Il Pianificatore LPG

Local Search for Planning Graphs

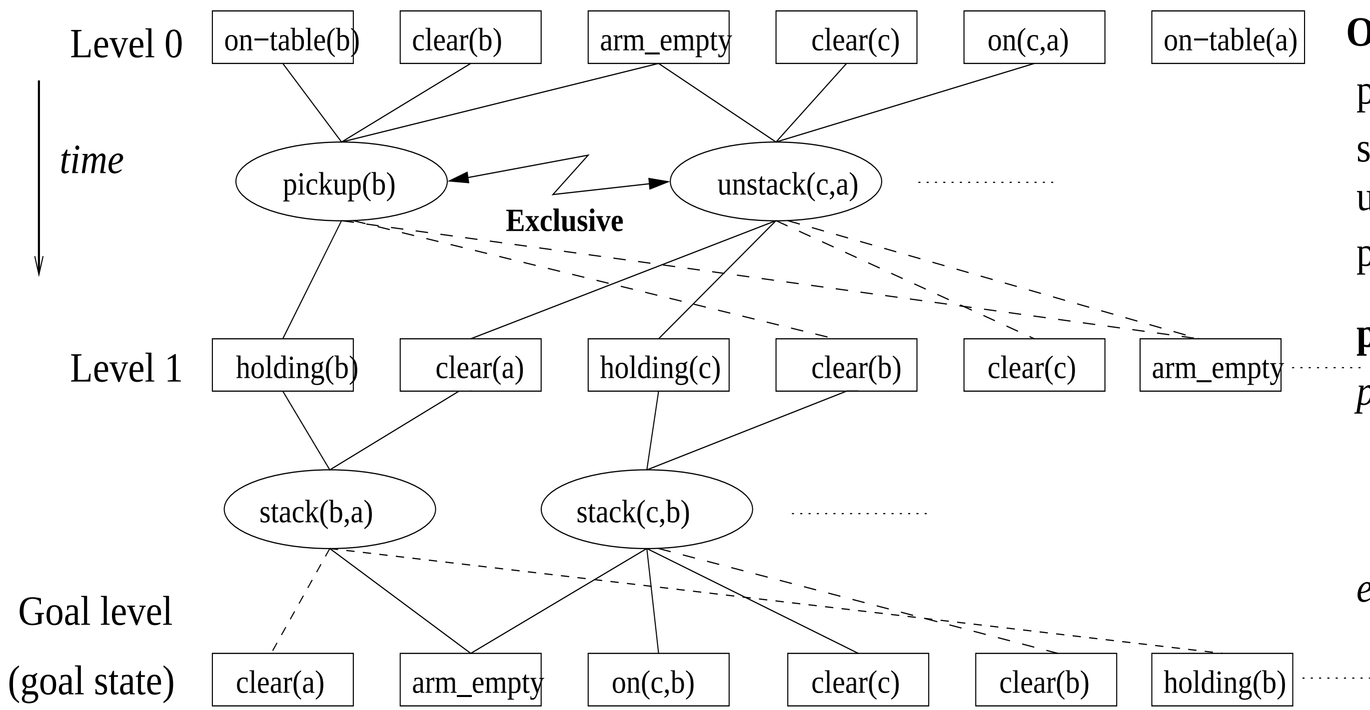
`http://zeus.ing.unibs.it/lpg`

Graphplan [Blum & Furst '95]

- **Planning Graph** (PG): *Directed acyclic "leveled" graph* automatically constructed from the problem specification.
- **Nodes** represent *facts* (goals, preconds, effects) or *actions* (and *no-ops* = dummy actions propagating facts of previous levels)
- **Edges** connect action-nodes to precondition/effect nodes
- **Levels** correspond to *time steps* (points); each level and has a layer of fact-nodes and a layer of action-nodes.
- **Mutual exclusion relations** between action-nodes and fact nodes E.g., A *mutex* B because one deletes a precondition or effect of the other

Planning = *finding a subgraph of the PG representing a valid plan*

Example of Planning Graph



Operators

pickup(?o)

stack(?o, ?under_o)

unstack(?o, ?under_o)

putdown(?o)

pickup(?o)

preconditions:

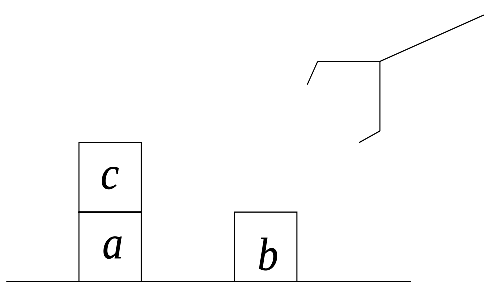
clear(?o)
ontable(?o)
arm_empty

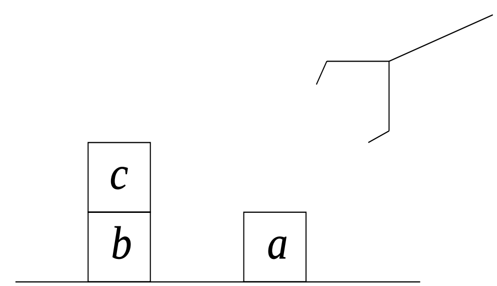
effects:

holding(?o)
not(arm_empty)
not(ontable(?o))
not(clear(?o))

Initial: on−table(a) on−table(b) on(c,a)
clear(c) clear(b) arm−empty

Goal: clear(a) arm−empty
on(c,b) clear(c)

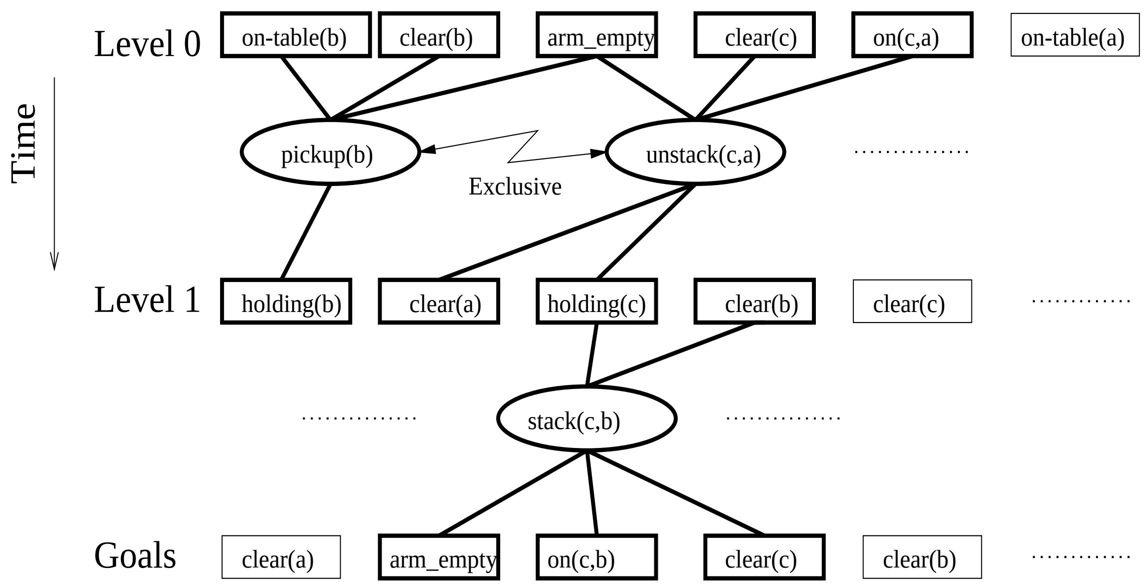




Action Graphs

An **action graph** ($\mathcal{A}$-graph) of a planning graph $\mathcal{G}$ is a subgraph of $\mathcal{G}$ such that, if an action-node a is in $\mathcal{A}$, then

- all the precondition-nodes/edges of a are in $\mathcal{A}$
- all the effect-nodes and add-edges of a are in $\mathcal{A}$



Inconsistency in an action graph $\mathcal{A}$:

- a pair of action-nodes in $\mathcal{A}$ that are "mutex"
- an action-node in $\mathcal{A}$ with an unsupported precondition-node

Linear Action Graph (LA-graph)

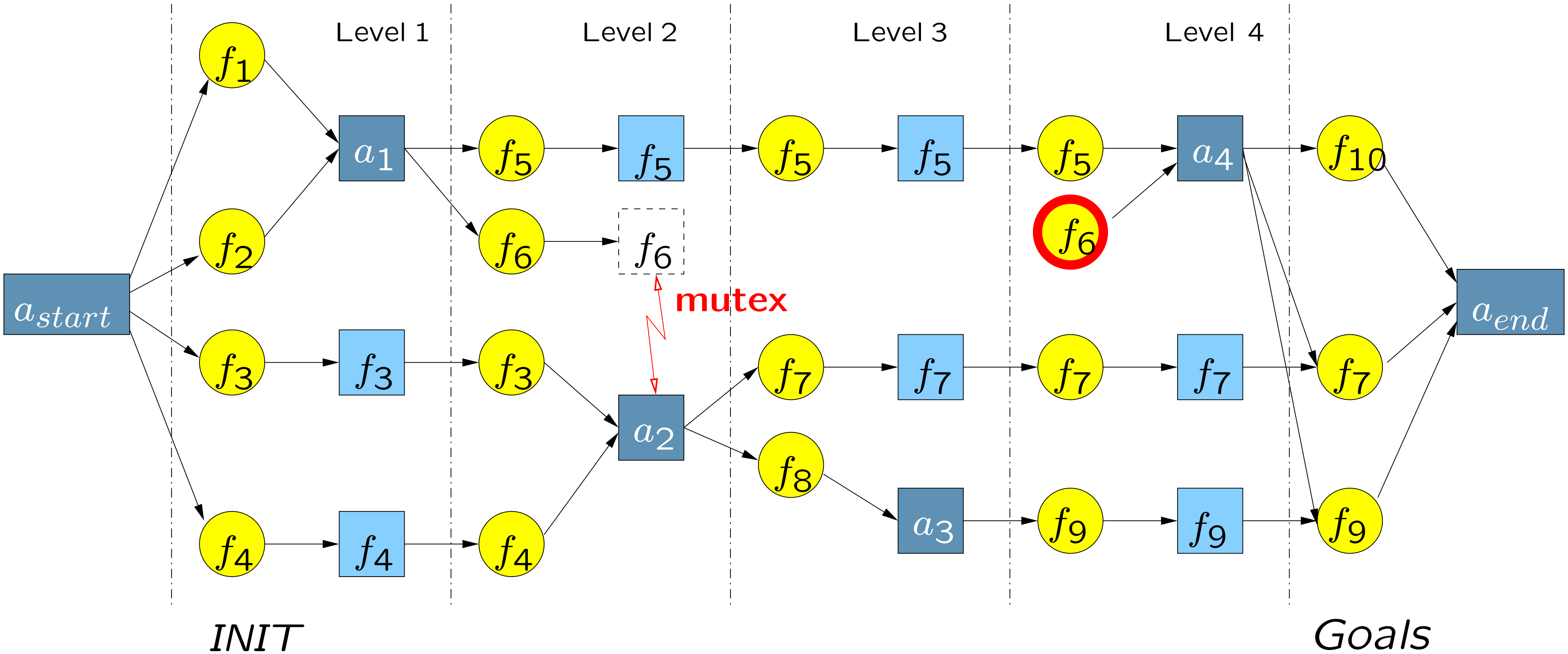
- **Linearity**: in each action layer *one* node representing an action plus no-ops (does *not* imply linear output plans)

- **Ordering constraints** Ω
  - from the causal structure:
    if a is used for a precondition of b, then $a^+ \prec b^- \in \Omega$
  - to order mutex actions:
    if a and b are mutex, then $a^+ \prec b^- \in \Omega$ or $b^+ \prec a^- \in \Omega$

- **Represented plan**: actions in the graph ordered by Ω
  Correct plan if there is no flaw in the LA-graph (*solution graph*)

- **Plan flaw**: **un***supported* precondition-node of an action node

Example of Linear Action Graph



Plan actions: $\{a_1, a_2, a_2, a_3\}$

Plan flaw: **unsupported precondition** f_6 **of** a_4 (not executable)

Local Search in the Space of $\mathcal{A}$-Graphs

Search space: set of all the $\mathcal{A}$-graphs of the planning graph ($\mathcal{G}$)

Initial state: any $\mathcal{A}$-graph of $\mathcal{G}$, e.g.,

- *random $\mathcal{A}$-graph*
- *$\mathcal{A}$-graph with supported precondition/goal nodes*
- *$\mathcal{A}$-graph from a valid plan for a similar problem* (**plan adaptation**)

Search steps: graph modifications to resolve an inconsistency:

- *graph extensions* (inserting one or more actions into $\mathcal{A}$);
- *graph reductions* (removing one or more actions from $\mathcal{A}$);
- *graph replacements* (replacing an action with another action).

Goal states: $\mathcal{A}$-graphs with no inconsistency (**solution graphs**)

Graph extension: automatic when a search limit is exceeded

General Local Search Procedure

1. *While $\mathcal{A}$ is not a solution graph do*
2. *Choose an inconsistency (flaw) s in $\mathcal{A}$;*
3. *Identify the* **neighborhood** $N(s, \mathcal{A})$ *and weight its elements using a parametrized* **action evaluation function** E*;*
4. **Select** *an $\mathcal{A}$-graph from $N(s, \mathcal{A})$ and apply the corresponding graph modification to $\mathcal{A}$.*

$N(s, \mathcal{A})$: *set of all the action graphs derivable from $\mathcal{A}$ by applying a graph modification resolving s.*

Prefer flaws at the earliest *graph level*

Stochastic Search: Walkplan

Similar to the heuristic in Walksat [Selman *et al.*]

The $\mathcal{A}$-graph selected from $N(s, \mathcal{A})$ is:

- *with probability* p *a graph in* $N(s, \mathcal{A})$ **randomly** *chosen*;
- *with probability* $1 - p$ *the* **best** *graph in* $N(s, \mathcal{A})$ according to E.

Action evaluation function

$$E : \begin{cases} E(a, A)^{insertion} = \alpha^i \cdot pre(a, \mathcal{A}) + \beta^i \cdot |Threats(a, \mathcal{A})| \\ E(a, \mathcal{A})^{removal} = \gamma^r \cdot unsup(a, \mathcal{A}) \end{cases}$$

$pre(a, \mathcal{A})$: number of unsupported preconditions/goals of a
$unsup(a, \mathcal{A})$: num. of supported preconditions *becoming unsupported* by adding a
$Threats(a, \mathcal{A})$ = supported preconditions becoming unsupported by adding a to $\mathcal{A}$.

Effect Propagation

- An action effect f can be **propagated** to preconditions of the next actions, unless there is another action interfering with f.

- If an action a interferes with f, the propagation is **blocked** at the time step t of a. When a is removed, f is propagated from t.

- Propagation performed using the "no-ops" of the planning graph.

$\Rightarrow$ **Stronger search steps** *One graph modification (search step) can remove more than one inconsistency* *at different levels*.

$\Rightarrow$ **Extended neighborhood**: *a precondition can be supported by inserting an action* *at any previous level (time step)*.

Heuristic Evaluation based on Relaxed Plans: E_π

$$E_\pi(a,\mathcal{A})^i = |\pi(a,\mathcal{A})^i| + \sum_{a' \in \pi(a,\mathcal{A})^i} |Threats(a',\mathcal{A})|$$

$$E_\pi(a,\mathcal{A})^r = |\pi(a,\mathcal{A})^r| + \sum_{a' \in \pi(a,\mathcal{A})^r} |Threats(a',\mathcal{A})|$$

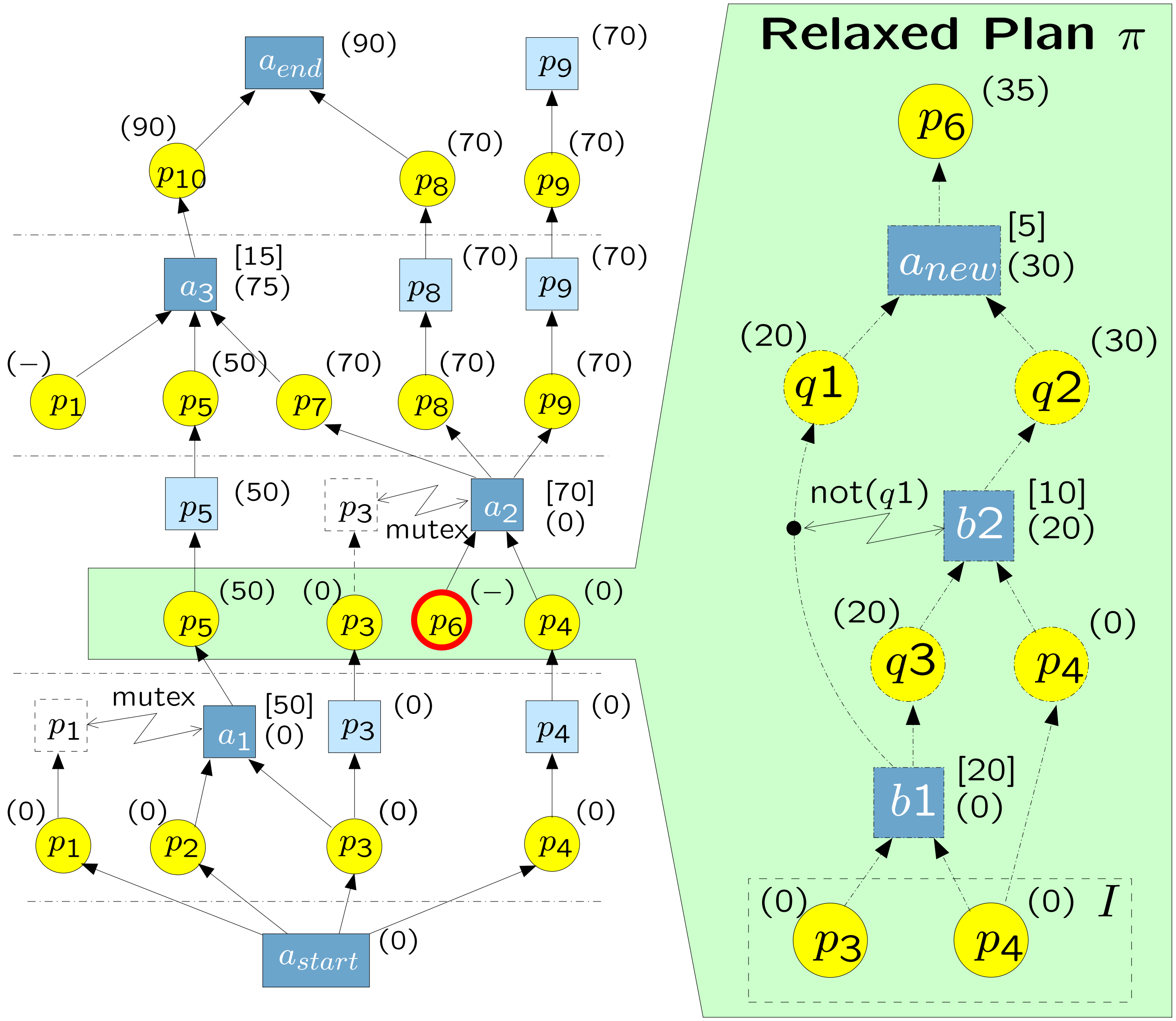
where

- $\pi(a,\mathcal{A})^i$ is an estimate of a minimal set of actions forming a relaxed plan achieving *Pre*(a) and *Threats*$(a,\mathcal{A})$;

- $\pi(a,\mathcal{A})^r$ is an estimate of a minimal set of actions forming a relaxed plan achieving *Unsup*$(a,\mathcal{A})$.

Relaxation: negative effects are ignored

Example of the Relaxed Plan



RelaxedPlan(G, $I(l)$, A)

Input: A set of goal facts (G), the set of facts that are true after executing the actions of the current LA-graph up to level l ($I(l)$), a possibly empty set of actions (A);

Output: An estimated minimal set of actions required to achieve G.

1. $G \leftarrow G - I(l)$; $Acts \leftarrow A$;
2. $F \leftarrow \bigcup_{a \in Acts} Add(a)$;
3. **while** $G - F \neq \emptyset$
4. $\quad$ $g \leftarrow$ a fact in $G - F$;
5. $\quad$ $bestact \leftarrow Bestaction(g)$;
6. $\quad$ $Rplan \leftarrow$ RelaxedPlan$(Pre(bestact), I(l), Acts)$;
7. $\quad$ $Acts \leftarrow Rplan \cup \{bestact\}$;
8. $\quad$ $F \leftarrow \bigcup_{a \in Acts} Add(a)$;
9. **return** $Acts$.

$$Bestaction(g) = \underset{\{a' \in A_g\}}{ARGMIN} \left\{ \underset{p \in Pre(a') - F}{MAX} Num\_acts(p,l) + |Threats(a')| \right\},$$

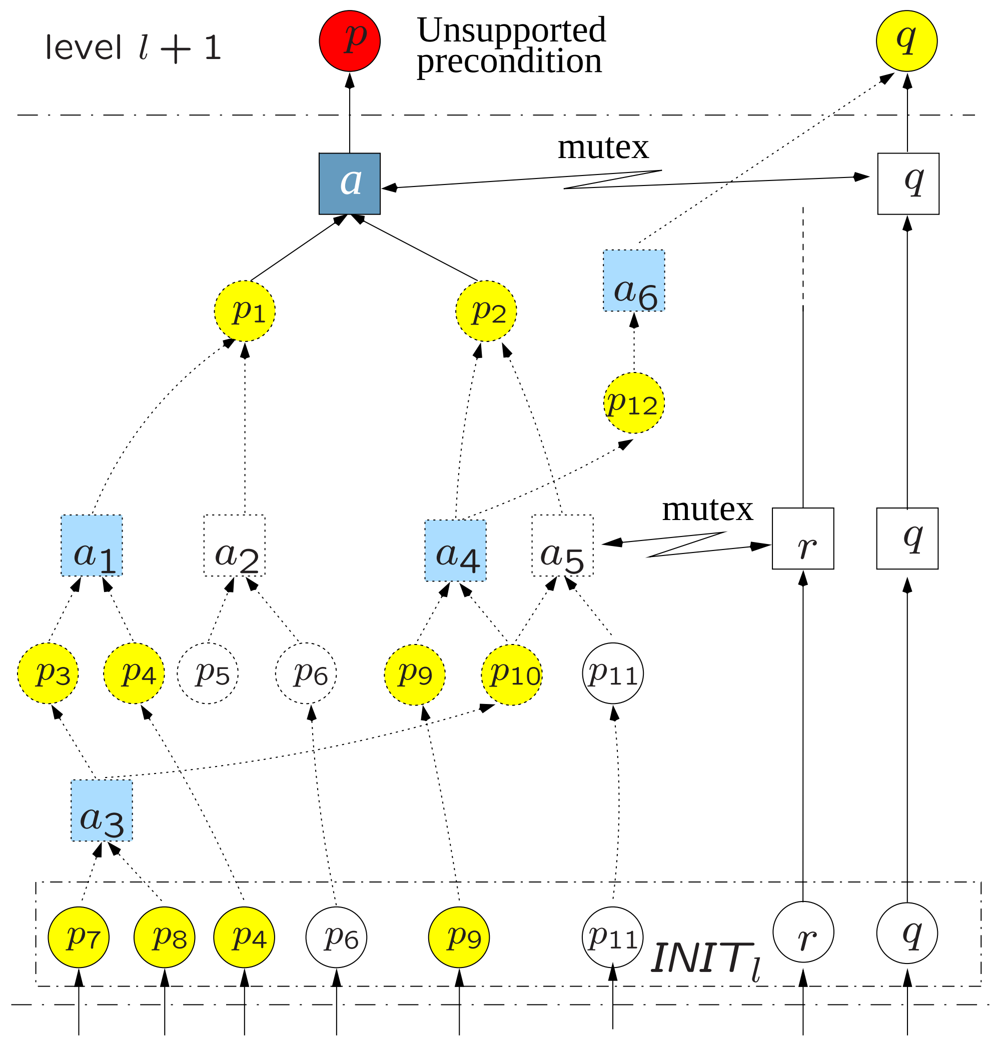
where F is the set of positive effects of the actions currently in *Acts*, and A_g is the set of actions with the effect g and with all preconditions reachable from the initial state.

Relaxed Plan Construction (example)

Fact	Num_acts
p_1	2
p_2	2
p_3	1
p_5	6
p_{10}	1
p_{12}	2

Fact	$Time$
p_4	170
p_6	300
p_7	50
p_8	30
p_9	170
p_{11}	30

Action	$Duration$
a	30
a_1	70
a_3	100
a_4	30
a_6	90



Relaxed plan $=$ $\boxed{\{a_1, a_3, a_4\} \cup \{a_6\}}$

$End\_time(\{a_1, a_3, a_4\}) = 240$

Simulation of plan generation using InLPG

Performance di LPG

- Attualmente uno dei pianificatori piú espressivivi

- Attualmente uno dei migliori pianificatori in termi di qualitá dei piani

- Ma anche uno dei piú veloci:
  - Nel 2002 ha vinto la international planning competition (IPC)
  - Nel 2004 ha vinto il secondo posto nella IPC

Experimental Results: Computing a Plan

Planning problem	LPG Wplan	Blackbox Wsat	Blackbox Chaff	GP-CSP	IPP	STAN
`rocket-a`	0.05	1.25	5.99	1.55	20.2	6.49
`rocket-b`	0.06	1.51	6.16	3.02	38.83	4.24
`log-a`	0.22	3.21	5.93	1.60	777.8	0.24
`log-b`	0.28	5.76	6.74	22.7	341.0	1.11
`log-c`	0.32	14.28	7.19	28.8	—	896.6
`log-d`	0.42	35.10	11.5	98.0	—	—
`bw-large-a`	0.24	2.06	0.69	6.82	0.17	0.21
`bw-large-b`	0.61	131.0	51.6	783	12.39	5.4
`TSP-7`	0.02	0.14	0.11	0.13	0.04	0.01
`TSP-10`	0.03	0.72	6.47	8.48	1.96	0.04
`TSP-15`	0.07	31.23	—	—	419.0	0.26
`TSP-30`	0.39	*out*	—	—	—	11.9
`gripper10`	0.31	—	—	—	40.38	36.3
`gripper12`	0.74	—	—	—	330.1	810.2

"—" means $> 1,500$; *out* means out of memory (768 Mbytes)

LPG is up to 4 orders of magnitude faster

Temporal Action Graphs

Temporal Action Graph (TA-graph): a triple $\langle\mathcal{A}, \mathcal{T}, \Omega\rangle$ such that

- $\mathcal{A}$: A-graph with only one action-node per level (+ "no-ops")
- $\mathcal{T}$: assignment of real values to the fact and action nodes of $\mathcal{A}$
- Ω: set of ordering constraints between action nodes of $\mathcal{A}$

Inconsistencies in TA-graphs:

- action-nodes with an unsupported precondition node

No-ops propagation [AIPS-02]:

- *No-ops* nodes used to *propagate effect nodes of actions in* $\mathcal{A}$ to the next levels
- *No-op* propagation *blocked* by action nodes that are mutex with the no-op

TA-Graphs: Temporal Values and Ordering Constraints

Assumption (in the talk): preconditions overall and effects at end

$\mathcal{T}$-values of action and fact nodes ($Time(x)$):

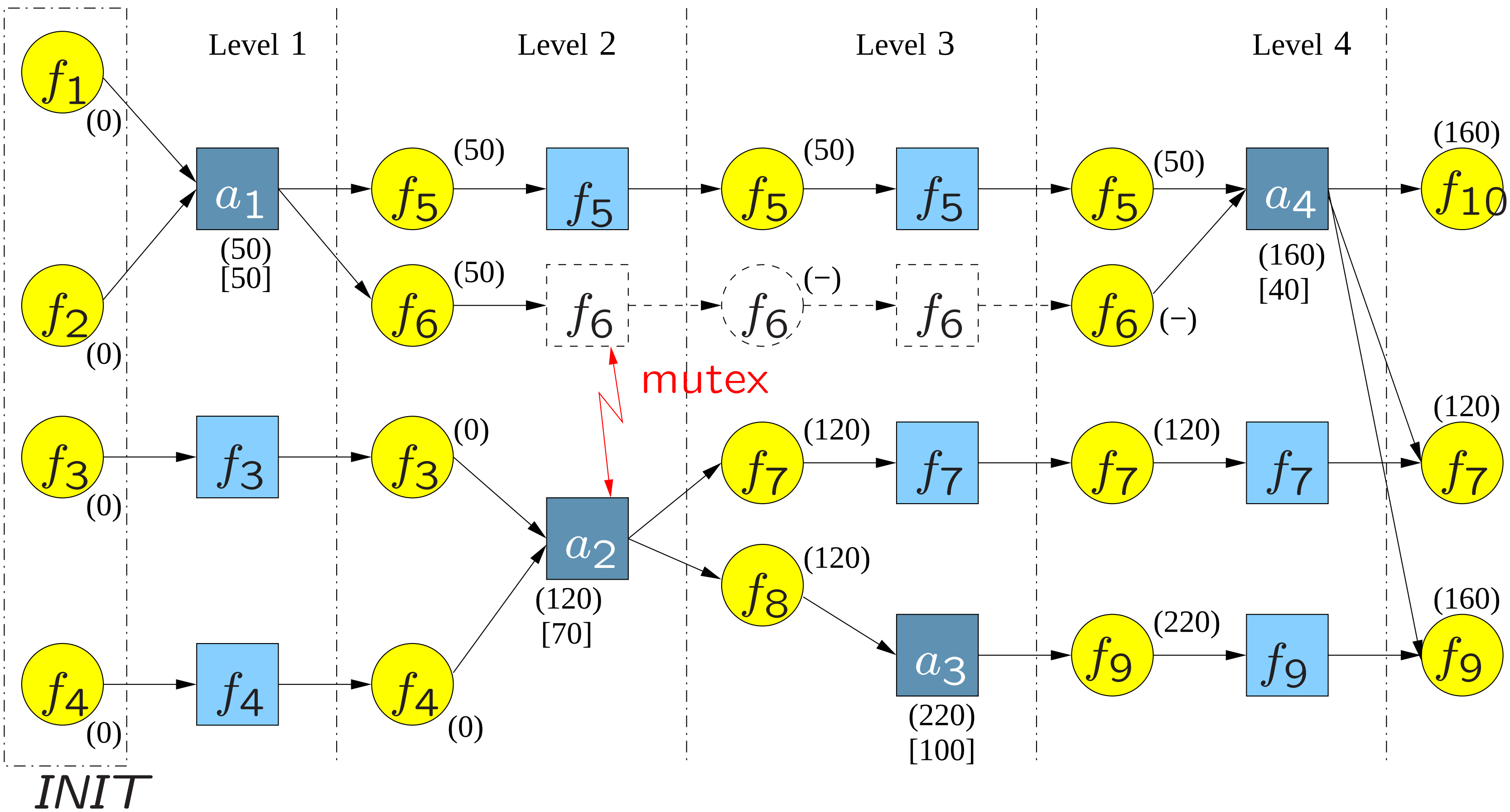
- $Time(f) =$ **minimum** *over the time values of the action-nodes supporting f*
- $Time(a) =$ **duration of** a + **maximum** *over time values of its preconditions* and times of actions preceding a according to Ω

Two types of Ω-constraints ($\prec_C$ and $\prec_E$):

- $a \prec_C b \in \Omega$ *if an effect of a is used to achieve a precondition of b*
- $a \prec_E b \in \Omega$ *if a and b are mutually exclusive and* Level(a) $<$ Level(b)

Plan action start times derived from the $Time$-values ($\leadsto$ parallelism)

Example of TA-Graph



$$\Omega = \{a_1 \prec_C a_4,\ \ a_2 \prec_C a_3\} \cup \{a_1 \prec_E a_2,\ \ a_2 \prec_E a_4\}$$

Causal precedence *Exclusion* precedence

Local Search in the Space of TA-Graphs

Initial state: TA-graph containing only a_{start}, a_{end} (+ no-ops)

Search steps: graph changes removing an inconsistency σ at level l:

- *Inserting an action node at a level l' preceding l*

  $\Rightarrow$ TA-graph extended by one level (all actions from l' shifted forward)

- *Removing the action node a responsible of σ*

  $\Rightarrow$ Action nodes used only to support the preconds of a are removed as well

Goal states (solution TA-graphs): TA-graphs $\langle \mathcal{A}, \mathcal{T}, \Omega \rangle$ where

- $\mathcal{A}$ is a solution graph
- $\mathcal{T}$ is consistent with Ω and the duration of the actions in $\mathcal{A}$
- Ω is consistent, and if a and b are mutex, $\Omega \models a \prec b$ or $\Omega \models b \prec a$.

Maintaining Temporal Information During Search

When an action node a is **added** to support a precondition of b:

- $\boxed{\Omega = \Omega \cup \{a \prec b\}}$
- $\forall c \; mutex(a, c) \; \& \; Level(a) < Level(c)$: $\boxed{\Omega = \Omega \cup \{a \prec c\}}$
- $\forall d \; mutex(a, d) \; \& \; Level(d) < Level(a)$: $\boxed{\Omega = \Omega \cup \{d \prec d\}}$
- $\forall$ action/fact node x "temporally influenced" by a: $\boxed{Time(x) \text{ is updated}}$

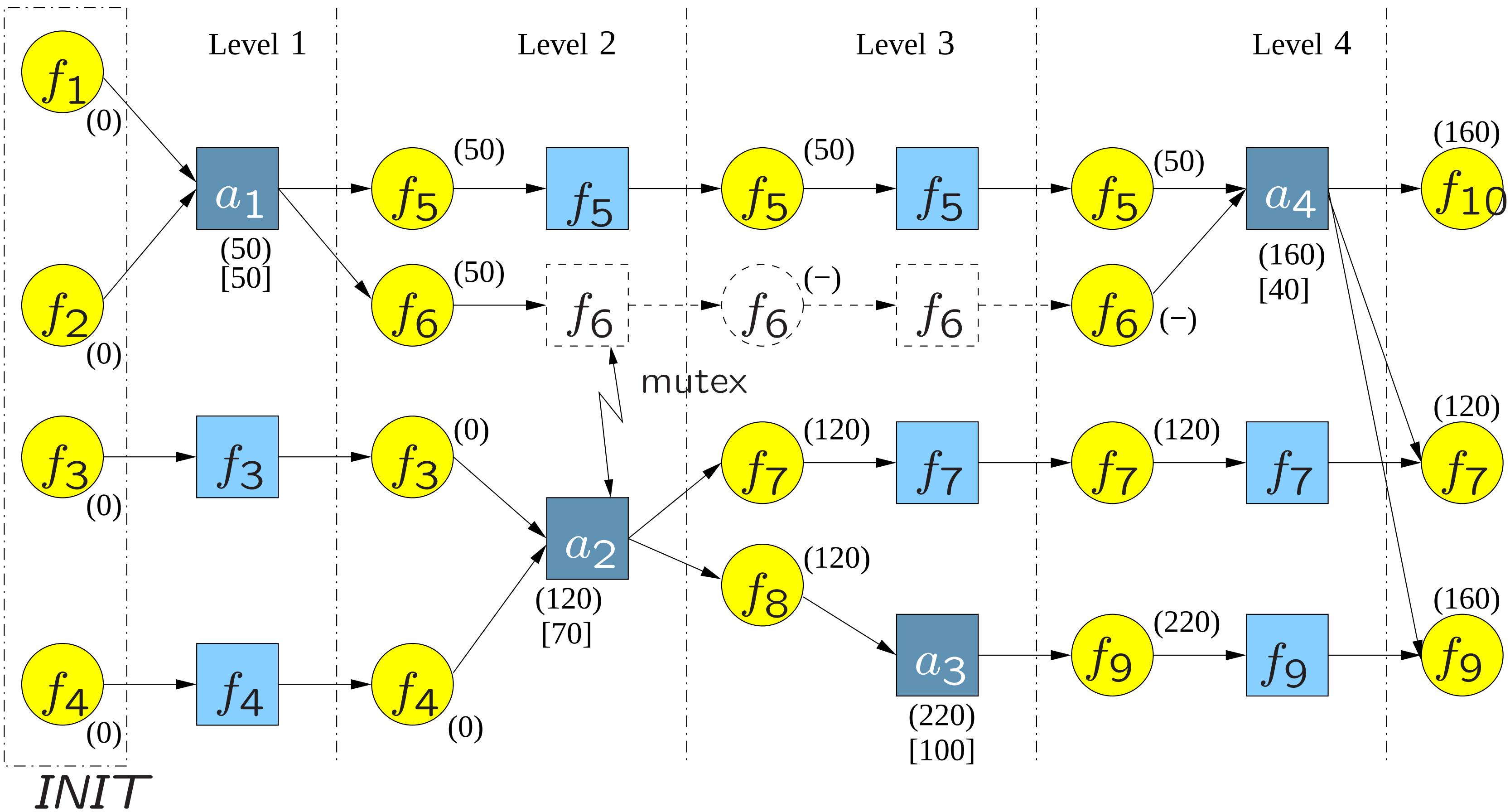
When an action node a with unsupported precondition is **removed**:

- $\forall$ ordering constraint ω involving a: $\boxed{\Omega = \Omega - \{\omega\}}$
- $\forall$ action/fact node x "temporally influenced" by a: $\boxed{Time(x) \text{ is updated}}$

$\Rightarrow$ *The computation of $Time(x)$ takes account of different types of preconditions (`overall`, `at start`, `at end`) and effects (`at start`, `at end`).*

Example of Action Insertion

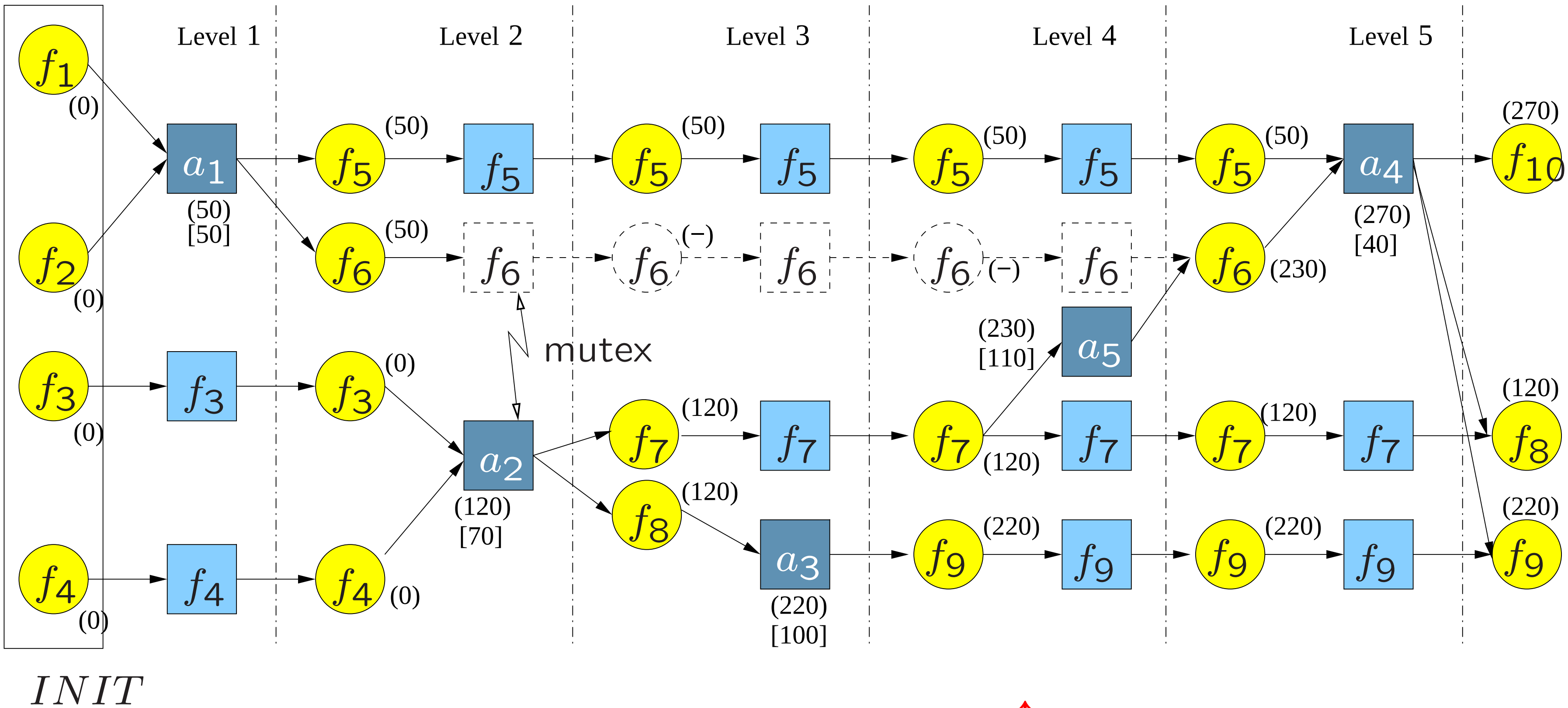
(original graph)



$$\Omega = \{a_1 \prec_C a_4, \;\; a_2 \prec_C a_3\} \cup \{a_1 \prec_E a_2, \;\; a_2 \prec_E a_4\}$$

Causal precedence *Exclusion* precedence

TA-graph after Insertion of a_5



$\uparrow$ *new action and level*

a_5 = new action to support f_6

$$\Omega = \{a_1 \prec_C a_4,\ a_2 \prec_C a_3,\ \boxed{a_5 \prec_C a_4}\} \cup \{a_1 \prec_E a_2,\ a_2 \prec_E a_4\}$$

Action Evaluation Function (E)

Estimates the cost of inserting a ($E(a)^i$) or removing a ($E(a)^r$):

$$E(a)^i = \alpha \cdot Exec\_cost(a)^i + \beta \cdot Temporal\_cost(a)^i + \gamma \cdot Search\_cost(a)^i$$

$$E(a)^r = \alpha \cdot Exec\_cost(a)^r + \beta \cdot Temporal\_cost(a)^r + \gamma \cdot Search\_cost(a)^r$$

The three terms of E estimate the

- *increase of the plan execution cost:* $\boxed{Exec\_cost}$
- *end time of a:* $\boxed{Temporal\_cost}$
- *increase of # of the search steps to reach a solution:* $\boxed{Search\_cost}$

α, β and γ normalize the terms and weight their relative importance

(*dynamically computed* during search – see paper)

Relaxed Plans for $E(a)^i$ (basic idea)

- Compute a **relaxed plan** π (no action interference) for

  (1) the unsupported preconds of a and

  (2) the preconds of actions "threatened by a" at the next levels

  $\boxed{a \textit{ threatens } p} = p$ *is supported and an effect of* a *denies* p

$\Longrightarrow$ $Search\_cost(a) = \#$ of actions in π + $\#$ of their *threats*

$Temporal\_cost(a) =$ end time of subplan for (1) + duration of a

$Execution\_cost(a) =$ sum of the costs of the actions in π

- π constructed *in the context of the current TA-graph $\mathcal{A}$*:
  - actions in $\mathcal{A}$ at preceding levels define the *initial state* for π
  - actions for π threatening other actions in $\mathcal{A}$ are *penalized*

Relaxed Plans for $E(a)^i$ (basic idea, cont.)

π constructed by a **backward process** from $Preconds(a)$ **and** $Threats(a)$

$\boxed{\mathit{INIT}_l}$ = state reached by the actions preceding the level l of a

b is the **best action** to achieve a (sub)goal g in π if

(1) g is an effect of b, and all preconds of b reachable from INIT_l

(2) satisfying the preconds of b from INIT_l requires a min number of actions

(3) b threatens a min number of preconds of actions in the TA-graph

$\Downarrow$

$$\boxed{\mathit{BestAction}(g)} = \underset{b' \rightarrow (1)}{ARGMIN} \left\{ \underset{p \in Pre(b') - F}{MAX} Num\_acts(p, l) + |Threats(b')| \right\}$$

(F = preconds already achieved in π)

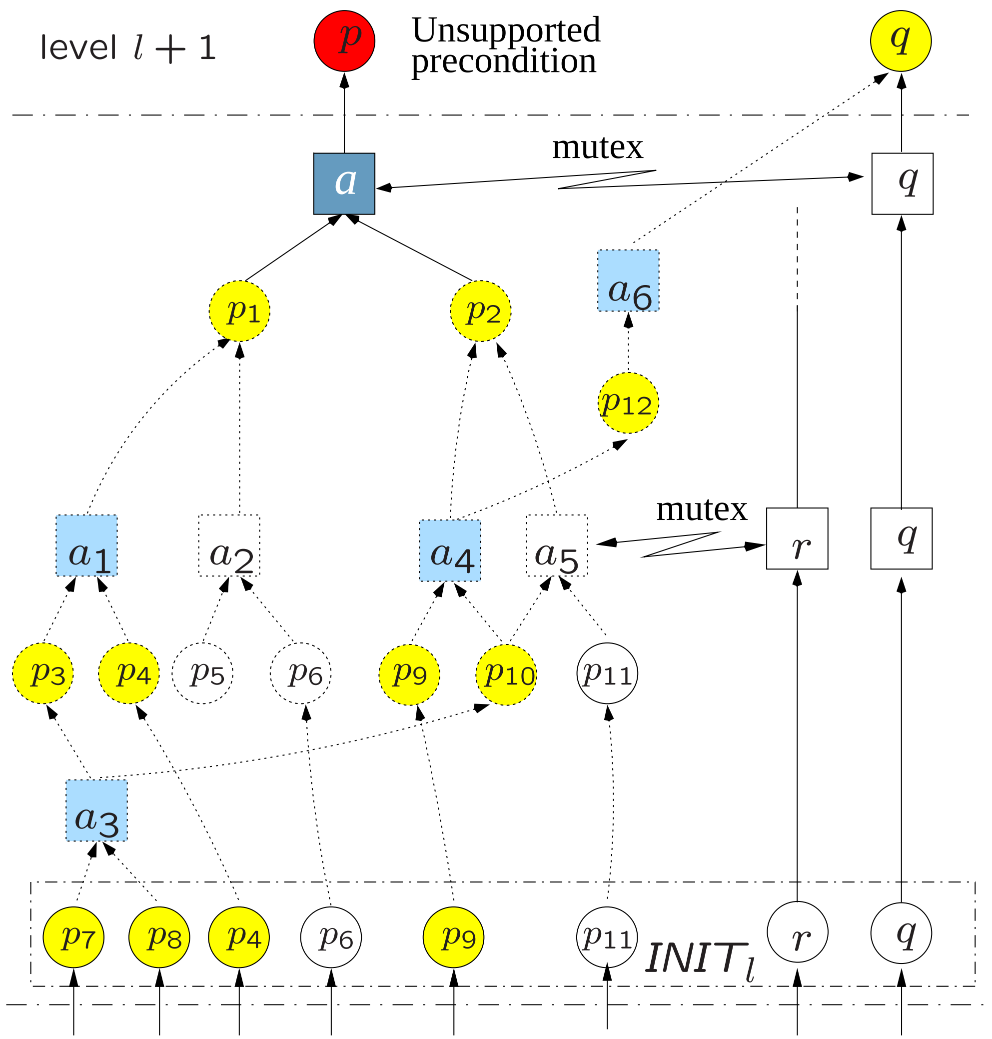
$\boxed{Num\_acts(p, l)}$ = *estimate of minimum number of actions required to achieve p from INIT_l (dynamically computed).*

Relaxed Plan for $E(a)^i$ (example)

Fact	Num_acts
p_1	2
p_2	2
p_3	1
p_5	6
p_{10}	1
p_{12}	2

Fact	$Time$
p_4	170
p_6	300
p_7	50
p_8	30
p_9	170
p_{11}	30

Action	$Duration$
a	30
a_1	70
a_3	100
a_4	30
a_6	90



Relaxed plan $= \boxed{\{a_1, a_3, a_4\} \cup \{a_6\}}$

$End\_time(\{a_1, a_3, a_4\}) = 240$

RelaxedPlan($G, INIT_l, A$)

$t \leftarrow \underset{g \in G \cap INIT_l}{MAX} Time(g)$;

$G \leftarrow G - INIT_l$; $ACTS \leftarrow A$;

$F \leftarrow \bigcup_{a \in ACTS} Add(a)$;

$t \leftarrow MAX \left\{ t, \underset{g \in G \cap F}{MAX} T(g) \right\}$;

while $G - F \neq \emptyset$

  $g \leftarrow$ a fact in $G - F$;

  $bestact \leftarrow Bestaction(g)$;

  $Rplan \leftarrow$ RelaxedPlan($Pre(bestact), INIT_l, ACTS$);

  forall $f \in Add(bestact) - F$

    $T(f) \leftarrow End\_time(Rplan) + Duration(bestact)$;

  $ACTS \leftarrow Aset(Rplan) \cup \{bestact\}$;

  $F \leftarrow \bigcup_{a \in ACTS} Add(a)$;

  $t \leftarrow MAX\{t, End\_time(Rplan) + Duration(bestact)\}$;

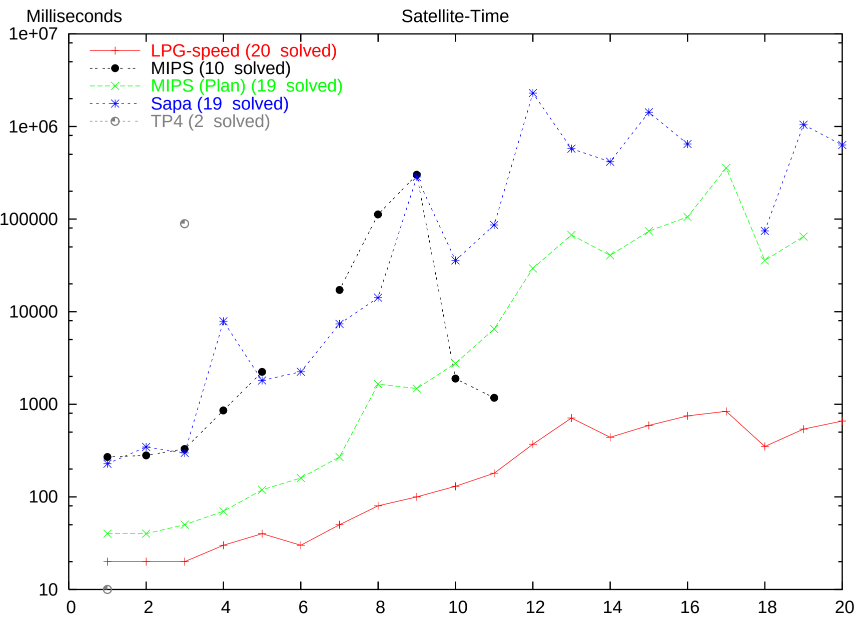
return $\langle ACTS, t \rangle$.

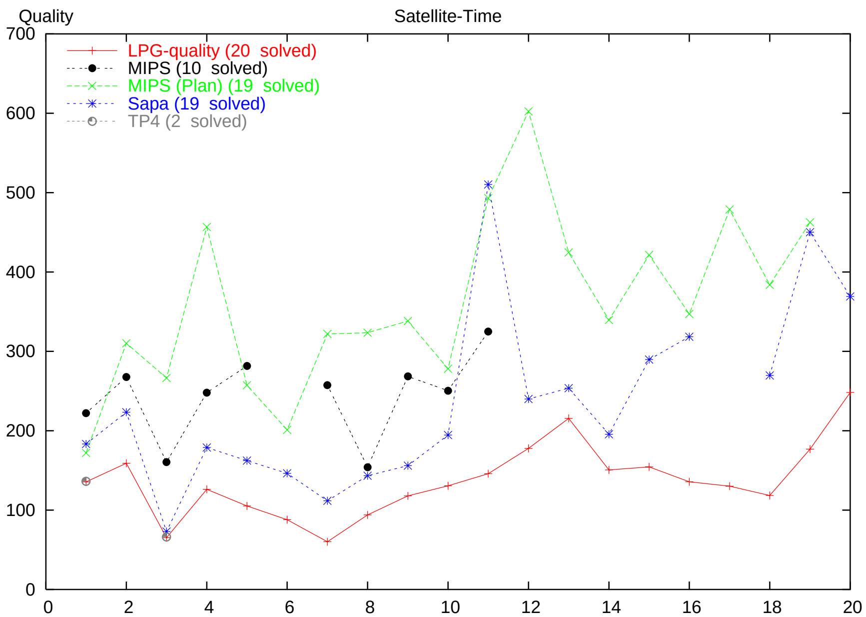
EvalAdd(a)

1. $INIT_l \leftarrow Supported\_facts(Level(a));$
2. $Rplan \leftarrow \mathsf{RelaxedPlan}(Pre(a), INIT_l, \emptyset);$
3. $t_1 \leftarrow MAX\{0, MAX\{Time(a') \mid \Omega \models a' \prec a\}\};$
4. $t_2 \leftarrow MAX\{t_1, End\_time(Rplan)\};$
5. $A \leftarrow Aset(Rplan) \cup \{a\};$
6. $Rplan \leftarrow \mathsf{RelaxedPlan}(Threats(a), INIT_l - Threats(a), A);$
7. **return** $\langle Aset(Rplan), t_2 + Duration(a)\rangle.$

$$E(a)^i \begin{cases} Execution\_cost(a)^i = \sum_{a' \in\ Aset(\mathsf{EvalAdd}(a))} Cost(a') \\ Temporal\_cost(a)^i = End\_time(\mathsf{EvalAdd}(a)) \\ Search\_cost(a)^i = |Aset(\mathsf{EvalAdd}(a))| + \\ \qquad\qquad \sum_{a' \in Aset(\mathsf{EvalAdd}(a))} |Threats(a')| \end{cases}$$

Experimental Results
(All IPC 2004 Planners)





LPG data are median values over five runs

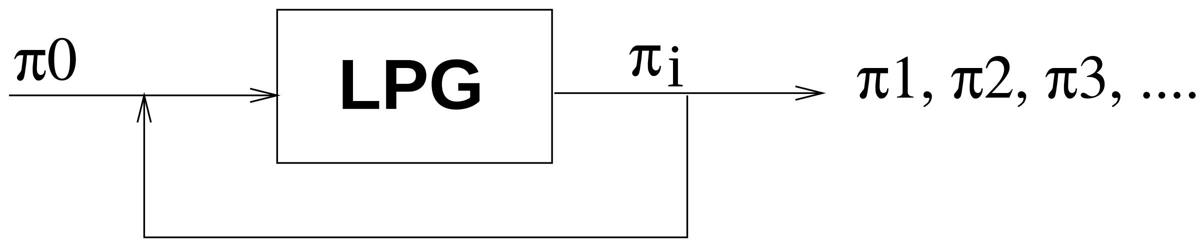
Plan quality: minimization of a metric expression

CPU-time: milliseconds in logarithmic scale

Incremental Plan Quality

- Generation of a *sequence* of valid plans.

- Each plan improves the quality of the previous one.

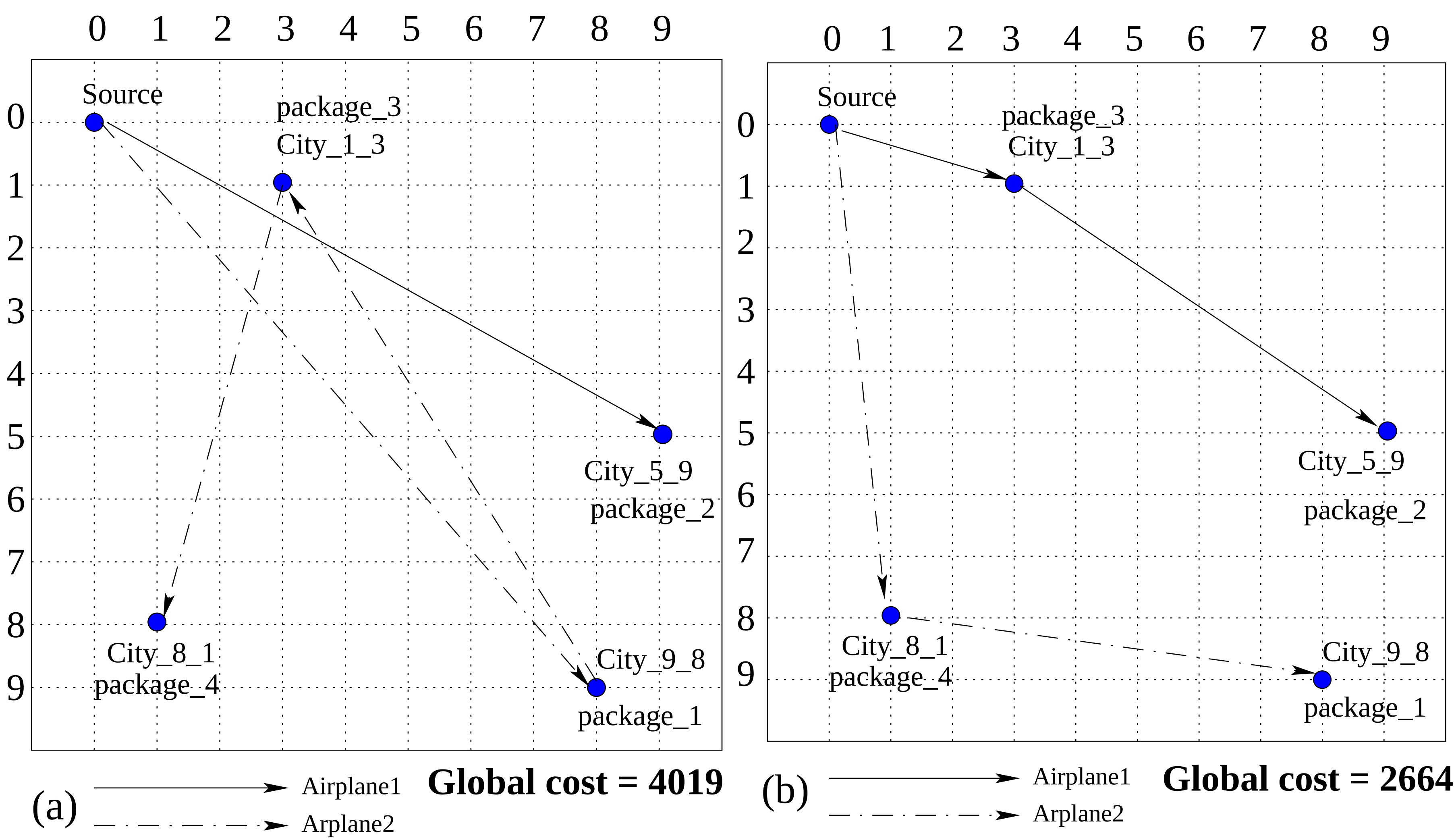


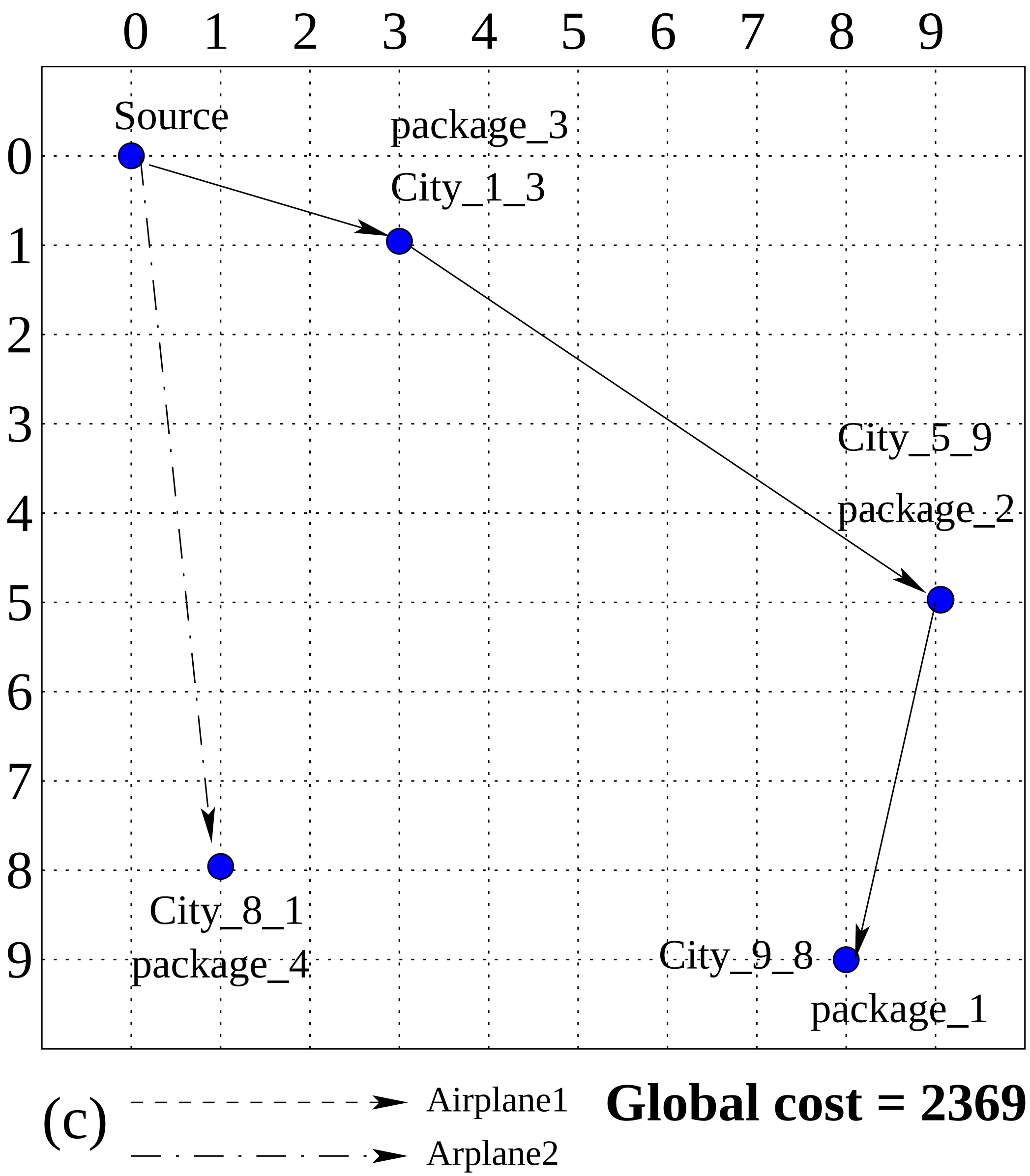
π_0 = initial $\mathcal{A}$-graph
π_1 = first valid plan computed by LPG
π_i = i-th valid plan (of quality better than π_{i-1})

- Each computed plan (*with some forced inconsistencies*) becomes the initial $\mathcal{A}$-graph of a new search.

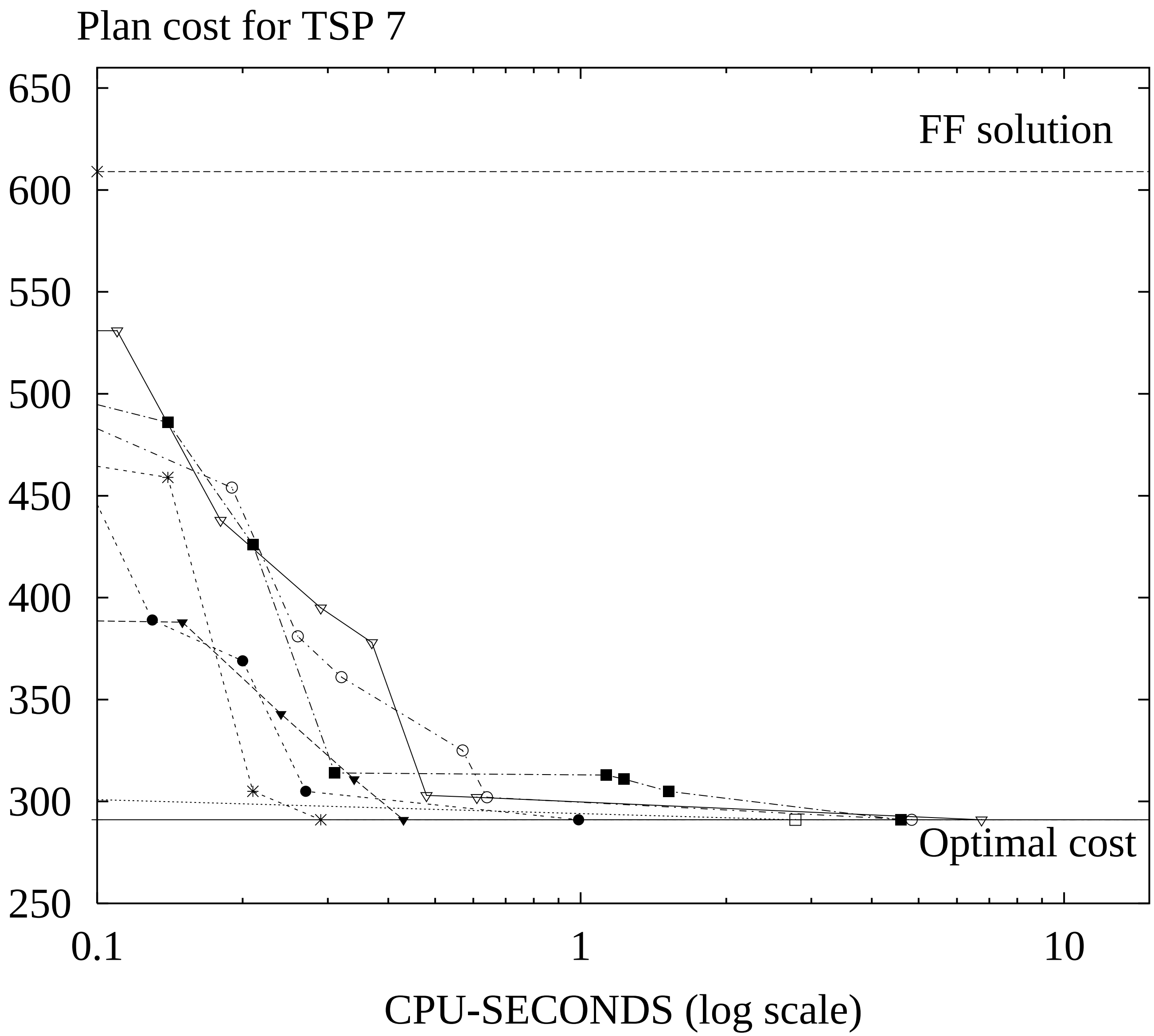
$\Rightarrow$ **Anytime process**: the system can be stopped at any time to give the best plan computed so far.

Experimental Results: Plan Quality

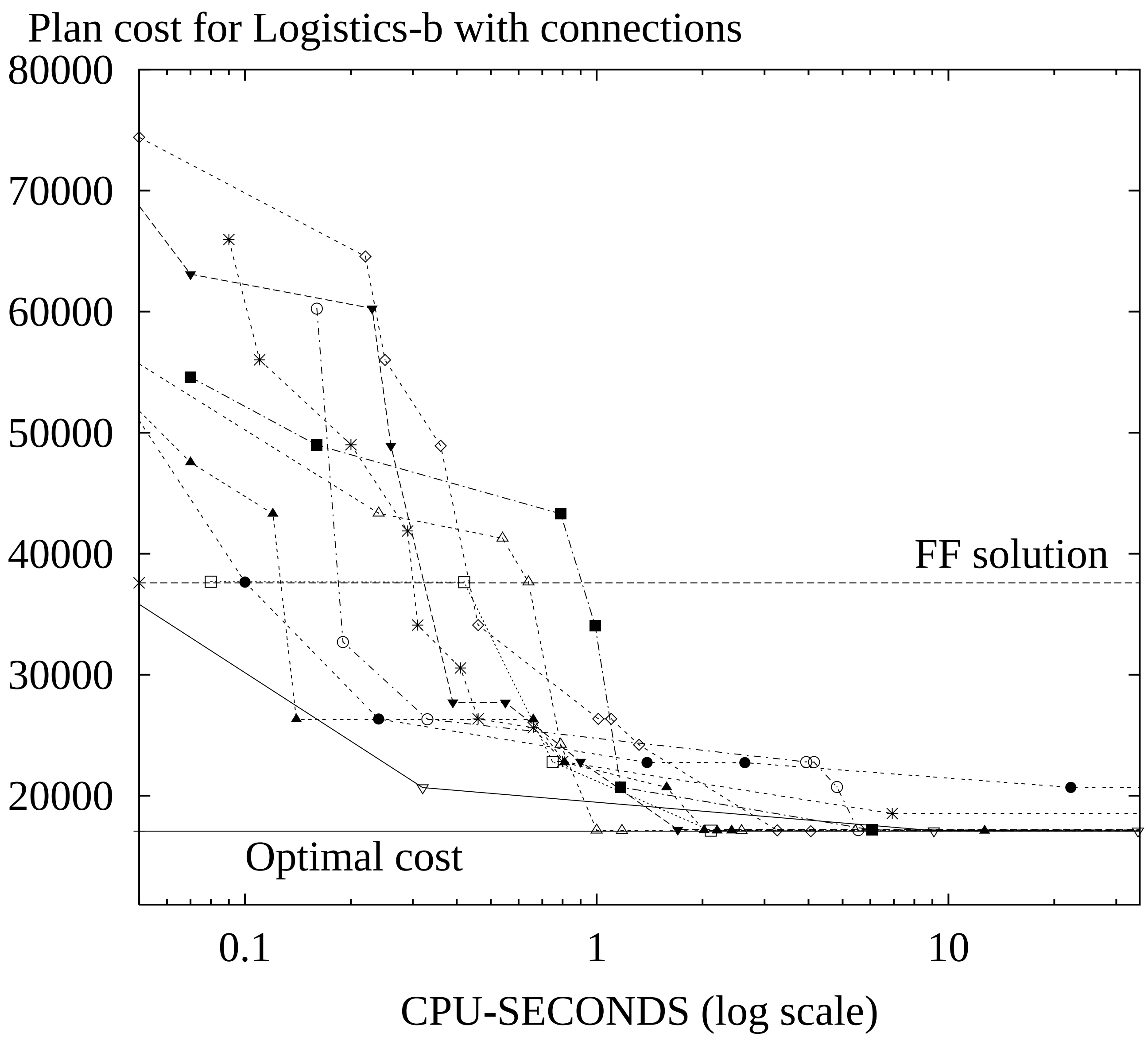



0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
Source
package_3
City_1_3
City_5_9
package_2
City_8_1
package_4
City_9_8
package_1
(c)
Airplane1
Arplane2
Global cost = 2369

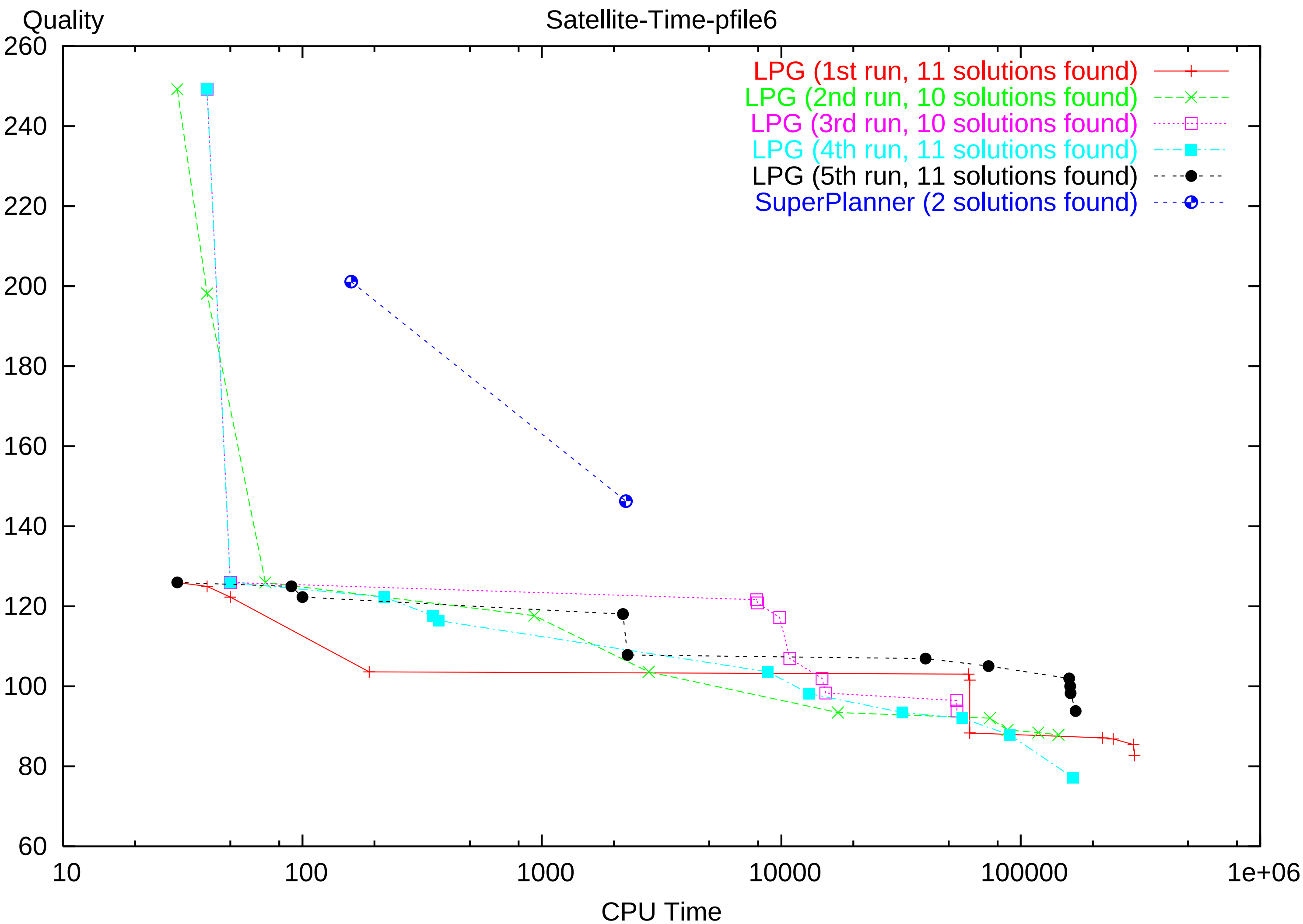
Incremental Plan Quality: TSP



Incremental Plan Quality: Logistics



Incremental Plan Quality



Incremental Plan Quality with InLPG
(demo)

Timed Literals & Exogenous Events

- Useful to represent **predictable exogenous events** that happen at known times, and cannot be influenced by the planning agent.

  For instance (using PDDL notation):

  ```
  (at 8 (open-fuelstation city1))
  (at 12 (not (open-fuelstation city1)))
  (at 15 (open-fuelstation city1))
  (at 19 (not (open-fuelstation city1)))
  ```

- Timed literals in the preconditions of an action impose **scheduling constraints** to the action:

  If `(refuel car city1)` has `over all` condition `open-fuelstation`, *it must be executed during the time window* $[8, 12]$ *or* $[15, 19]$.

  (Similarly for other types of action conditions)

DTP Constraints for PDDL2.2 Domains

- **Action ordering constraints**

  E.g., a must end (a^+) before the start of b (b^-): $a^+ \prec b^-$
  $$a^+ \prec b^- \equiv a^+ - b^- \leq 0$$

- **Duration Constraints**

  E.g., $(a^+ - a^- \leq 10) \wedge (a^- - a^+ \leq -10))$

- **Scheduling constraints** (in *compact* DTP-form):

$$\bigvee_{w \in W(p)} \left(\left(a_{start} - a^- \leq -w^-\right) \wedge \left(a^+ - a_{start} \leq w^+\right)\right).$$

If p `over all` timed condition with windows $W(p) = \{w_1, \ldots, w_n\}$
(a_{start} is a special instantaneous action preceding all others)

Note: we can compile all timed conditions of an action into a single `over all` timed precondition (with more time windows)

Temporally Disjunctive LA-graph

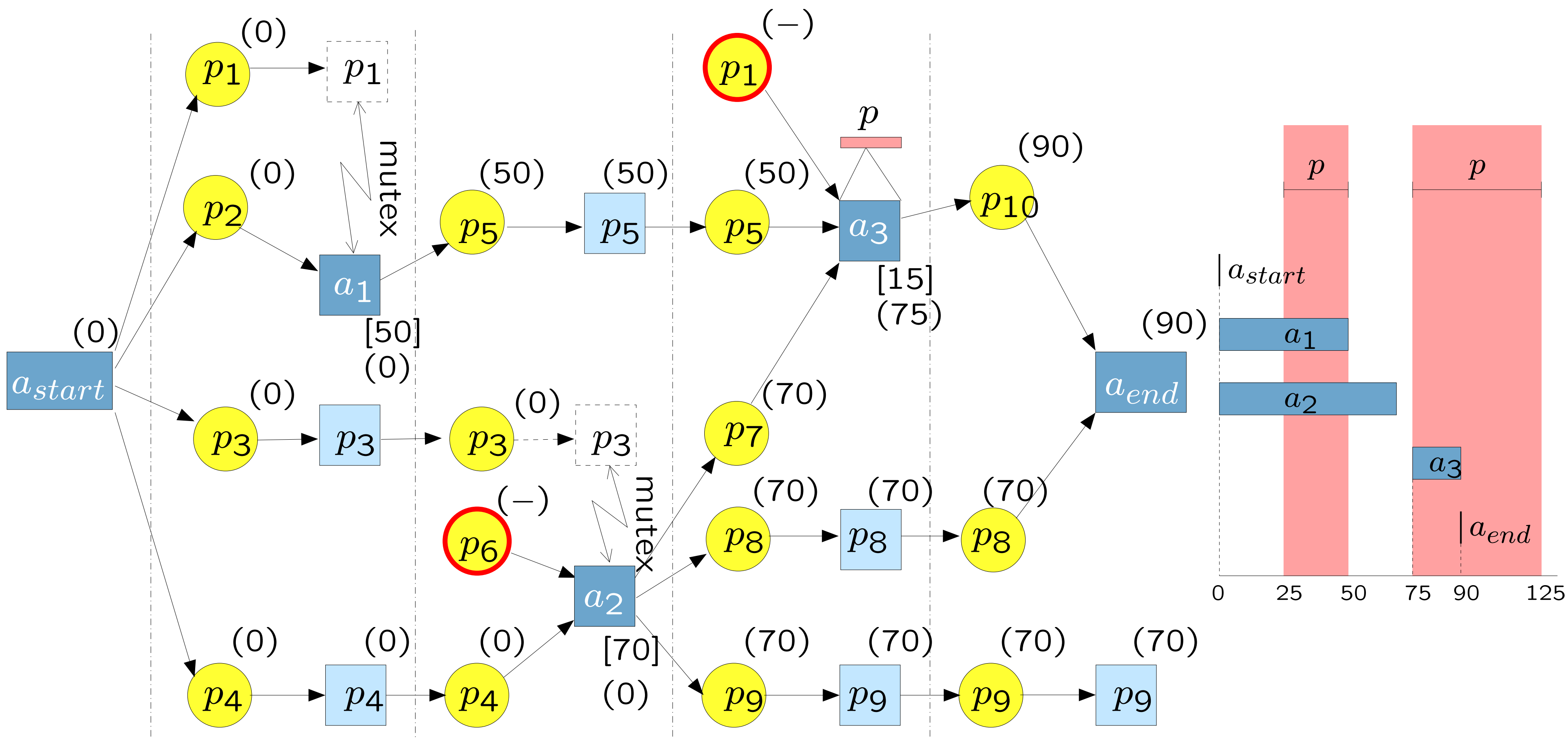
A **Temporally Disjunctive Action Graph (TDA-graph)** is a 4-tuple $\langle \mathcal{A}, \mathcal{T}, \mathcal{P}, \mathcal{C} \rangle$ where

- $\mathcal{A}$ is a linear action graph;
- $\mathcal{T}$ is an assignment of real values to the nodes of $\mathcal{A}$ (determined by solving the DTP $\langle \mathcal{P}, \mathcal{C} \rangle$)
- $\mathcal{P}$ is the set of time point variables representing the start/end times of the actions labeling the action nodes of $\mathcal{A}$;
- $\mathcal{C}$ is a set of ordering constraints, duration constraints and scheduling constraints involving variables in $\mathcal{P}$.

Propositional flaw: unsupported precondition node

Temporal flaw: action *un*scheduled by $\mathcal{T}$ ($\langle \mathcal{P}, \mathcal{C} \rangle$ is unsolvable)

Example of TDA-graph



$$\mathcal{C} = \begin{cases} a_1^+ \prec a_3^-, \ \ a_2^+ \prec a_3^-, \ \ a_{start} \prec a_i^-, \ \ a_i^+ \prec a_{end} \quad (i = 1 \cdots 3) \\ a_1^+ - a_1^- = 50, \ \ a_2^+ - a_2^- = 70, \ \ a_3^+ - a_3^- = 15 \\ W_p = \{[25, 50), [75, 125)\} \ \Rightarrow \ a_3 \text{ during } [25, 50] \text{ or } [75, 125] \end{cases}$$

Temporal values in a TDA-graph

- The DTP $\mathcal{D} = \langle \mathcal{P}, \mathcal{C} \rangle$ of a TDA-graph $\langle \mathcal{A}, \mathcal{T}, \mathcal{P}, \mathcal{C} \rangle$ represents a set of *induced STPs*

- **Induced STP**: satisfiable STP with all unary constraints of $\mathcal{C}$ and one disjunct (time window) for each disjunctive constraint

- **Optimal induced STP for** a_{end}: an induced STP with a solution assigning to a_{end} the minimum possible value

- **Optimal schedule for** $\mathcal{D}$ **=** $\mathcal{T}$**-values**: $\Longrightarrow$ *optimal solution of an optimal induced STP for* a_{end}

Can be computed in polytime by a backtrack-free algoritm!

Solving the DTP of a TDA-graph

Finding a solution for a DTP $\Rightarrow$ solving a meta CSP:
[Stergiou & Koubarakis, Tsamardinos & Pollack, and others]

- *Meta variables*: constraints of the DTP
- *Meta variable values*: constraint disjuncts
- *Implicit meta constraint*: the values (constraint disjuncts) of the meta variables form a satisfiable STP

Solution of the meta CSP = complete induced STP of the DTP

In general NP-hard, but polynomial for the DTP of a TDA-graph:

Theorem: *Given the DTP $\mathcal{D}$ of a TDA-graph, deciding satisfiability of $\mathcal{D}$ and finding an optimal schedule for $\mathcal{D}$ (if one exists) can be accomplished in polynomial time.*

Solving the DTP of a TDA-Graph

[Stergiou & Koubarakis '00, Tsamardinos & Pollack '03]

Solve-DTP(X, S)
  if $X = \emptyset$ **then stop** and **return** S;
  $x \leftarrow$ SelectVariable(X); $X' \leftarrow X - \{x\}$;
  while $D(x) \neq \emptyset$ **do**
    $d \leftarrow$ SelectValue($D(x)$); $D(x) \leftarrow D(x) - \{d\}$;
    $D'(x) \leftarrow D(x)$;
    if ForwardChecking-DTP(X', S) **then** Solve-DTP($X', S \cup \{d\}$);
    $D(x) \leftarrow D'(x)$;
  return *fail*;   /* backtracking */

ForwardChecking-DTP(X, S)
  forall $x \in X$ **do**
    forall $d \in D(x)$ **do**
      if *not* Consistency-STP($S \cup d$) **then** $D(x) \leftarrow D(x) - \{d\}$;
      if $D(x) = \emptyset$ **then return** *false*;
  return *true*.

$SelectVariable$: variables ordered w.r.t. the levels of the TDA-graph

$SelectValue$: values ordered w.r.t. the windows in the constraint

$\Longrightarrow$ **No backtracking + Optimality of the induced STP!**

Planning with TDA-Graphs

Initial state: TDA-graph containing only a_{start} (initial state), a_{end} (problem goals) + no-ops

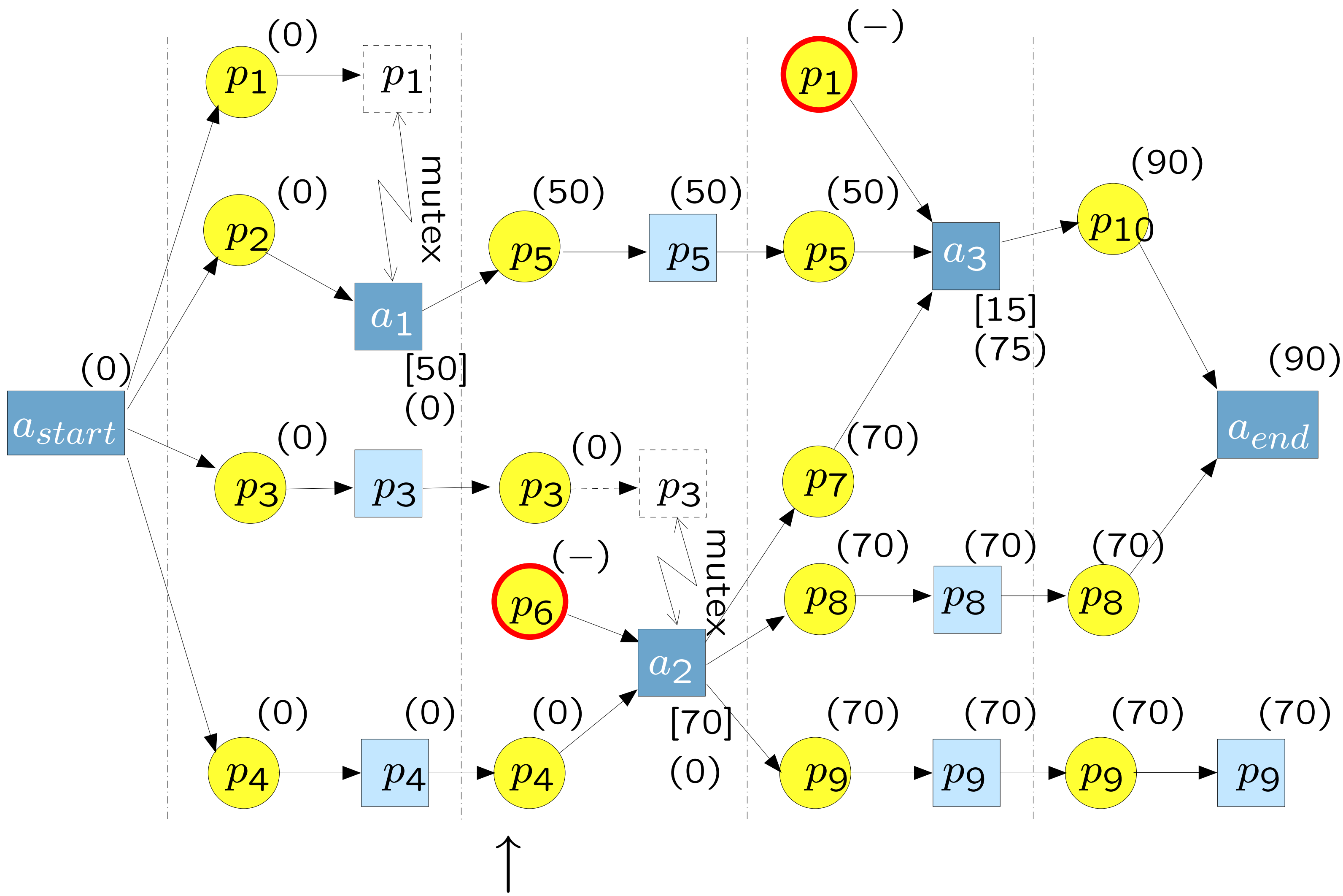
Goal states: TDA-graphs without flaws (*solution TDA-graph*)

Basic search steps: graph changes for repairing a flaw σ at a level ℓ

- *Inserting an action node at a level $\ell' < \ell$ (for propositional flaws)*
- *Removing an action node*:
  - at a level $\ell' \leq \ell$ (if σ is a propositional flaw), or
  - an action at $\ell' < \ell$ *decreasing* the earliest start time of σ (if σ is a temporal flaw = unscheduled action node).

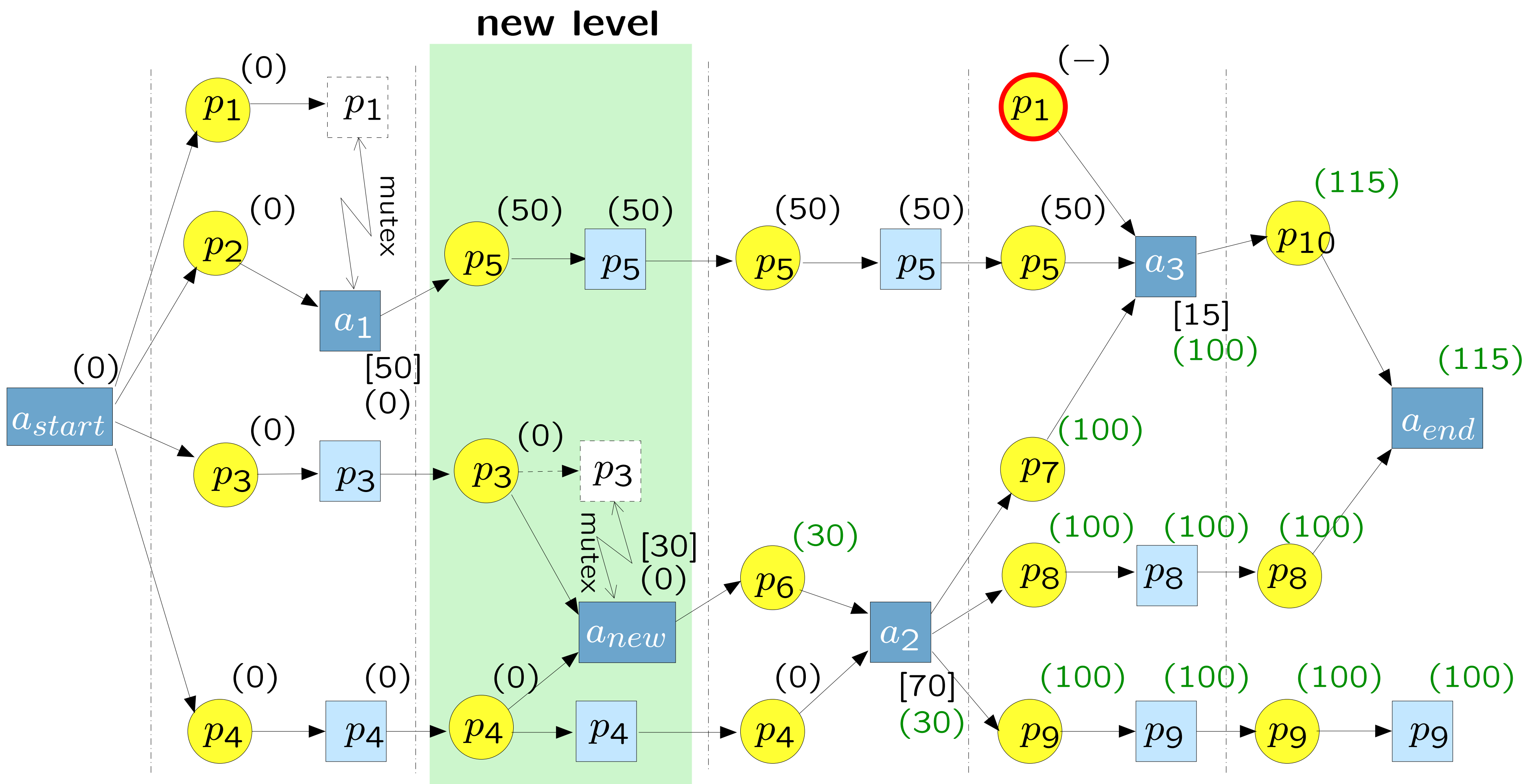
The DTP of the TDA-graph is **dynamically updated** at each search step

Example: TDA-graph before Action Insertion



Selected flawed level (propositional flaw: p_6)

TDA-graph after Insertion of a_{new}



New temporal variables/constraints: $a_{new}^{+} \prec a_2^{-}$, Dur($a_{new}$) = 30, $Win(a_{new})$=[0, +∞]

In general: also constraints for mutex actions; actions can become unscheduled