LPG: Local search for Planning Graphs

Seung H. Chung

16.412J
Cognitive Robotics

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Outline

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

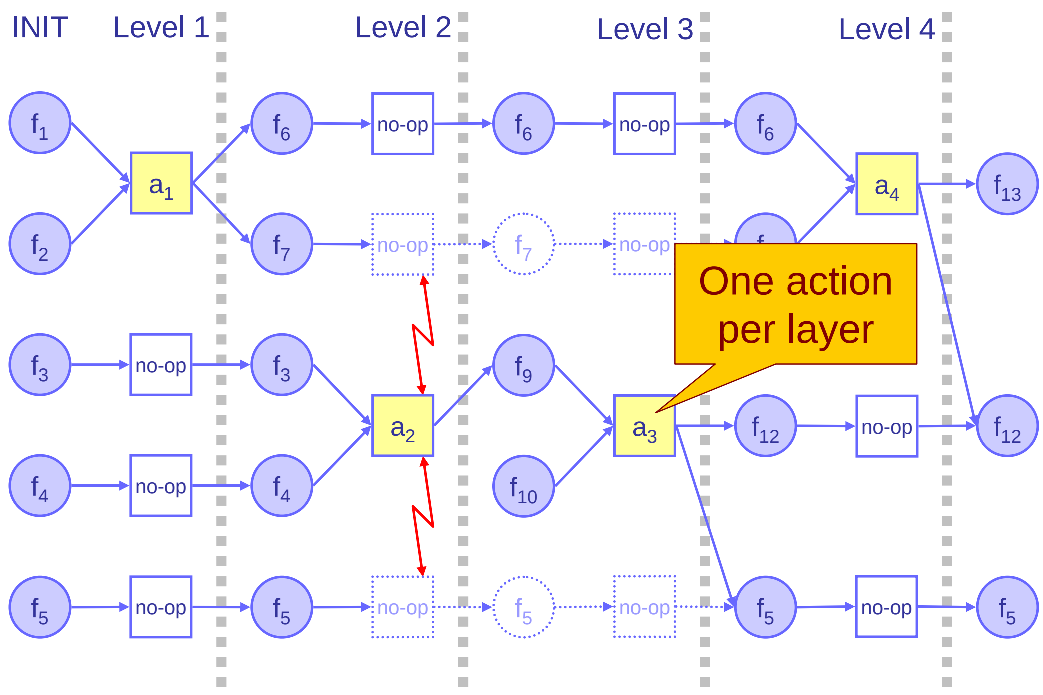
- Temporal Action Graph
- Walksat: Stochastic Local Search
- Better Neighbor
- Relaxed Plan

- A. Gerevini, A. Saetti, I. Serina "Planning through Stochastic Local Search and Temporal Action Graphs", to appear in *Journal of Artificial Intelligence Research* (JAIR).

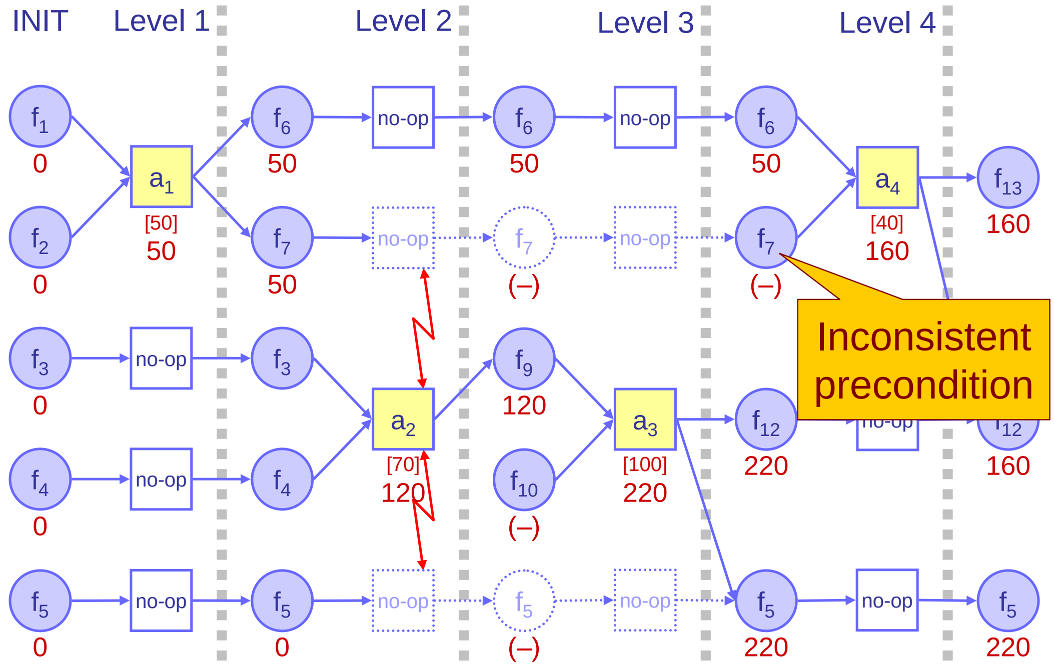
Overview

- Uses Strips-like operator but adds time and metric resources to the planning description: Planning Domain Definition Language (PDDL) 2.1
- Features:
  - Uses Temporal Action graphs (TA-graphs): Similar to a plan graph, but adds temporal representation
  - Uses stochastic local search: Similar to Walksat
  - Uses relaxed plan for heuristic to guide the search: similar to FF
- LPG outperformed all general purpose planners in the time and metric resource domains (3rd IPC)

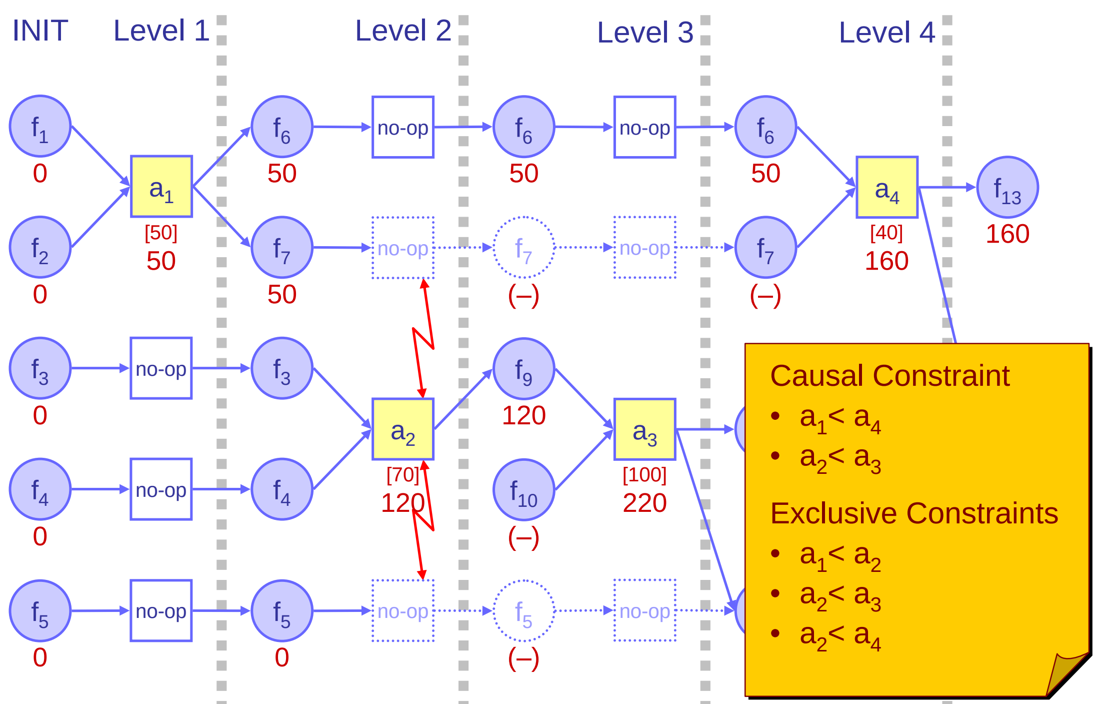
Linear Action Graph (LA-graph)



Temporal Action Graph (TA-Graph)



Ordering Constraint



Walkplan: Stochastic Local Search

- **Walkplan(**Π,max_steps,max_restarts,p**)**
  - Input
    - Π : Planning problem description
    - max_steps : Maximum number of search
    - max_restarts : Maximum number of restart
    - p : Noise factor
  - Output
    - Solution TA-graph

- Idea:
  - With probability p use stochastic local search to find a plan
  - Search the plan space max_steps number of times
  - If no plan is found, try restarting the search from the beginning up to max_restarts number of time

Walkplan Algorithm

```
Walkplan(Π,max_steps,max_restarts,p)
for i = 1 to max_restarts do
  A = an initial TA-graph derived from Π
  for j = 1 to max_steps do
    if A is solution then return A
    σ = an inconsistency in A
    N(σ,A) = neighborhood of A for σ
    if ∃A'∈ N(σ,A) such that A' is no worse than A then
       A = A'
    else if random < p then
       A = A'∈ N(σ,A)
    else
       A = best A'∈ N(σ,A)
return fail
```

Set of TA-graphs in which an action was inserted or removed to resolve the inconsistency

What is a better neighbor?

Better Neighbor?

- A neighbor $A' \in N(\sigma, A)$ resolves the inconsistency σ by inserting or deleting an action.
- Use evaluation function E(a)

$$E(a) = \begin{cases} E(a)^i = \alpha \cdot Execution\_cost\,(a)^i + \beta \cdot Temporal\_cost(a)^i + \gamma \cdot Search\_cost(a)^i \\ E(a)^r = \alpha \cdot Execution\_cost\,(a)^r + \beta \cdot Temporal\_cost(a)^r + \gamma \cdot Search\_cost(a)^r \end{cases}$$

Relaxed Plan

- Idea: Don't consider the mutex relation and perform reachability analysis.
- Insert action a
  - Find all actions that is required to support the preconditions of a
  - Compute the maximum time duration required for all actions
  - Return:
    - Set of actions added: Aset(EvalAdd(a))
    - Max time duration of the actions: End_time(EvalAdd(a))
- Remove action a
  - Find all actions that is required to support all preconditions that were unsupported due to removal of a
  - Compute the maximum time duration required for all newly inserted actions
  - Return:
    - Set of actions added: Aset(EvalDel(a))
    - Max time duration of the actions: End_time(EvalDel(a))

Better Neighbor

- Insert an action

$$\text{Execution\_cost }(a)^i = \Sigma_{a' \in Aset(EvalAdd(a))} Cost(a')$$
$$\text{Temporal\_cost}(a)^i = \text{End\_time}(EvalAdd(a))$$
$$\text{Search\_cost}(a)^i = |Aset(EvalAdd(a))| + \Sigma_{a' \in Aset(EvalAdd(a))} Threats(a')$$

- Remove an action

$$\text{Execution\_cost }(a)^r = \Sigma_{a' \in Aset(EvalDel(a))} Cost(a')$$
$$\text{Temporal\_cost}(a)^r = \text{End\_time}(EvalAdd(a))$$
$$\text{Search\_cost}(a)^r = |Aset(EvalAdd(a))| + \Sigma_{a' \in Aset(EvalDel(a))} Threats(a')$$

Advantages and Disadvantages

- Pros
  - One of the fastest domain-independent planners
  - Relatively expressive domain description languages
  - Can easily be extended to be anytime algorithm
- Cons
  - Algorithm is not guaranteed to be complete
  - No guarantee on the quality of the plan
  - Does not allow flexible time bounds