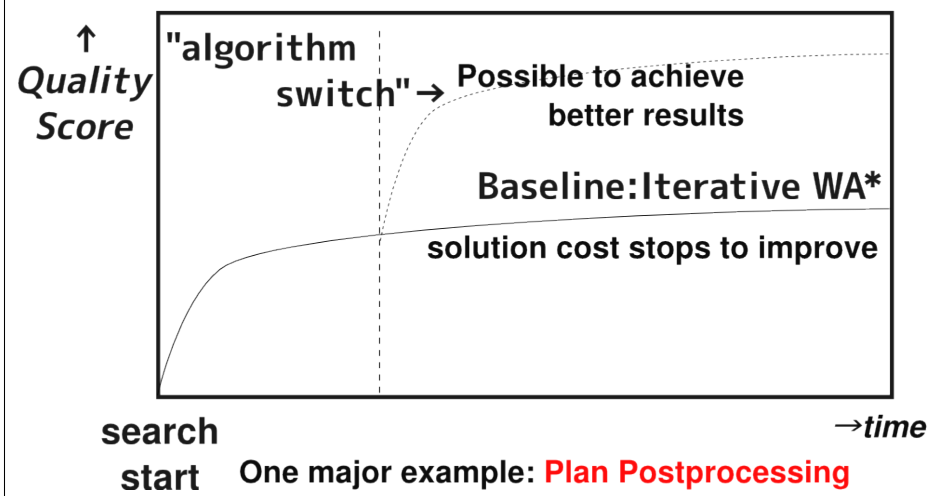
1 Satisficing Planning + Online Cost Refinements

Typical Characteristics of Anytime Algorithms

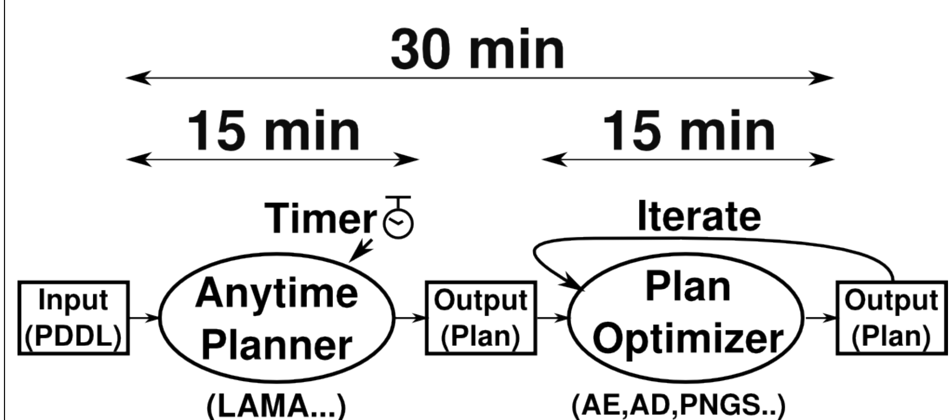


2 Satisficing Planning + Algorithm Switch

Typical Characteristics of Anytime Algorithms



3 Main Topic of the paper: Plan Postprocessing



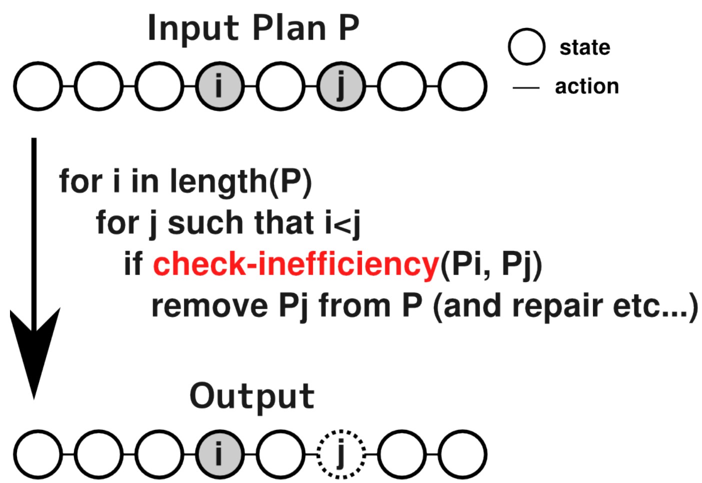
4 Various Postprocessing Systems

- Aras (Nakhost and Muller 2010)
  - Action Elimination(**AE**) — remove trivially redundant actions
  - Plan Neighborhood Graph Search(**PNGS**) — replan on neighborhood graph
- Action Dependency (**AD**) (Chrpa, McCluskey and Osborne 2012)
  - Remove actions of inverse effects etc.
- Block Deordering (BDPO2) (Siddiqui and Haslum 2013;2015)
  - convert the input to partial order blocks, replan each block
- Anytime Iterative Refinement of Solution (**AIRS**) (Estrem and Krebsbach 2012)

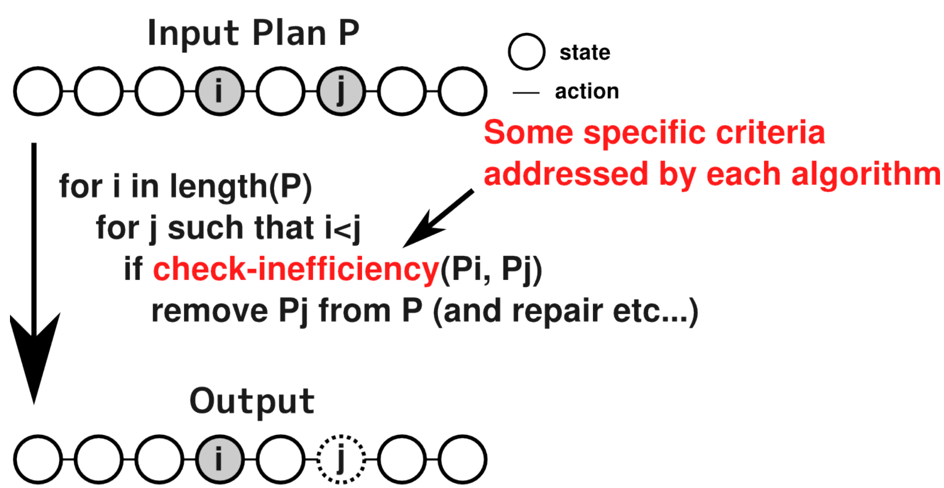
5 Two groups of Optimization Algorithms

Polytime Optimizer	Search-based Optimizer
Based on the direct analysis of a given plan	External solver refines a partial segment in a plan

5.1 Polytime Optimizers



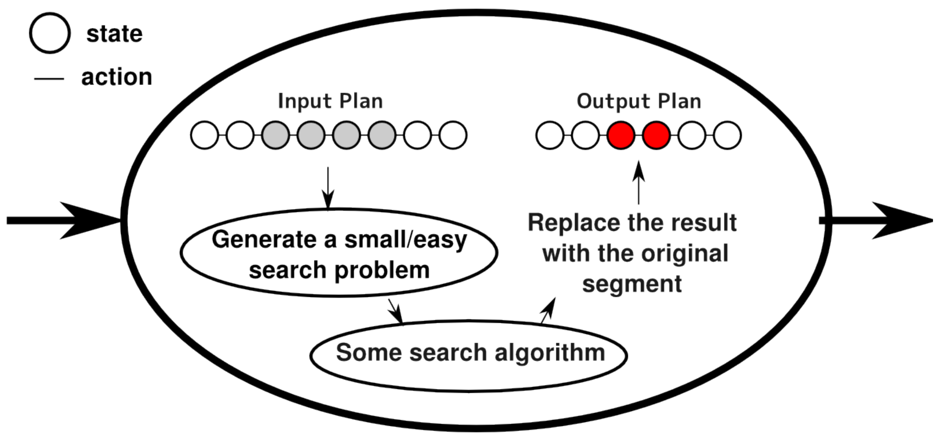
5.2 Polytime Optimizers



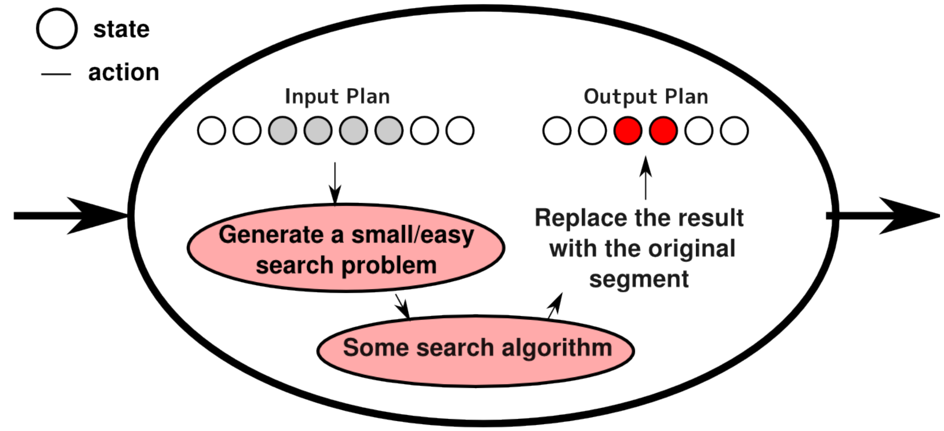
5.3 Two groups of Optimization Algorithms

Polytime Optimizer	Search-based Optimizer
Based on the direct	External solver refines
analysis of a given plan	a partial segment in a plan
AE : O(n^2)	
AD : O(n^2)	
n: plan length	

5.4 Search-based Optimizers



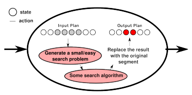
5.5 Search-based Optimizers



5.6 Two groups of Optimization Algorithms

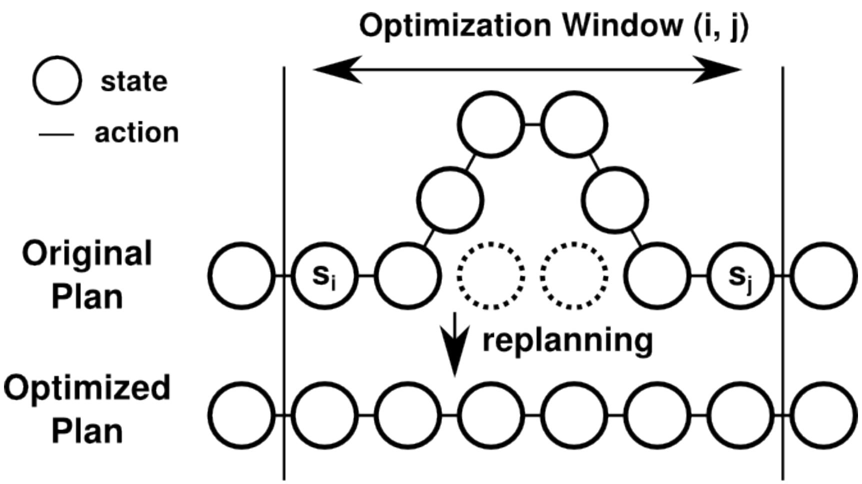
Polytime Optimizer	Search-based Optimizer
Based on the direct	External solver refines
analysis of a given plan	a partial segment in a plan
AE : O(n^2)	**PNGS**
AD : O(n^2)	**AIRS**
n: plan length	BDPO2

5.7 Search-based Optimizers Taxonomy

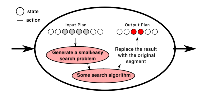


	Subproblems are based on	Subproblems are solved with
PNGS	neighborhood graph	
	(search space around the plan)	
AIRS	windows (ordered by Δ cost/Δ h)	
BDPO2	windows (block decomposition)	

5.8 Window-based optimization



5.9 Search-based Optimizers Taxonomy



	Subproblems are based on	Subproblems are s
PNGS	neighborhood graph	Blind unit-cost A*
	(search space around the plan)	or Backward Brea
AIRS	windows (ordered by Δ cost/Δ h)	Blind Bidirectiona
BDPO2	windows (block decomposition)	Admissible A*
		or PNGS (layered

6 Open Questions

What is the *baseline* of various search-based optimizers?

i.e. Are these ***complex*** algorithms necessary?

How to *combine* Polytime and Search-based Optimizers?

Also, how effective is it to combine them?

7 Various Minor Differences in Search-based Optimizers

	Subproblems are based on	Subproblems are solved with
PNGS	**neighborhood graph**	**Blind** unit-cost A*
	(search space around the plan)	or **Backward Breadths First**
AIRS	windows (**ordered by △ cost/△ h**)	**Blind Bidirectional**
BDPO2	windows (**block deordering**)	Admissible A*
		or **PNGS** (layered approach)

→ **The amount of observation from the experiments are limited**

8 *Simplest baseline* of search-based optimizers : *R-WIN*

	Subproblems are based on	Subproblems are solved with
PNGS	neighborhood graph	Blind unit-cost A*
	(search space around the plan)	or Backward Breadths First
AIRS	windows (ordered by △ cost/△ h)	Blind Bidirectional
BDPO2	windows (block deordering)	Admissible A*
		or PNGS (layered approach)
R-WIN	**windows** (**random**)	**Admissible A* + LMcut**

9 Evaluate them in an "*Equal Condition*"

	Subproblems are based on	Subproblems are solved with
PNGS	neighborhood graph	**Admissible A* + LMcut**
	(search space around the plan)	
AIRS	windows (ordered by △ cost/△ h)	**Admissible A* + LMcut**
BDPO2	windows (block deordering)	**Admissible A* + LMcut**
R-WIN	windows (random)	**Admissible A* + LMcut**

10 Evaluation

39 IPC domains

Optimize the 15 min, 2GB results of LAMA

Resource 15 min, 2GB

Participants AD, AE, AIRS, PNGS, R-WIN

(BDPO2 is not tested)

Algorithm	Harmonic Means of Ratios
LAMA(15min)	100%
LAMA(30min)	%
AE	%
AD	%
AIRS	%
PNGS	%
R-WIN	%

11 Results

39 IPC domains

Optimize the 15 min, 2GB results of LAMA

Resource 15 min, 2GB

Participants AD, AE, AIRS, PNGS, R-WIN

(BDPO2 is not tested)

Algorithm	Harmonic Means of Ratios
LAMA(15min)	100%
LAMA(30min)	99.3%
AE	98.4%
AD	97.4%
AIRS	97.9%
PNGS	96.0%
R-WIN	**95.9%**

Complex tweaks did not outperform the simplest variant

12

Sadly "Improvements" did not outperform the simplest baseline

BDPO2 unconfirmed

13

Now let's reliably improve upon the baseline

14 Improve AIRS to outperform R-WIN: CH-WIN

Why we chose AIRS?

→ Easy to implement, minimum diff from R-WIN

Problem in AIRS:

Original AIRS is tested only on

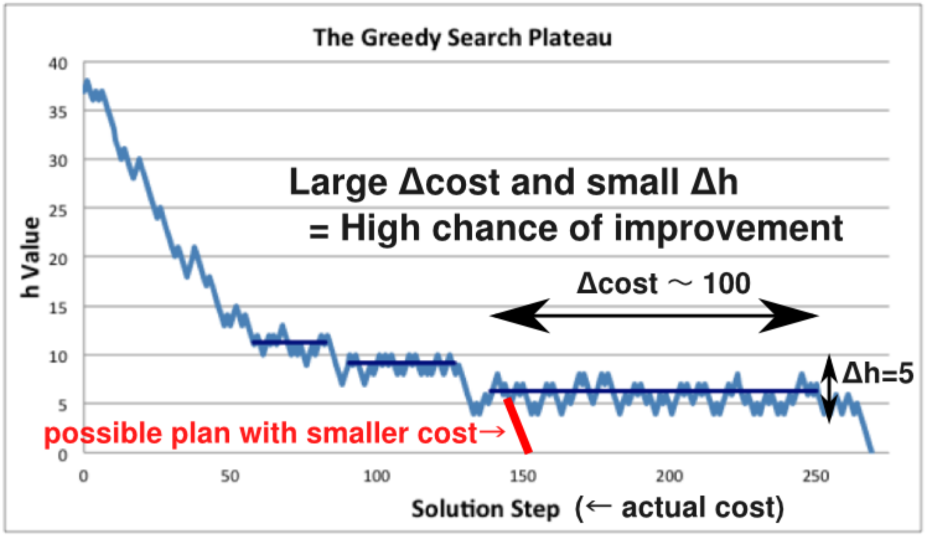
small-scale 15-puzzle/grid-pathfinding

$<$ 0.1 sec replanning time

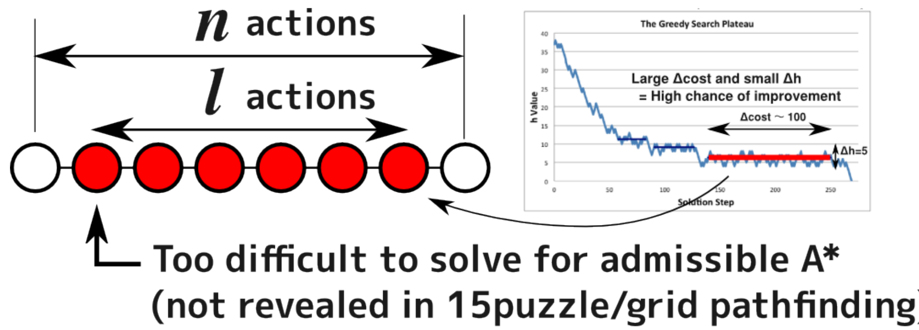
15 Window Priority Scheme in AIRS

Window priority: Δ **cost(i,j) /** Δ **h(i,j)**

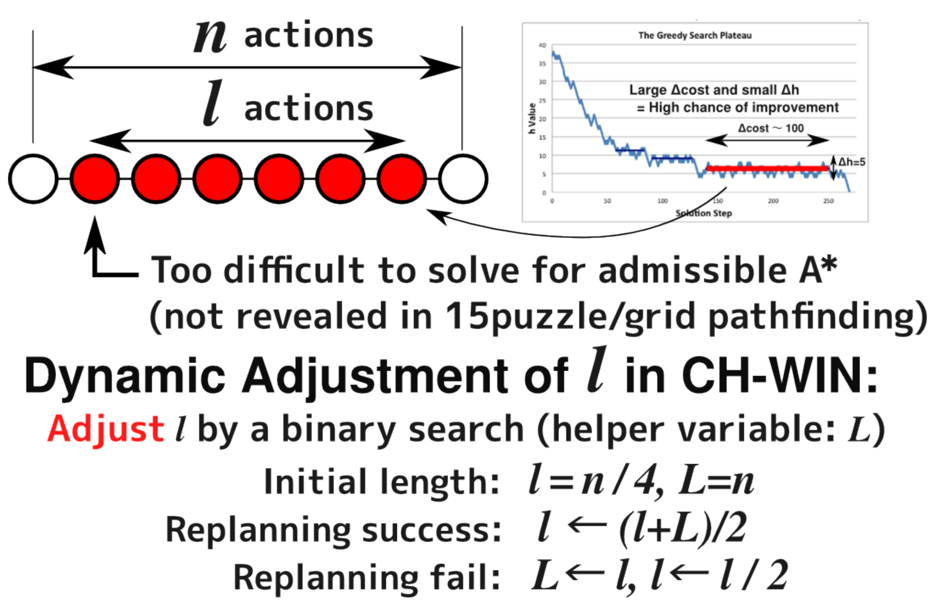
from (Estrem, Krebsbach 2012)



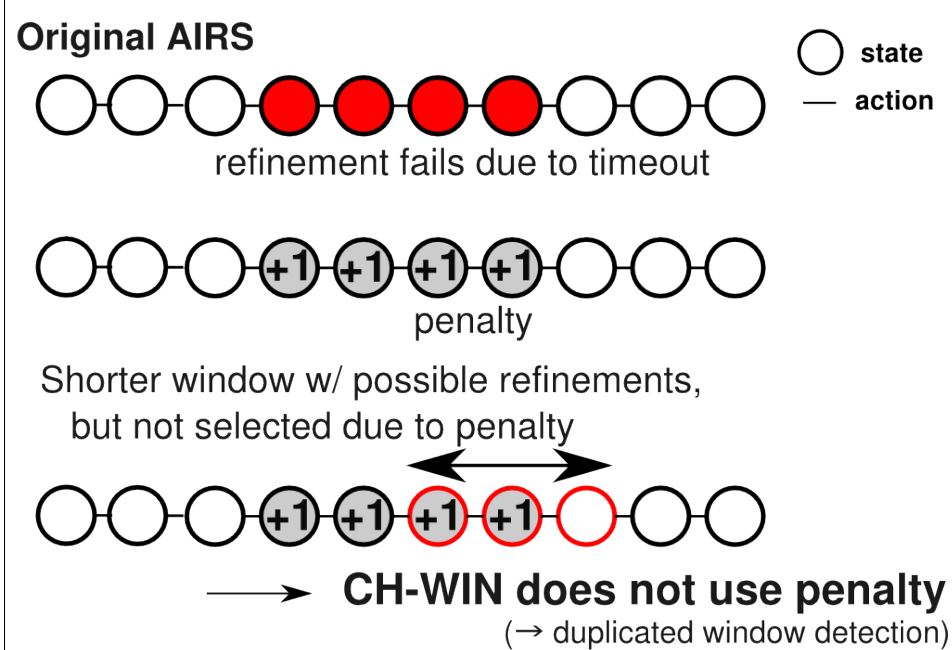
16 AIRS Pitfall 1 : windows too long



16.1 AIRS Pitfall 1 : windows too long



17 AIRS Pitfall 2 : Penalty could inhibit solving promising windoes



18 AIRS pitfall 3 : Keep it simple

```
while x < Length(Sol) − 2 do
    y ← x + 2;
    while y < Length(Sol) do
        Ratio ← h_2(Sol_x, Sol_y) / (g(Sol_y) − g(Sol_x) − p) +
                h_2(Sol_x, Sol_y) / ((g(Sol_y) − g(Sol_x) − p) − MaxOverlap(x + 1, y − 1, Sol, FS));
        if Ratio < Br then
            Br ← Ratio;
            (Bx,By) ← (x, y);
```

```
Function MaxOverlap(start, end, Sol, fails)
Greatest ← 0;
x ← 0;
while x < Length(fails) do
    q ← fails_x.second;
    s ← fails_x.first;
    overlap ← Min(g(Sol_end), g(Sol_q)) − Max(g(Sol_start), g(Sol_s));
    if overlap > Greatest then
        Greatest ← overlap;
    x ← x + 1;
return Greatest;
```

No explanation to why these complications are necessary

→ Remove it

19 Result

CH-WIN achieved the **best performance**

Algorithm	Harmonic Means of Ratios
LAMA(15min)	100%
LAMA(30min)	99.3%
AE	98.4%
AD	97.4%
AIRS	97.9%
PNGS	96.0%
R-WIN	95.9%
CH-WIN (proposed)	**93.3%**

20 Open Questions

~~What is the *baseline* of various search-based optimizers?~~

~~i.e. Are these ***complex*** algorithms necessary?~~ → Simple algorithms perform better

How to *combine* Polytime and Search-based Optimizers?

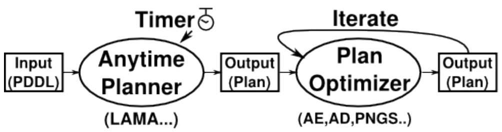
Also, how effective is it to combine them?

In this paper, we answer the following two open questions.

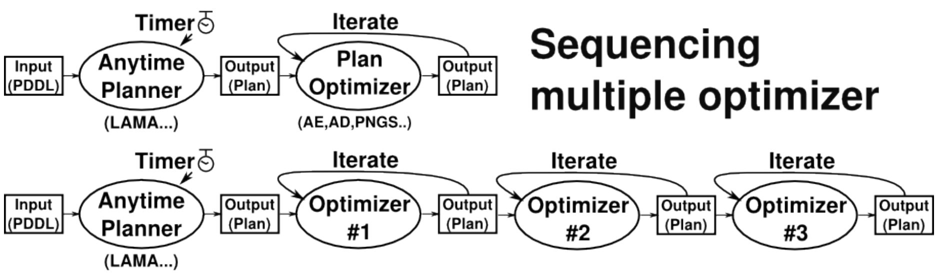
the first one is, . . .

the second one is. . .

21 Multiple Optimizers

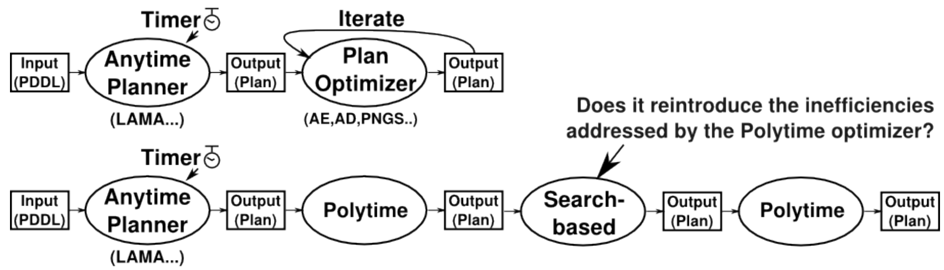


21.1 Multiple Optimizers



now lets start discussing the first one.

21.2 Polytime + Search-based Optimizers



21.3 Iterating Polytime Optimizer

domain	AD	AD AE	AD AE AD	AD AE AD AE	AE	AE AD	AE AD AE	AE AD AE AD
mean(harmonic)	98.4	97.4	97.4	97.4	97.4	97.4	97.4	97.4

Table 4: Result of applying poly-time optimizers iteratively to plans obtained by 15-minute runs of LAMA. The harmonic means of the ratios over all instances are shown.

- Iterating AE and AD is not effective
- we can safely assume that we apply them **at most once**

22 Polytime + Search-based Result

Algorithm x	x only	AE+AD+ x
LAMA(15min)	100%	-
LAMA(30min)	99.3%	-
AE	97.4%	
AD	98.4%	
AIRS	97.9%	95.6%
PNGS	96.0%	94.4%
R-WIN	95.9%	94.0%
CH-WIN (proposed)	93.3%	**91.8%**

23 Lessons Learned

- **Avoid** unnecessary complexity
- **R-WIN**, baseline, outperformed AIRS, PNGS etc.
- Use the simplest variant as a baseline, improve upon it
- CH-WIN, an improved AIRS variant, outperformed previous algorithms
- Reconfirmed poly+search effectiveness

Thank you for Listening!

Presentation available on

https://guicho271828.github.io/2016-6-13-hsdip/