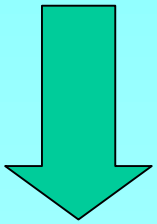
LPG
Quantità Numeriche e Raggiungibilità

- Dr. Ivan Serina
- serina@ing.unibs.it

Università degli studi di Brescia

Estensioni: PDDL 2.1

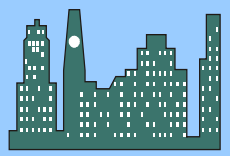
- Gestione esplicita del tempo
  - azioni non istantanee
- Gestione di risorse numeriche
  - azioni che producono/consumano risorse



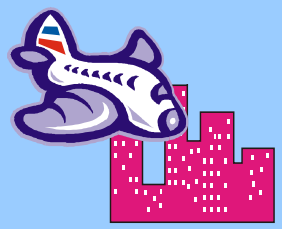
PDDL 2.1

- Modello del dominio più realistico

Esempio di azione



Milano

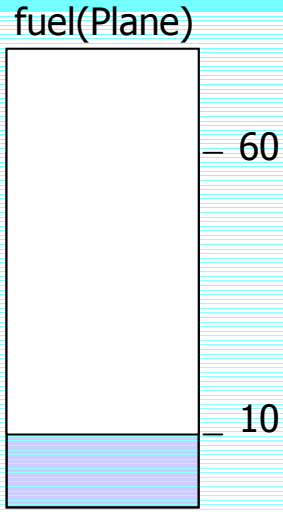


Parigi

fly (Plane,Milano,Parigi)

Durata = distanza(Milano,Parigi) / velocità(Plane)

- Precondizioni: *at(Plane,Milano)*
  fuel(Plane) >50
- Effetti (add): *at(Plane,Parigi)*
  fuel(Plane) -=50
- Effetti (del): not *at(Plane,Milano)*

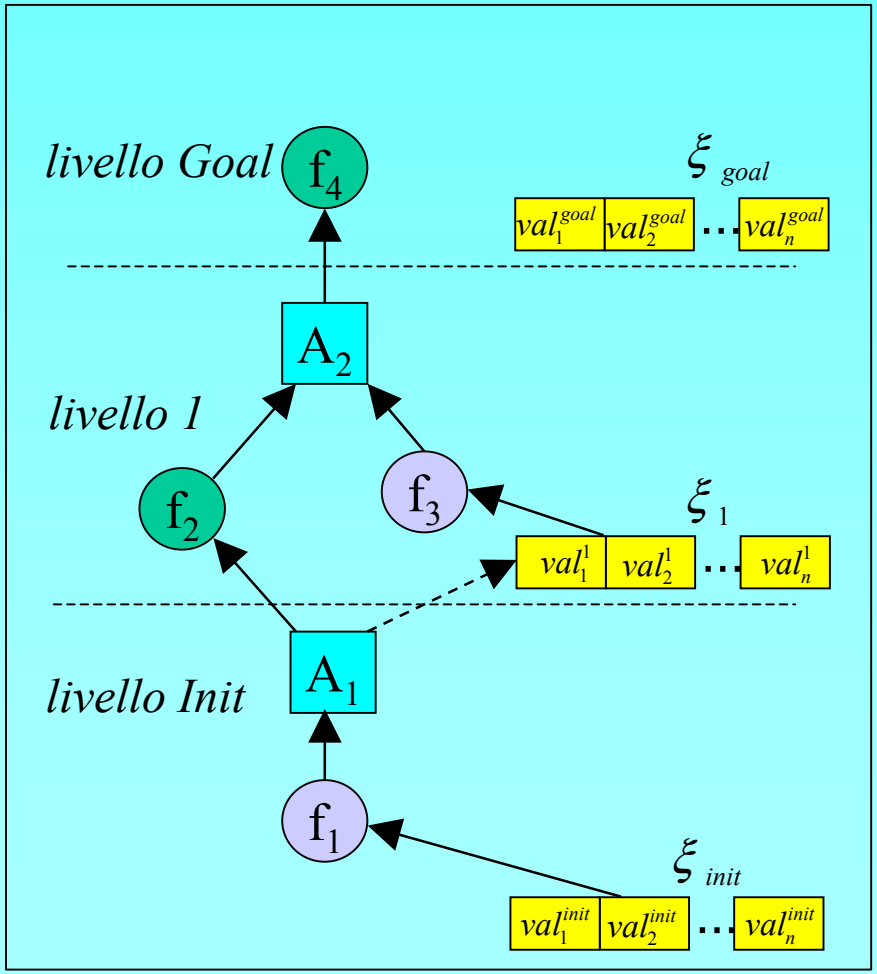


Action Graph e NA-Graph

- Action Graph $\mathcal{A}$: sottografo di $\mathcal{G}$
- Linear Action Graph: Action Graph $\mathcal{A}$ di $\mathcal{G}$ con al più un nodo azione per livello
- Numeric Action Graph: coppia $\langle \mathcal{A}, \Xi \rangle$
  - $\mathcal{A}$ è un Linear Action Graph
  - Ξ è un insieme i cui elementi ξ_i sono un insieme di coppie del tipo

$$\xi_i = \{ \langle v_1^i, val_1^i \rangle, \ldots, \langle v_j^i, val_j^i \rangle, \ldots, \langle v_n^i, val_n^i \rangle \}$$

Esempio di NA-Graph



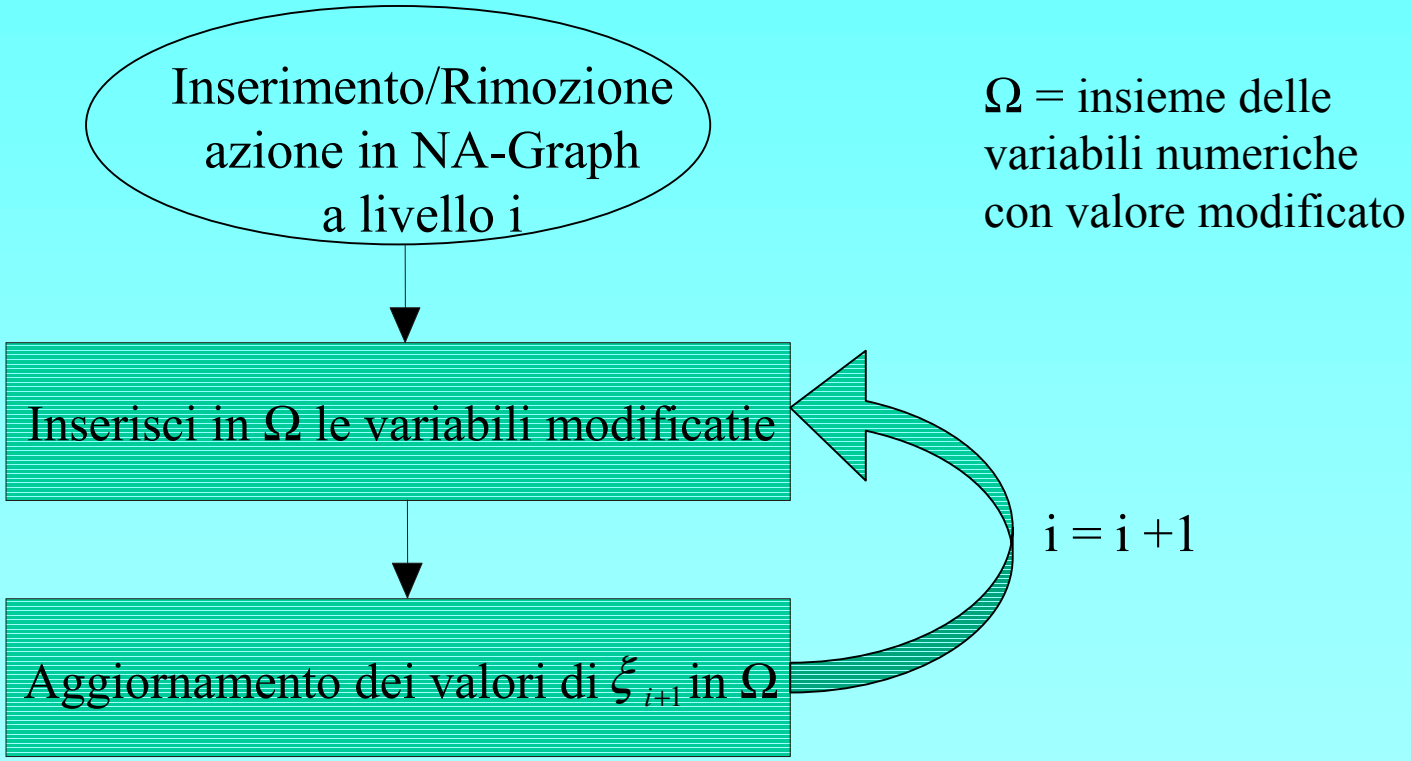
Precondizioni Logiche

Vincoli numerici

Azioni

Valori numerici in ξ_i

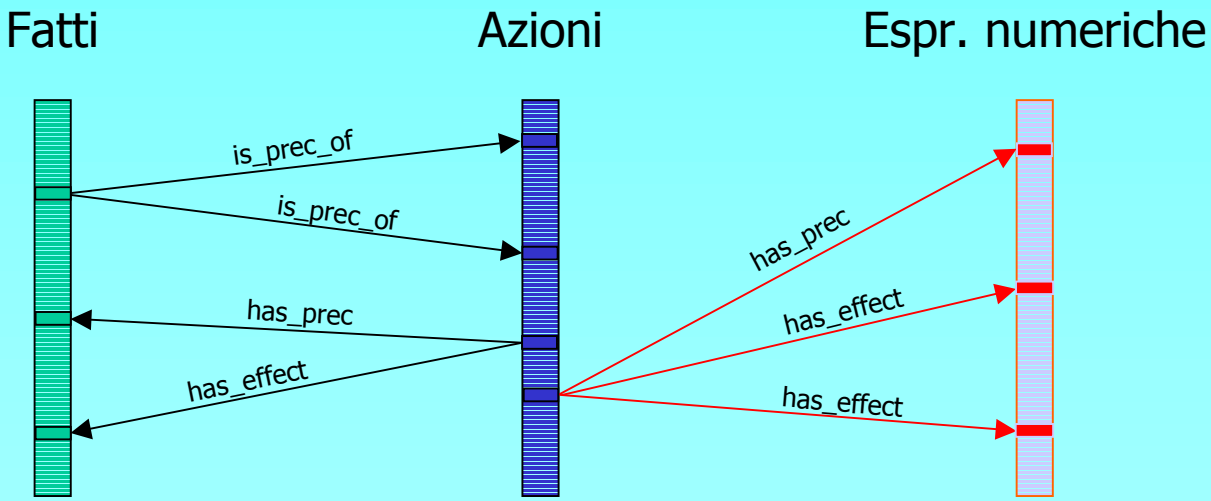
Aggiornamento dei valori numerici



Ω = insieme delle variabili numeriche con valore modificato

Fatti, azioni ed espressioni

- Uso del *Connectivity Graph* di J.Hoffmann (2000)

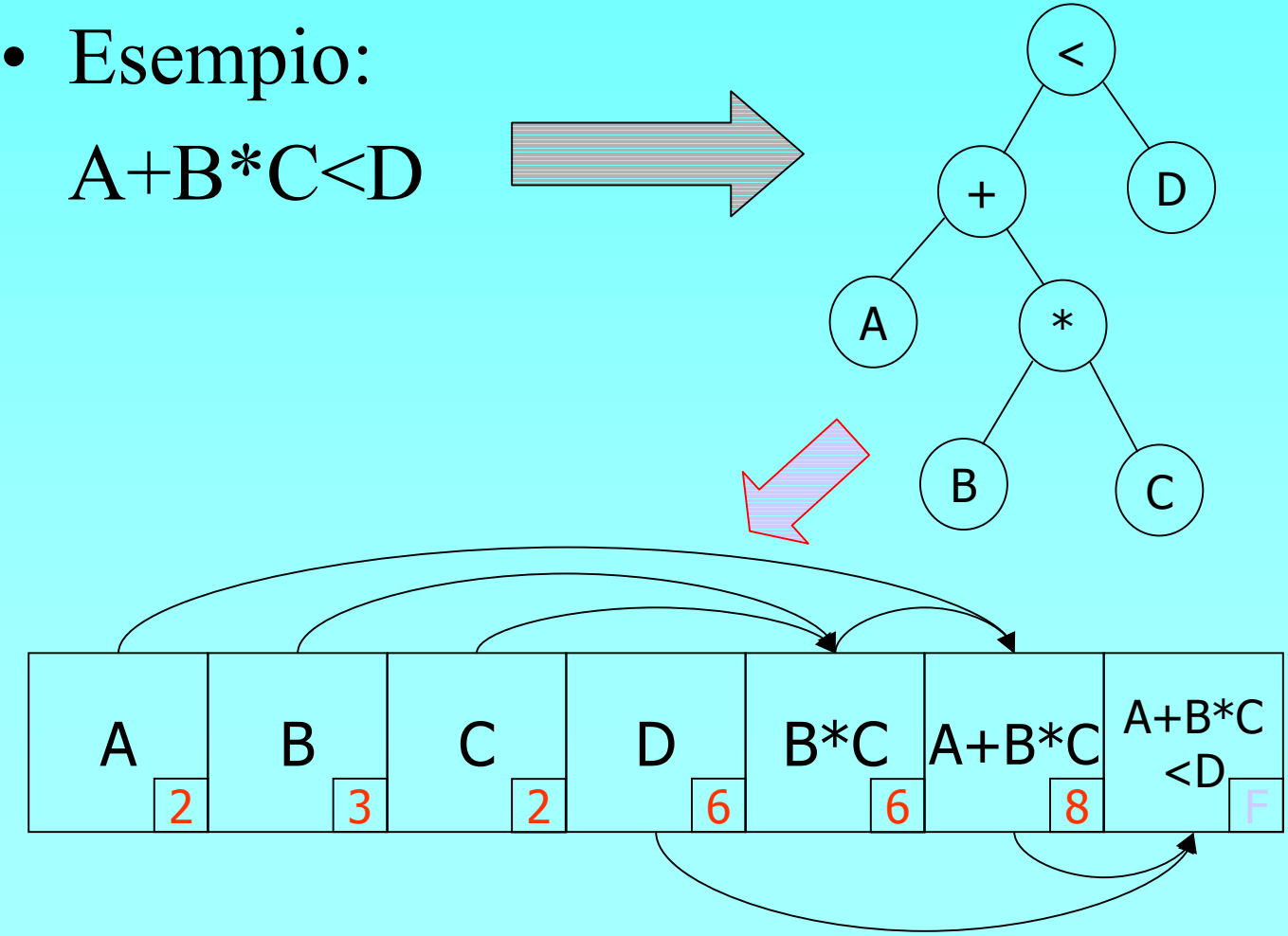


- Estensioni per gestire:
  - quantità numeriche
  - gestire azioni a durata non nulla

Espressioni numeriche: rappresentazione compatta

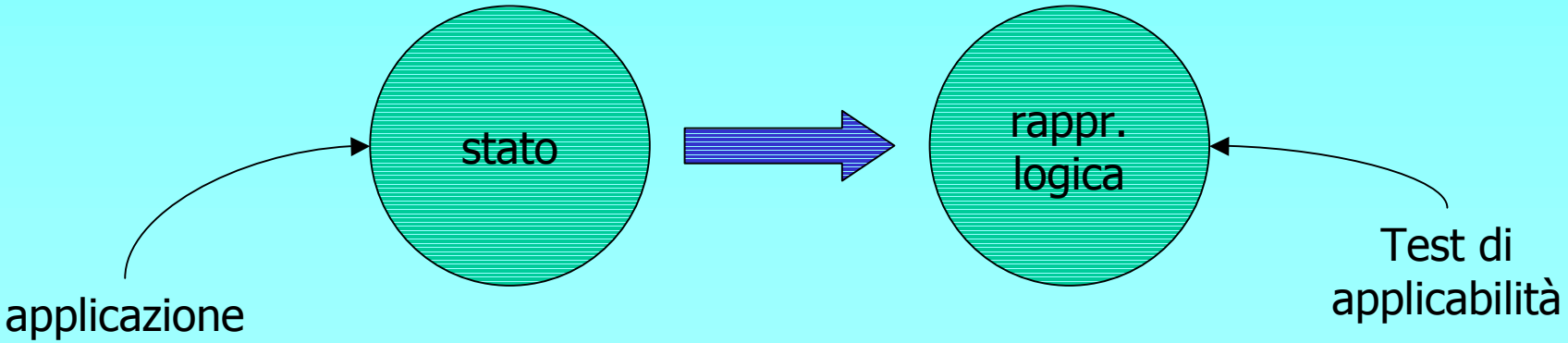
- Esempio:

  A+B*C<D



Strato numerico nascosto

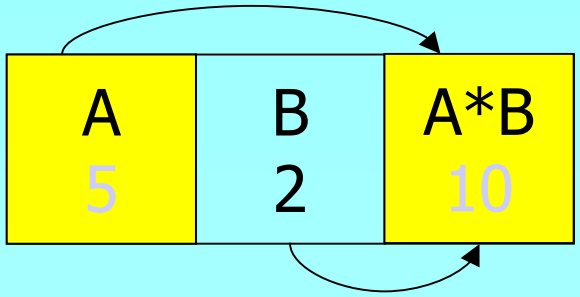
- Mantenendo aggiornato il valore di verità di tutti i confronti, ci si può ricondurre ad una rappresentazione di tipo logico, più efficiente



Riutilizzo delle tecniche di ricerca tradizionali

Propagazione dei valori: introduzione

- Ad ogni livello è associato il vettore dei valori delle espressioni numeriche
- Esso deve essere aggiornato in base all'applicazione delle azioni, mantenendo la coerenza tra i valori presenti



Propagazione dei valori

- Nella implementazione corrente si considera l'insieme Φ delle variabili modificate.
- Se l'azione *a* del livello *l* non agisce sulla variabile numerica *v* si ricopia il valore di *v* nel livello $l+1$
- Altrimenti si aggiungono in Φ le variabili *K* influenzate da *v,* presenti negli effetti di *a* e si ricalcola il valore di queste variabili

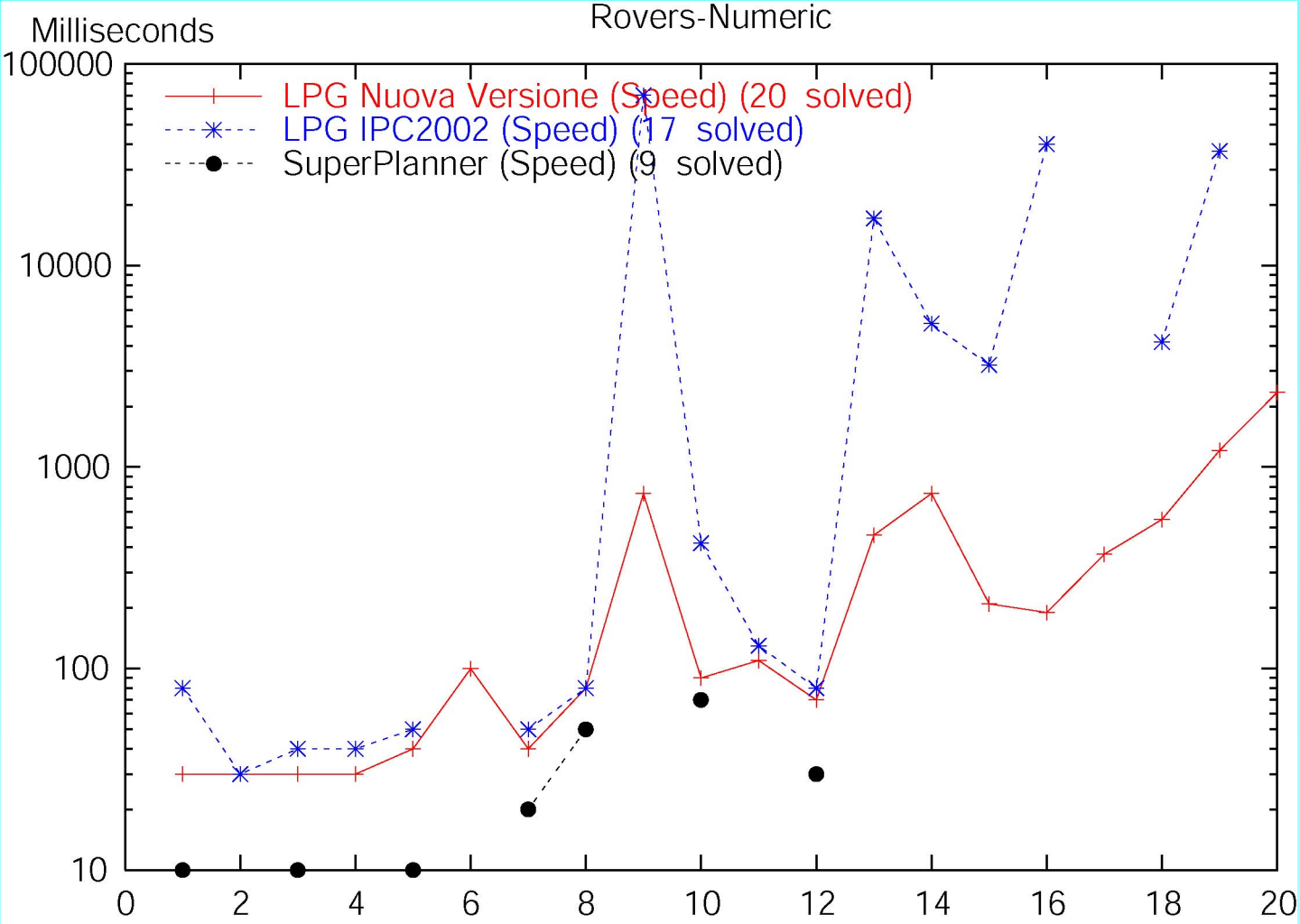
Propagazione dei valori: conseguenze

- Applicazione selettiva delle azioni
- Applicazione selettiva degli effetti
- Copia selettiva delle variabili modificate

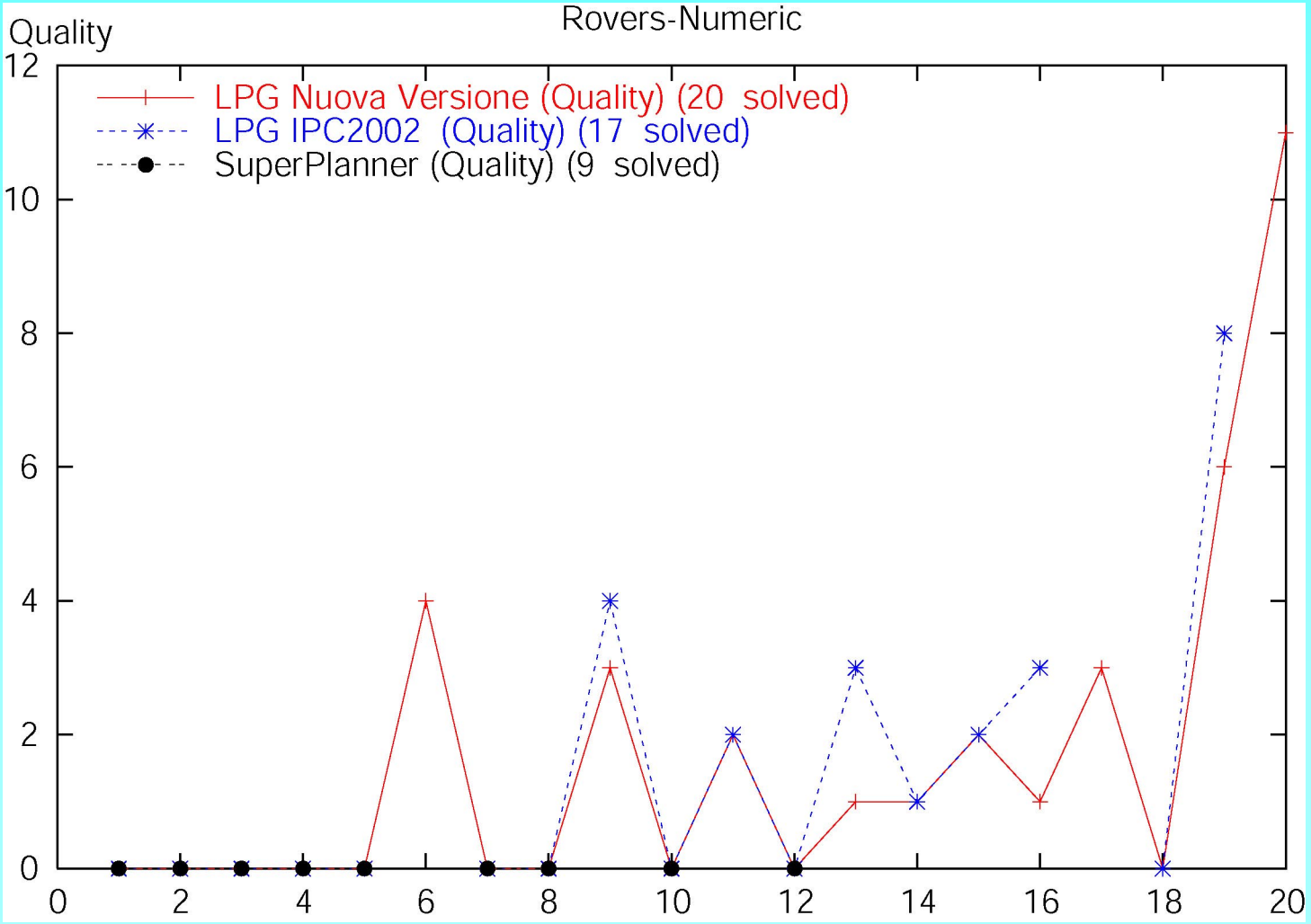
Propagazione dei valori

- Ad ogni istante temporale è associato il vettore dei valori delle espressioni numeriche
- Esso deve essere aggiornato in base all'applicazione delle azioni
- Sono state realizzate funzioni di aggiornamento efficiente che minimizzano il numero di azioni da rivalutare: se un'azione del livello n+1 lavora su valori che non sono stati modificati, essa non viene riapplicata

Risultati Sperimentali



Risultati Sperimentali



Informazioni Raggiungibilità

- Per ogni fatto f viene calcolato un insieme di informazioni di raggiungibilità
  - Numero di azioni per la raggiungibilità
  - Costo di raggiungibilità (es. risorse richieste, tempo)
- Calcolate costruendo un piano rilassato
- Informazioni di raggiungibilità estese per vincoli numerici
- Utilizzate durante la valutazione euristica del vicinato di ricerca

ComputeReachabilityInformation$(I, \mathcal{O})$

Input: The initial state of the planning problem under consideration (I) and all ground instances of the operators $(\mathcal{O})$;

Output: An estimate of the number of actions (Num_acts) and of the earliest time $(Time\_fact)$ required to achieve each fact from I.

1. **forall** facts f /* the set of all facts is precomputed by the operator instantiation phase */
2. **if** $f \in I$ **then**
3. $Num\_acts(f, 1) \leftarrow 0$; $Time\_fact(f, 1) \leftarrow 0$; $Action(f, 1) \leftarrow a_{start}$;
4. **else** $Num\_acts(f, 1) \leftarrow -1$;
5. $F \leftarrow I$; $F_{new} \leftarrow I$; $A \leftarrow \mathcal{O}$;
6. **while** $F_{new} \neq \emptyset$
7. $F \leftarrow F \cup F_{new}$; $F_{new} \leftarrow \emptyset$
8. **while** $A' = \{a \in A \mid Pre(a) \subseteq F\}$ is not empty
9. $a \leftarrow$ an action in A';
10. $ra \leftarrow$ RequiredActions$(I, Pre(a))$;
11. $t \leftarrow \underset{f \in Pre(a)}{MAX} Time\_fact(f, 1)$;
12. **forall** $f \in Add(a)$
13. **if** $f \notin F \cup F_{new}$ or $Time\_fact(f, 1) > (t + Duration(a))$ **then**
14. $Time\_fact(f, 1) \leftarrow t + Duration(a)$;
15. **if** $f \notin F \cup F_{new}$ or $Num\_acts(f, 1) > (ra + 1)$ **then**
16. $Num\_acts(f, 1) \leftarrow ra + 1$;
17. $Action(f, 1) \leftarrow a$;
18. $F_{new} \leftarrow F_{new} \cup Add(a) - F$;
19. $A \leftarrow A - \{a\}$;

RequiredActions(I, G)

Input: A set of facts I and a set of action preconditions G;

Output: An estimate of the minimum number of actions required to achieve all facts in G from I ($ACTS$).

1. $ACTS \leftarrow \emptyset$;
2. $G \leftarrow G - I$;
3. **while** $G \neq \emptyset$
4. $g \leftarrow$ an element of G;
5. $a \leftarrow Action(g, 1)$;
6. $ACTS \leftarrow ACTS \cup \{a\}$;
7. $G \leftarrow G \cup Pre(a) - I - \bigcup_{b \in ACTS} Add(b)$;
8. **return**$(|ACTS|)$.

Strutture dati Principali di LPG

Livello ActionGraph

```
typedef struct LEVEL
  in *prec_vect;     /* preconditions bit array */
  int *fact_vect;    /* fact bit array */
  inform * fact;     /* ptr -> fact level L */
  int *true_crit_vect;      /* bit array */
  int *false_crit_vect;     /* REDUNDANT */
  inform * noop_act;        /* pointer for all
                               noop of a level */
  int *noop_act_vect;       /* noop bit array */
  int *noop_prec_act_vect; /* noop precond bitarray */
  int level;                /* numero di livello */
  dg_inform ** dg_facts_array;/* Num_acts, Time_fact */
  inform action; /* pointer to the action of a level*/
  NumInfo * numeric;    /* Numeric info of level*/
```

Rappresentazione fatti e azioni

```
typedef struct INFORM

  node_cost cost; /* Cost of inserting or removing an action
                     node*/
  int *level; /* Pointer to the level index of the node*/
  int position; /* Index of the action/fact node */
  int w_is_used; /* TRUE if the node is in the action subgraph
   */
  int w_is_true;       /* Number of actions that make
                       supported the fact node */
  float time_f; /* Tempo del nodo fatto o azione */
  int ord_pos;  /* Action position in Ω */
  struct INFORM *action_f; /* for fact nodes: pointer to the
                              action that makes supported */
```

Connectivity Graph: Azioni

```
typedef struct _EfConn {
  int *PC;          /*Puntatore ai fatti precondizione (at_start)*/
  int num_PC;       /*Numero di precondizioni  (at_start)*/
  int *A;           /* Puntatore ad effetti additivi (at_end)*/
  int num_A;        /*Numero effetti additivi (at_end)*/
  int *D;           /* Puntatore ad effetti cancellanti (at_end)*/
  int num_D;        /*Numero effetti cancellanti (at_end)*/
  float cost;       /* costo e durata dell'azione */
  float duration; /* durata dell'azione */
  SpecialFacts *sf; /* Preco at end, over all, ed effetti at start */
  int *ft_exclusive_vect; /* Exclusive bit vector between
                             this action and facts (noop) */
  int *ef_exclusive_vect; /* Exclusive bit vector between  actions */
  int num_numeric_effects; /* numero di effetti numerici */
  DescNumEff *numeric_effects; /* effetto numerici */
  Bool is_numeric; /* Dice se l'azione ha effetti numerici*/
```

Connectivity Graph: Fatti

```
typedef struct _FtConn

  int *PC;  /* Azioni di cui il fatto è precondizione*/
  int num_PC; /* Numero di azioni di cui il fatto è
                 precondizione*/
  int *A; /* Azioni di cui il fatto è effetto additivo*/
  int num_A; /* Numero di azioni di cui il fatto è
                effetto additivo*/
  int *D; /* Azioni di cui il fatto è effetto
             cancellante*/
  int num_D; /* Numero di azioni di cui il fatto è
                effetto cancellante*/
  int *ft_exclusive_vect;  /* Exclusive bit vector between
                              this action and facts (noop) */
  int *ef_exclusive_vect;  /* Exclusive bit vector between
                              actions */
```

Mutue esclusioni

```
extern int **FT_FT_mutex; /*Matrice mutex fatti-fatti*/
extern int **FT_EF_mutex; /*Matrice mutex fatti-azioni*/
extern int **EF_EF_mutex; /*Matrice mutex azioni-fatti*/
extern int **EF_FT_mutex; /*Matrice mutex azioni-azioni*/
extern char **mat_ord;  /* Matrice Ω*/
extern inform_list *act_ord_vect; /*Corrispondeza tra Ω e i
                                    nodi azione*/
```