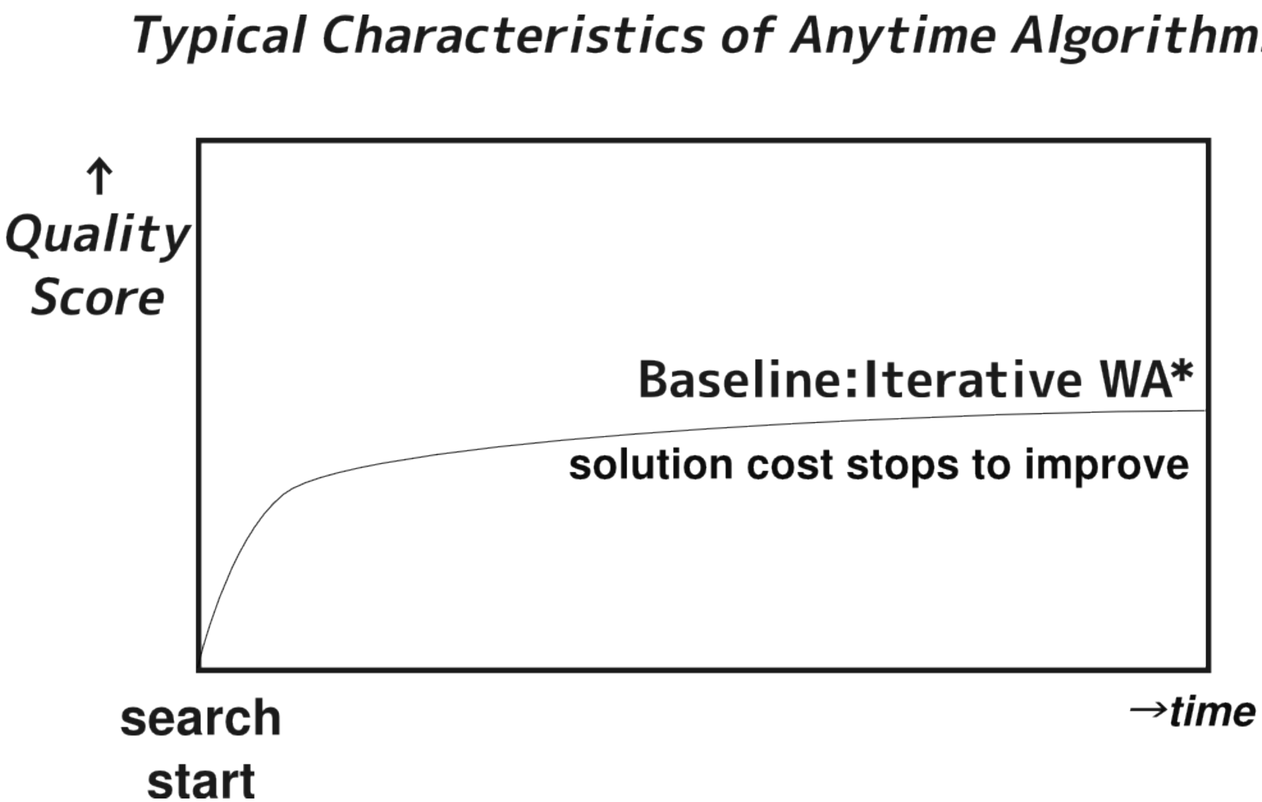
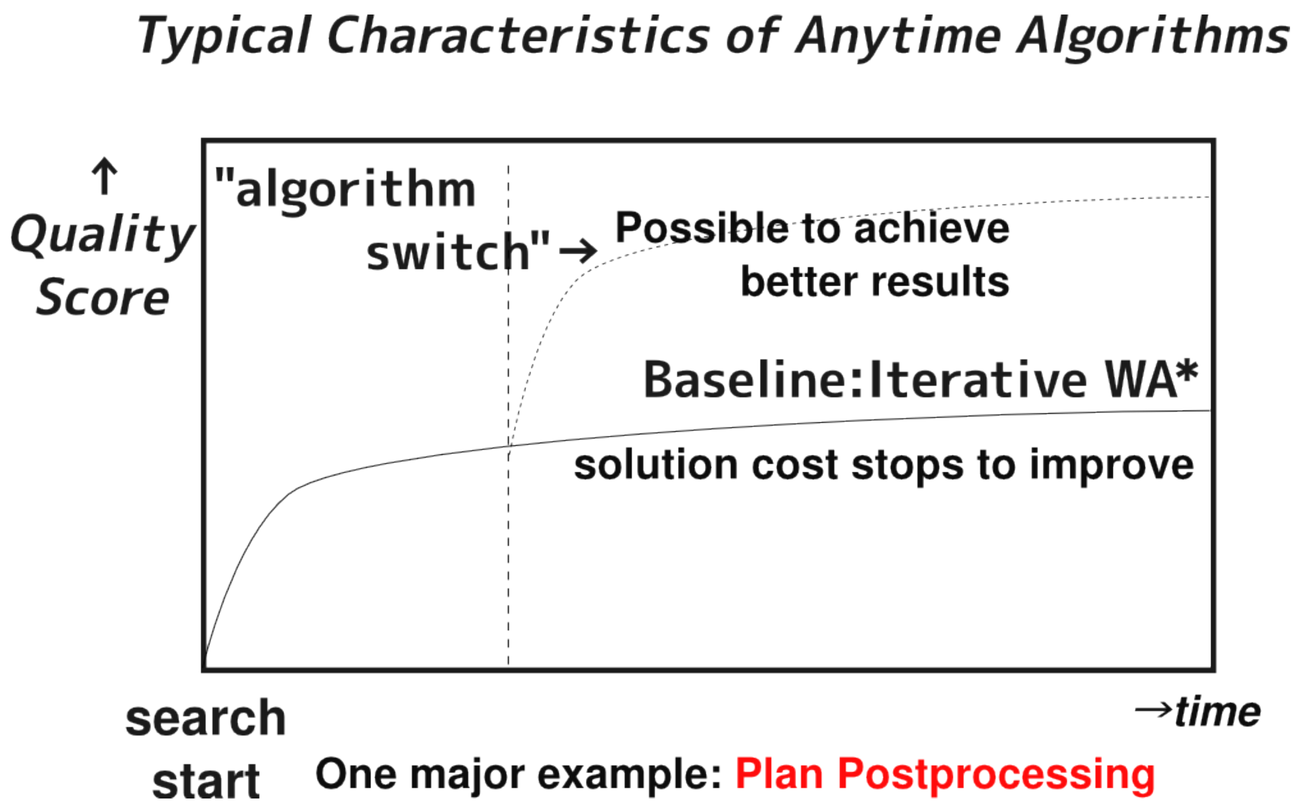
Shoma Endo, Masataro Asai, Alex Fukunaga

University of Tokyo

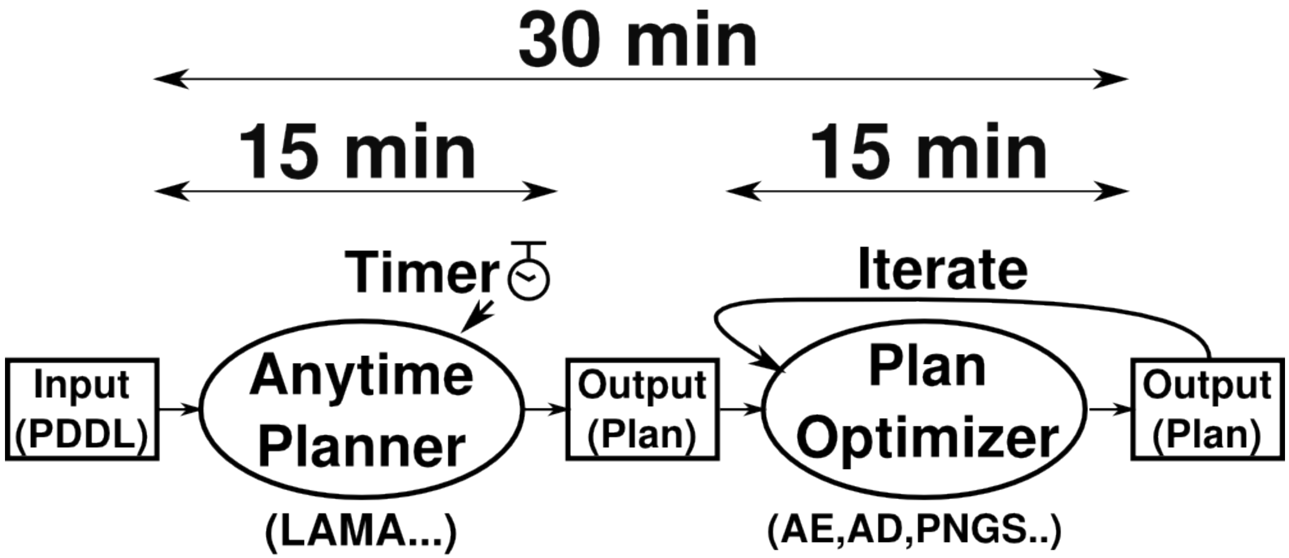
# 1 Satisficing Planning + Online Cost Refinements



# 2 Satisficing Planning + Algorithm Switch



# 3 Main Topic of the paper: Plan Postprocessing



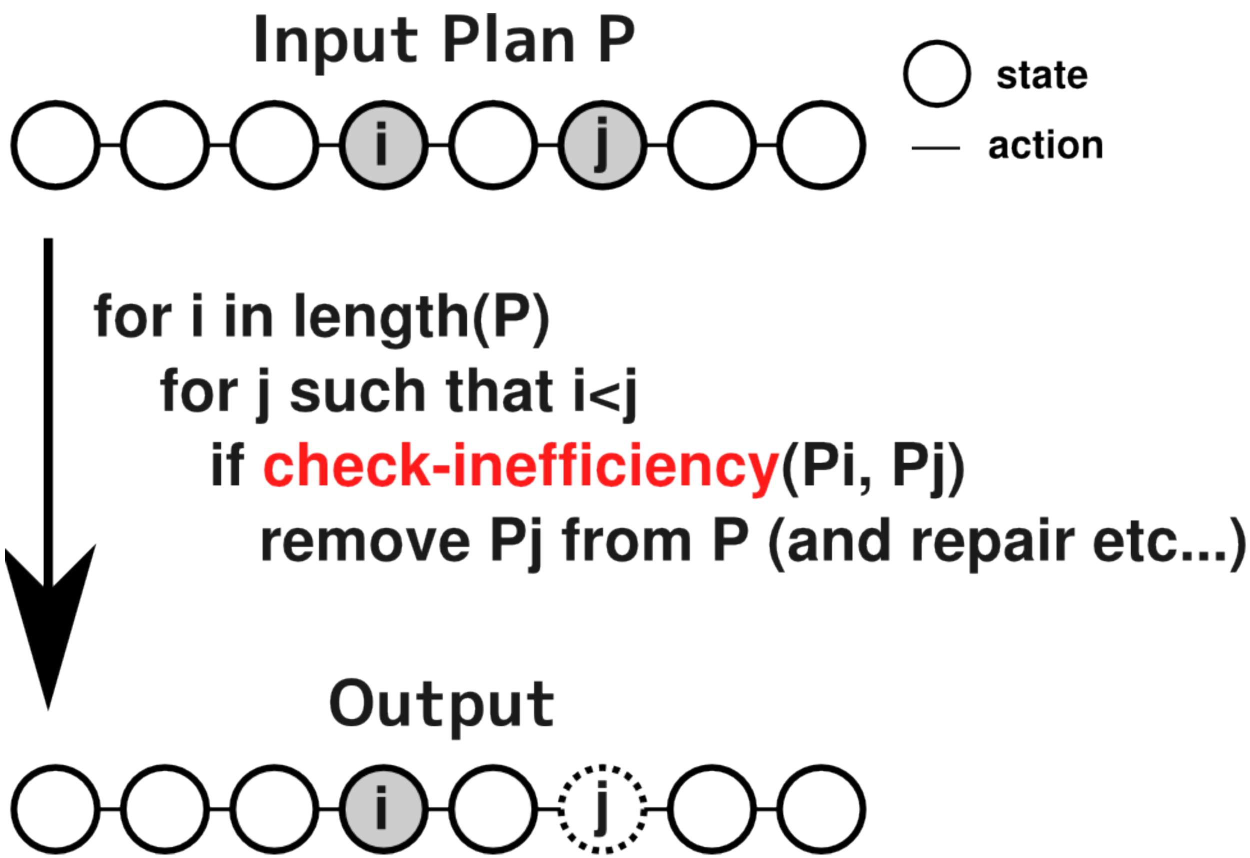
# 4 Various Postprocessing Systems

- 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)

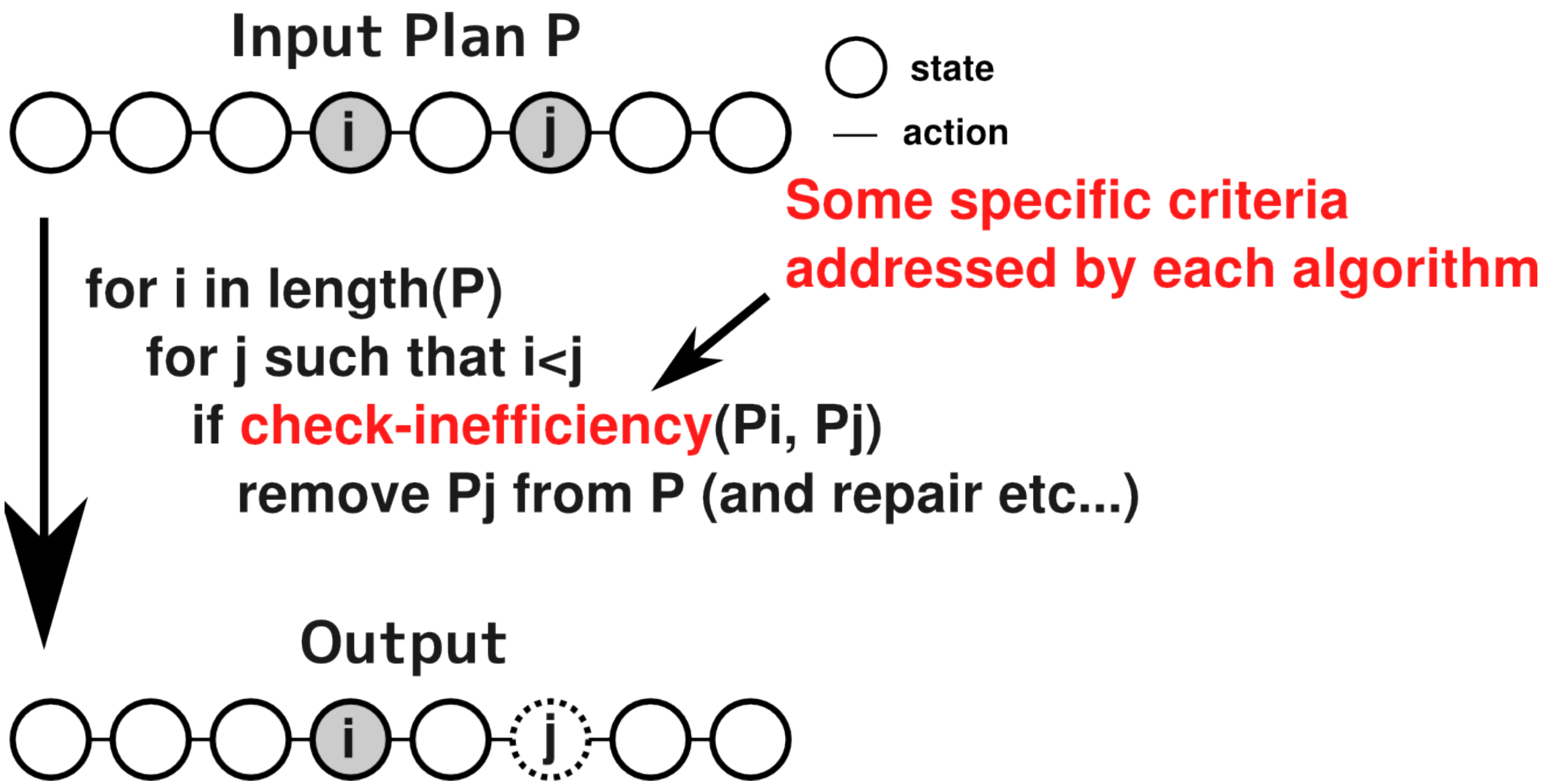
# 5 Two groups of Optimization Algorithms

| 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



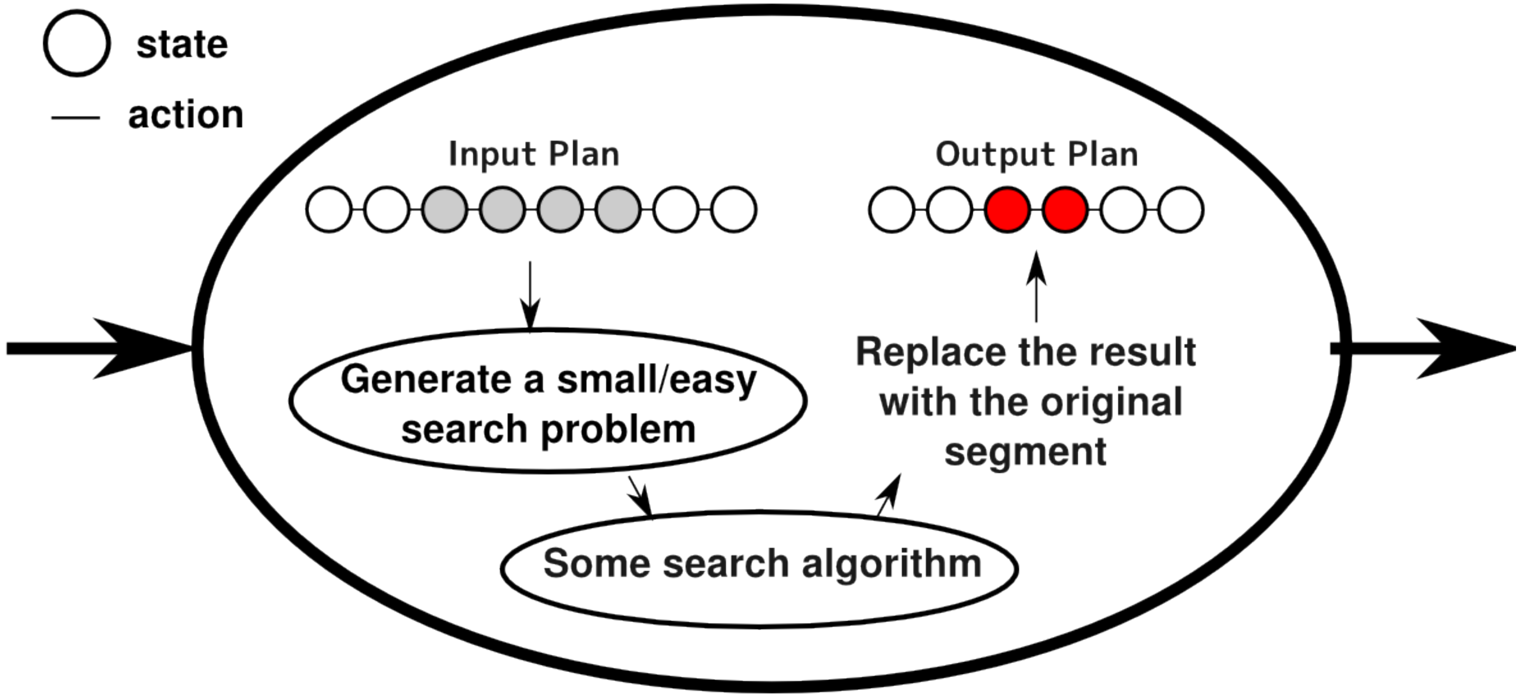
## 5.2 Polytime Optimizers



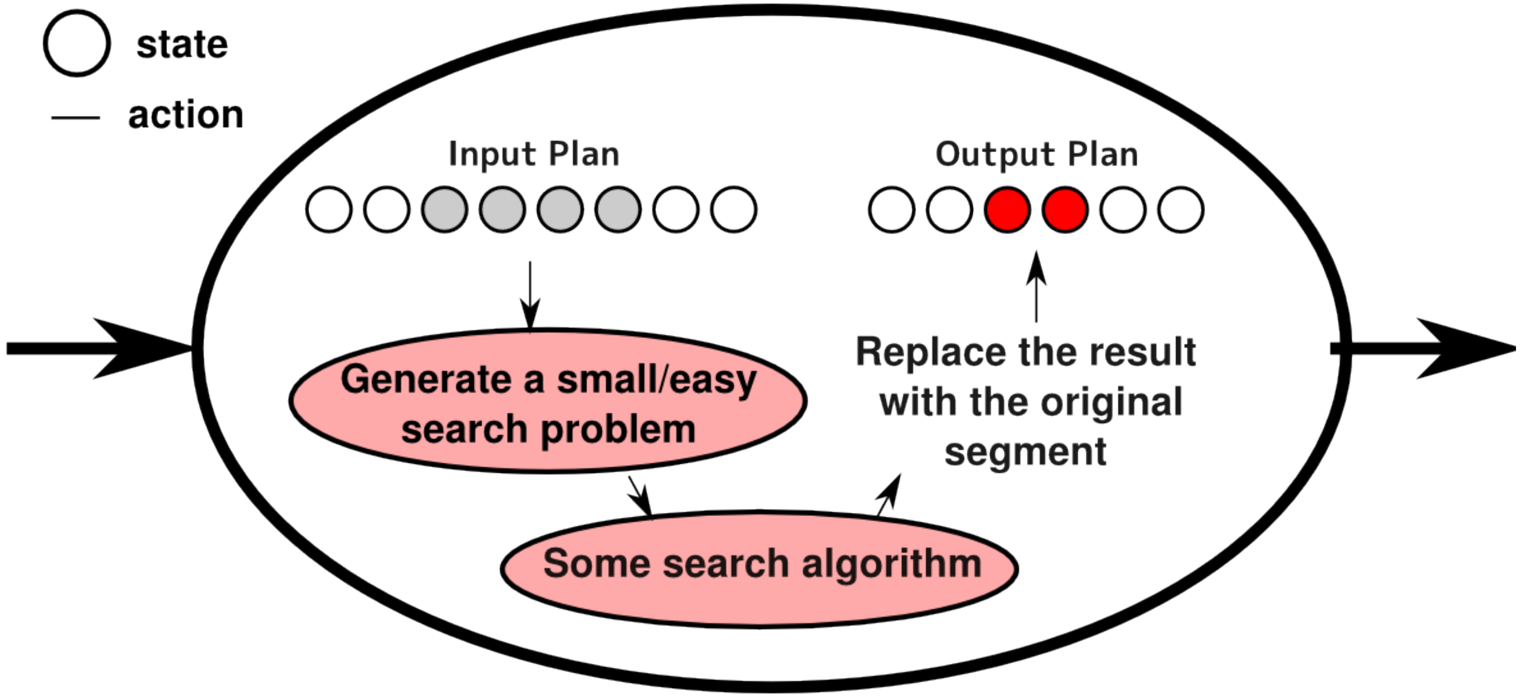
## 5.3 Two groups of Optimization Algorithms

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

## 5.4 Search-based Optimizers



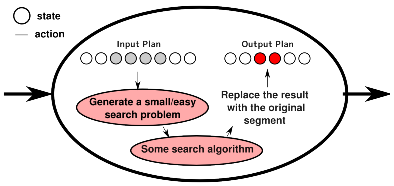
## 5.5 Search-based Optimizers



## 5.6 Two groups of Optimization Algorithms

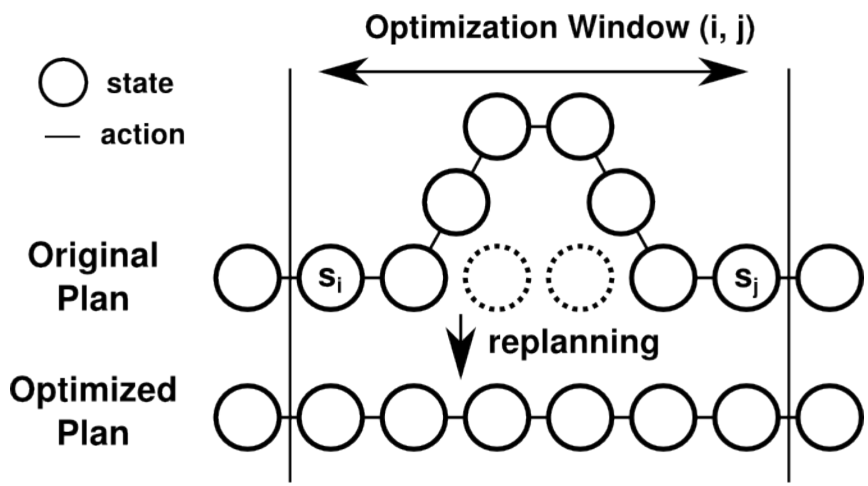
| Polytime Optimizer | Search-based Optimizer |
|---|---|
| Based on the direct analysis of a given plan | External solver refines a partial segment in a plan |
| AE : $O(n^2)$ | **PNGS** |
| AD : $O(n^2)$ | **AIRS** |
| n: plan length | BDPO2 |

## 5.7 Search-based Optimizers Taxonomy

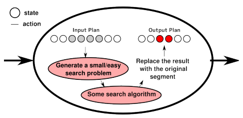



| | Subproblems Generation | Underlying Solver |
|---|---|---|
| PNGS | neighborhood graph (search space around the plan) | |
| AIRS | windows (ordered by $\Delta$ cost/$\Delta$ h) | |
| BDPO2 | windows (block decomposition) | |

## 5.8 Window-based optimization



## 5.9 Search-based Optimizers Taxonomy




| | Subproblems Generation | Underlying Solver |
|---|---|---|
| PNGS | neighborhood graph | Blind unit-cost A* |
| | (search space around the plan) | or Backward Breadths First |
| AIRS | windows (ordered by $\Delta$ cost/$\Delta$ h) | Blind Bidirectional |
| BDPO2 | windows (block decomposition) | Admissible A* |
| | | or PNGS (layered approach) |

# 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 Generation | Underlying Solver |
|---|---|---|
| PNGS | **neighborhood graph** | **Blind** unit-cost A* |
| | (search space around the plan) | or **Backward Breadths First** |
| AIRS | windows **(ordered by $\Delta$ cost/$\Delta$ h)** | **Blind Bidirectional** |
| BDPO2 | windows **(block deordering)** | Admissible A* |
| | | or **PNGS** (layered approach) |

→ **Hard to obtain a useful observation from the experiments**

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

| | Subproblems Generation | Underlying Solver |
|---|---|---|
| PNGS | neighborhood graph | Blind unit-cost A* |
| | (search space around the plan) | or Backward Breadths First |
| AIRS | windows (ordered by $\Delta$ cost/$\Delta$ 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 Generation | Underlying Solver |
|---|---|---|
| PNGS | neighborhood graph (search space around the plan) | **Admissible A$^*$ + LMcut** |
| AIRS | windows (ordered by $\Delta$ cost/$\Delta$ h) | **Admissible A$^*$ + LMcut** |
| BDPO2 | windows (block deordering) | **Admissible A$^*$ + LMcut** |
| R-WIN | windows (random) | **Admissible A$^*$ + LMcut** |

→ **Focus on the subproblem generation**

# 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)

(AD, AE as a point of reference)

| 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)

(AD, AE as a point of reference)

| 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**

# Sadly ”Improvements” did not outperform the simplest baseline

Luckily we have a reliable baseline to improve upon!

# 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

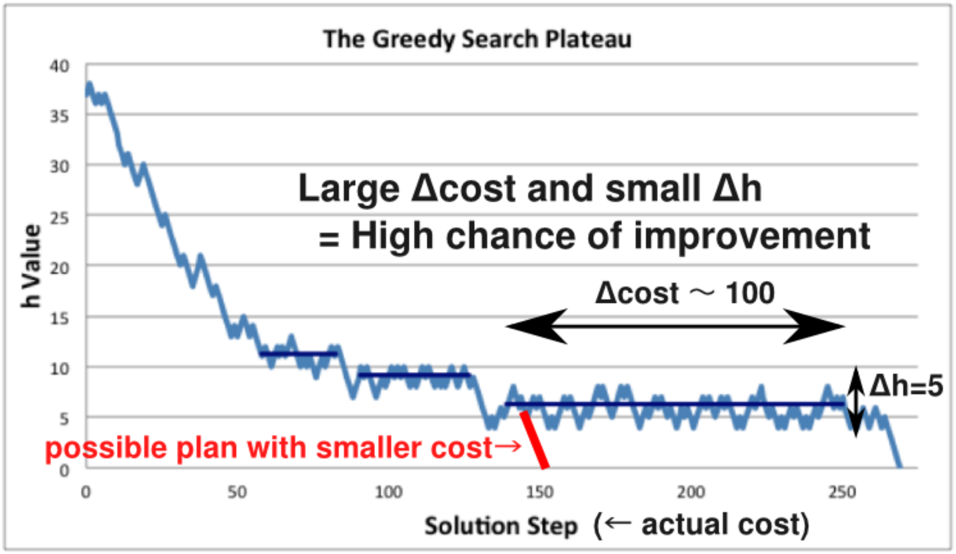
**small-scale 15-puzzle/grid-pathfinding**

**$<$ 0.1 sec replanning time**

# 15 Window Priority Scheme in AIRS

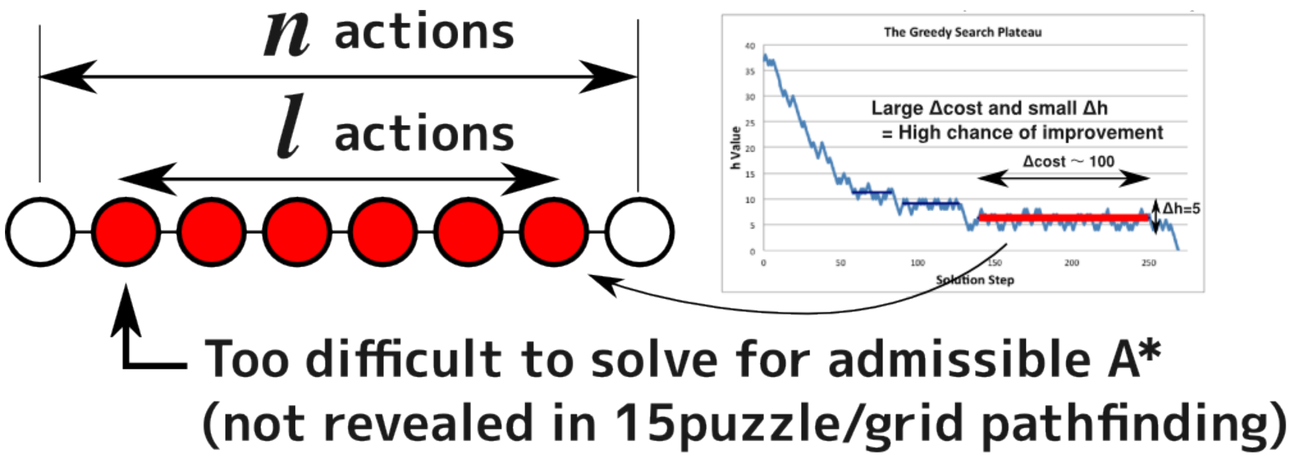
Priority: **Δ cost(i,j) / Δ h(i,j)**

from (Estrem, Krebsbach 2012)

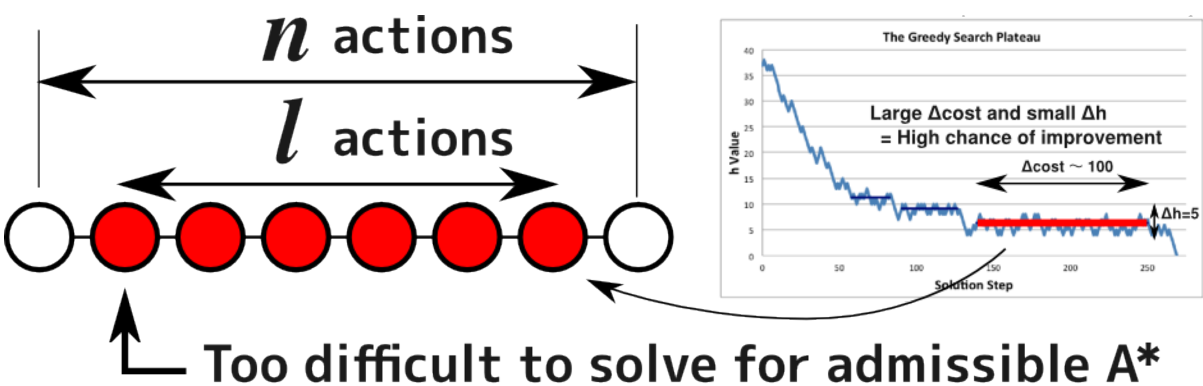



Priority seems promising, **but. . .**

# 16 AIRS Pitfall 1 : Windows may be too long/difficult



## 16.1 AIRS Pitfall 1 : Windows may be too long/difficult




Too difficult to solve for admissible A*
(not revealed in 15puzzle/grid pathfinding)

**Dynamic Adjustment of $l$ in CH-WIN:**

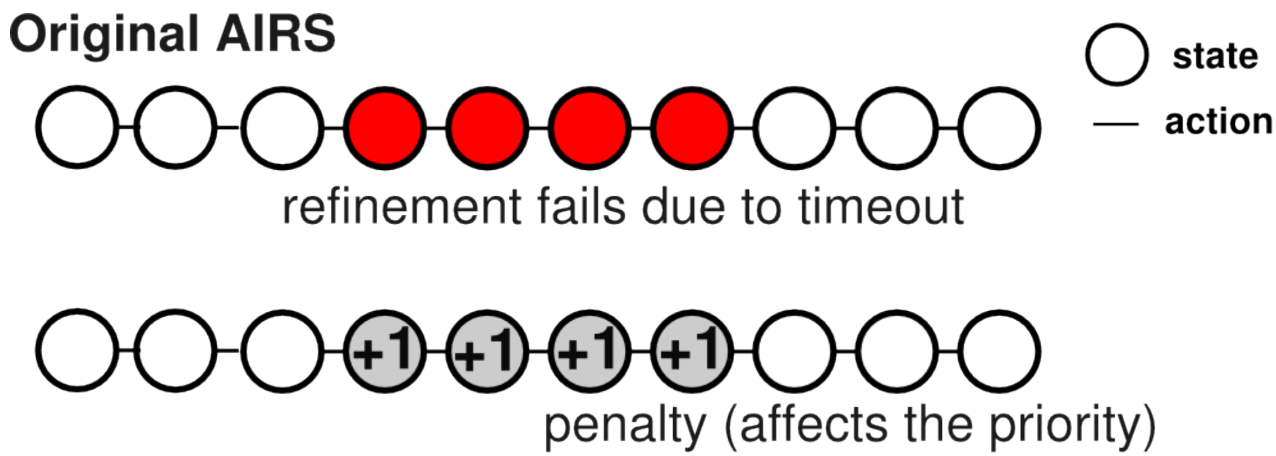
Adjust $l$ by a binary search (helper variable: $L$)

Initial length: $l = n/4$, $L = n$

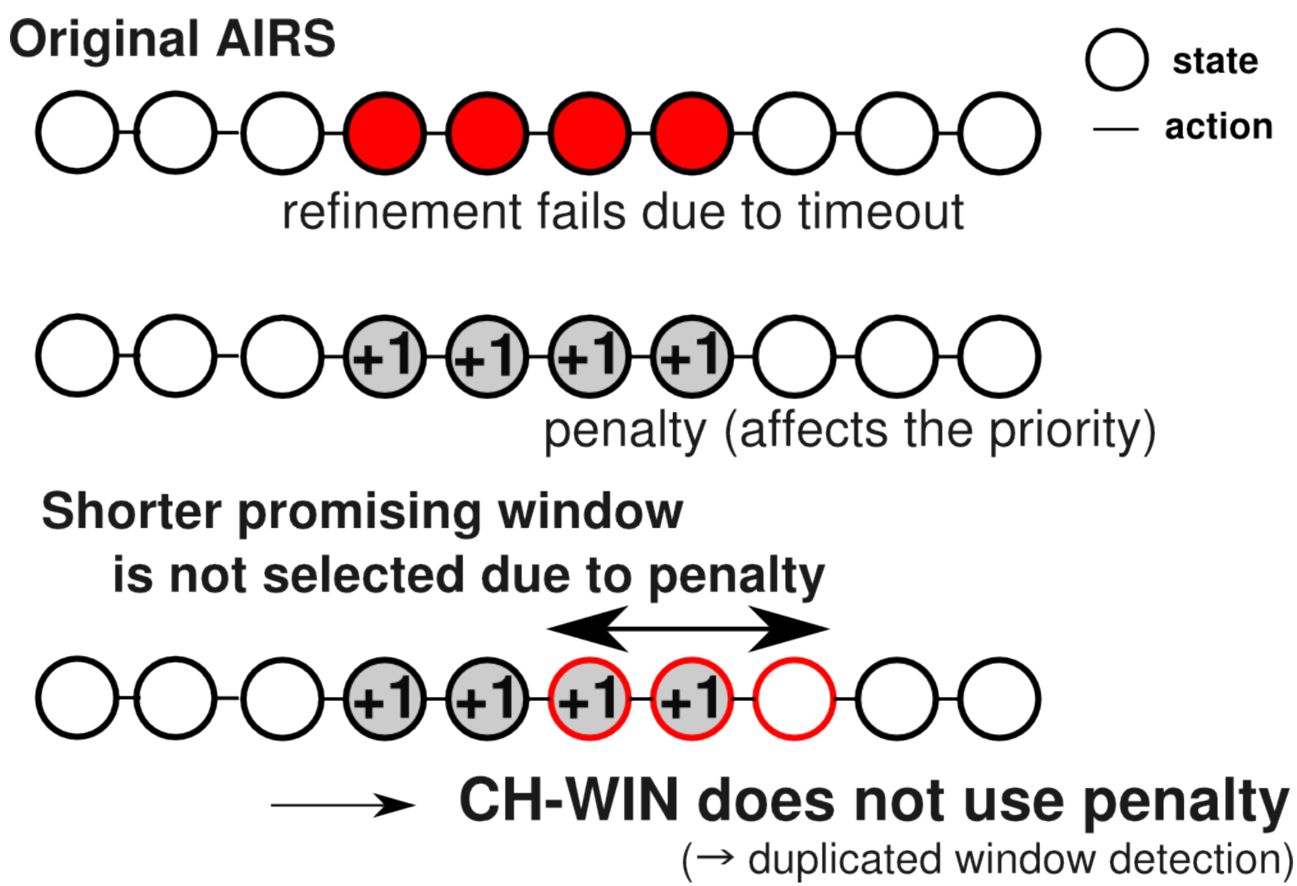
Replanning success: $l \leftarrow (l+L)/2$

Replanning fail: $L \leftarrow l$, $l \leftarrow l/2$

# 17 AIRS Pitfall 2 : Inappropriate Penalty Scheme



## 17.1 AIRS Pitfall 2 : Inappropriate Penalty Scheme



# 18 AIRS Pitfall 3 : Overly Complicated → Keep it simple

**while** $\mathsf{x} < \mathtt{Length}(\mathsf{Sol}) - 2$ **do**
  $\mathsf{y} \leftarrow \mathsf{x} + 2$;
  **while** $\mathsf{y} < \mathtt{Length}(\mathsf{Sol})$ **do**

$$\mathsf{Ratio} \leftarrow \frac{h_2(\mathsf{Sol_x}, \mathsf{Sol_y})}{\mathtt{g}(\mathsf{Sol_y}) - \mathtt{g}(\mathsf{Sol_x}) - \mathsf{p}} + \frac{h_2(\mathsf{Sol_x}, \mathsf{Sol_y})}{(\mathtt{g}(\mathsf{Sol_y}) - \mathtt{g}(\mathsf{Sol_x}) - \mathsf{p}) - \mathtt{MaxOverlap}(\mathsf{x}+1, \mathsf{y}-1, \mathsf{Sol}, \mathsf{FS})};$$

    **if** $\mathsf{Ratio} < \mathsf{Br}$ **then**
      $\mathsf{Br} \leftarrow \mathsf{Ratio}$;
      $(\mathsf{Bx}, \mathsf{By}) \leftarrow (\mathsf{x}, \mathsf{y})$;

**Penalty is not just +1 !**

**Function** MaxOverlap(start, end, Sol, fails)

Greatest $\leftarrow 0$;
$\mathsf{x} \leftarrow 0$;
**while** $\mathsf{x} < \mathtt{Length}(\mathsf{fails})$ **do**
  $\mathsf{q} \leftarrow \mathsf{fails_x}.second$;
  $\mathsf{s} \leftarrow \mathsf{fails_x}.first$;
  overlap $\leftarrow \mathtt{Min}(\mathtt{g}(\mathsf{Sol_{end}}), \mathtt{g}(\mathsf{Sol_q})) - \mathtt{Max}(\mathtt{g}(\mathsf{Sol_{start}}), \mathtt{g}(\mathsf{Sol_s}))$;
  **if** overlap > Greatest **then**
    Greatest $\leftarrow$ overlap;
  $\mathsf{x} \leftarrow \mathsf{x} + 1$;
**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

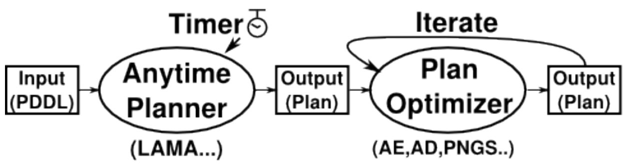
~~What is the *baseline* of various search-based optimizers?~~

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

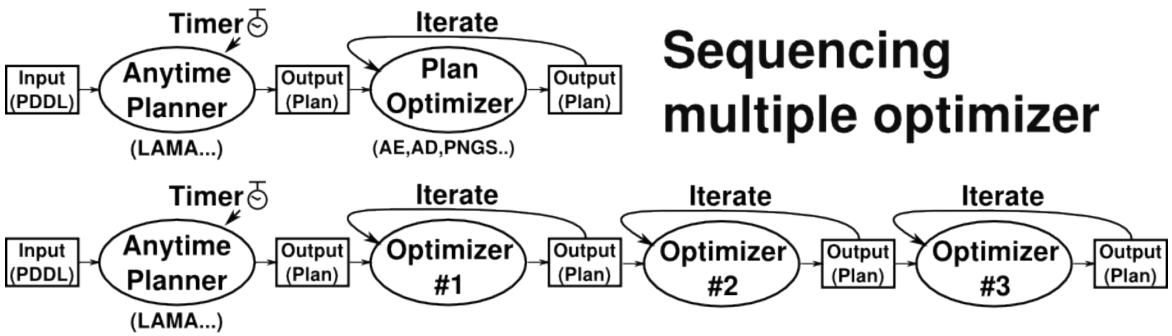
**How to *combine* Polytime and Search-based Optimizers?**

Also, how effective is it to combine them?

# 21 Multiple Optimizers



## 21.1 Multiple Optimizers



## 21.2 Iterating Polytime Optimizer

| domain | AD | AD<br>AE | AD<br>AE<br>AD | AD<br>AE<br>AD<br>AE | AE | AE<br>AD | AE<br>AD<br>AE | AE<br>AD<br>AE<br>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<br>$x$ | $x$ 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!