UNIVERSITÀ DEGLI STUDI DI BRESCIA
FACOLTÀ DI INGEGNERIA
DIPARTIMENTO DI ELETTRONICA PER L'AUTOMAZIONE

Sviluppo e Sperimentazione di Algoritmi per la Pianificazione Automatica in Intelligenza Artificiale: Adattamento di Piani Precostituiti attraverso il sistema ADJ

Dr. Ivan Serina
serina@ing.unibs.it

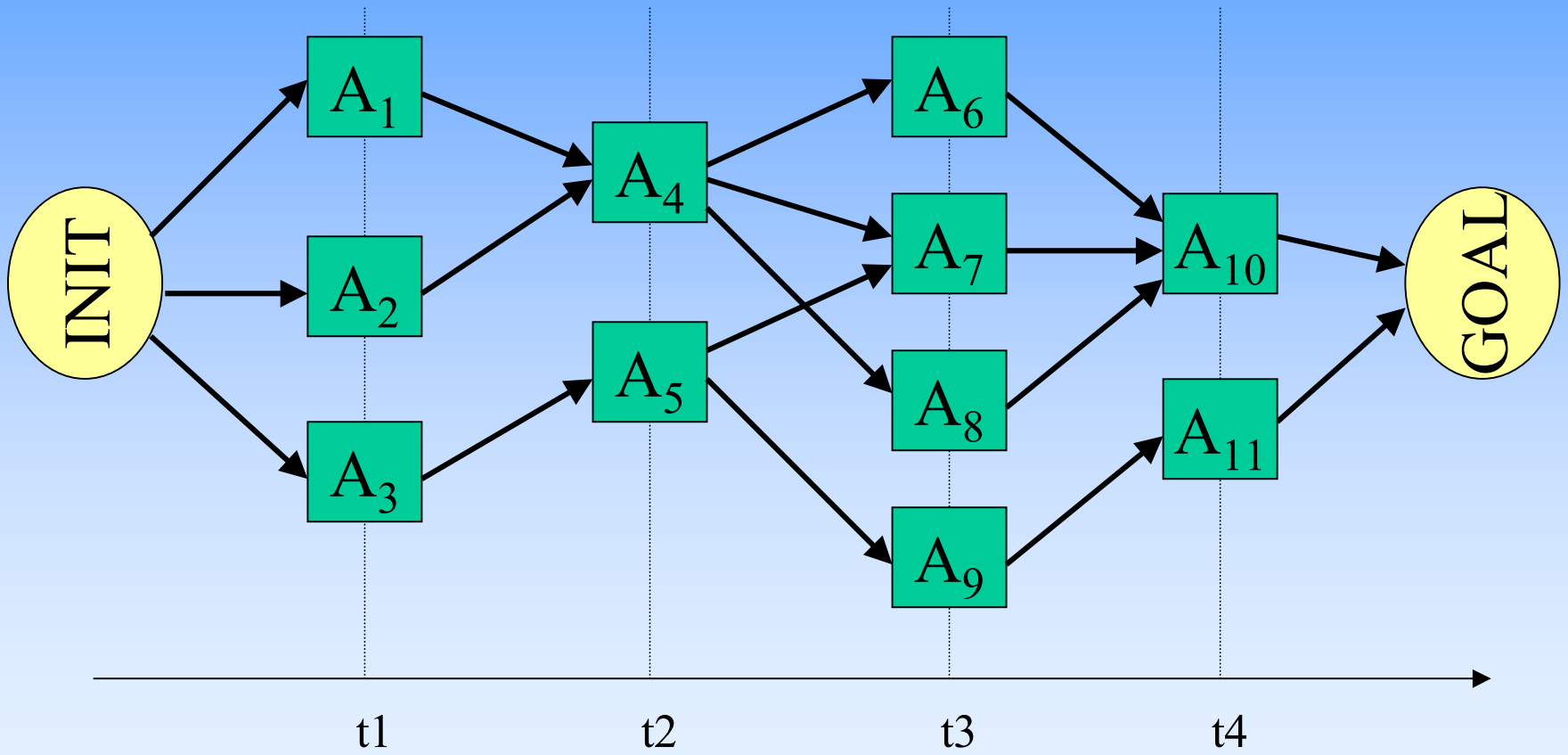
Indice della Presentazione

- Il Concetto di Piano ed Adattamento di Piani
- Descrizione del Sistema ADJ
- Risultati Sperimentali
- “Applicazioni” di ADJ
  - Simulatore per l’esecuzione-adattamento “real-time” di piani (in presenza di rumore casuale)
  - Il progetto Parsifal

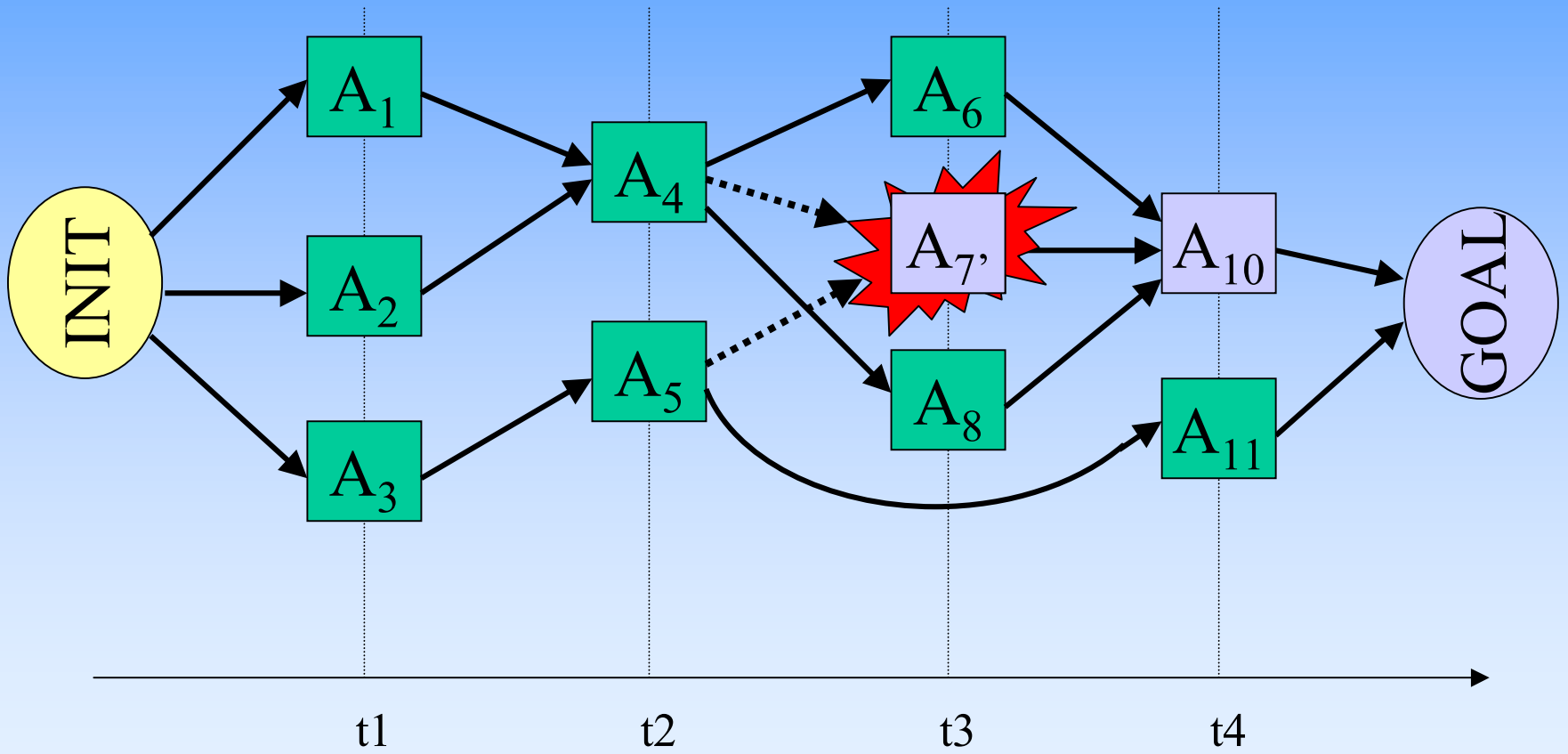
Il Concetto di Piano

- Si può esprimere con **P=(L,I,G,O)**
  - **L**, linguaggio che descrive un dominio **D** di pianificazione (STRIPS/PDDL)
  - **I**, stato iniziale
  - **G**, stato finale (o goal)
  - **O**, operatori applicabili nel dominio **D**
- Un piano è un insieme di azioni che soddisfano dei vincoli di ordinamento del tipo: $A_i < A_j$ e, quando eseguite, trasformano **I** in **G**

Un Piano valido



Un Piano non valido



Adattamento di Piani

Modificare un piano precedentemente generato π_0 per risolvere un nuovo problema.

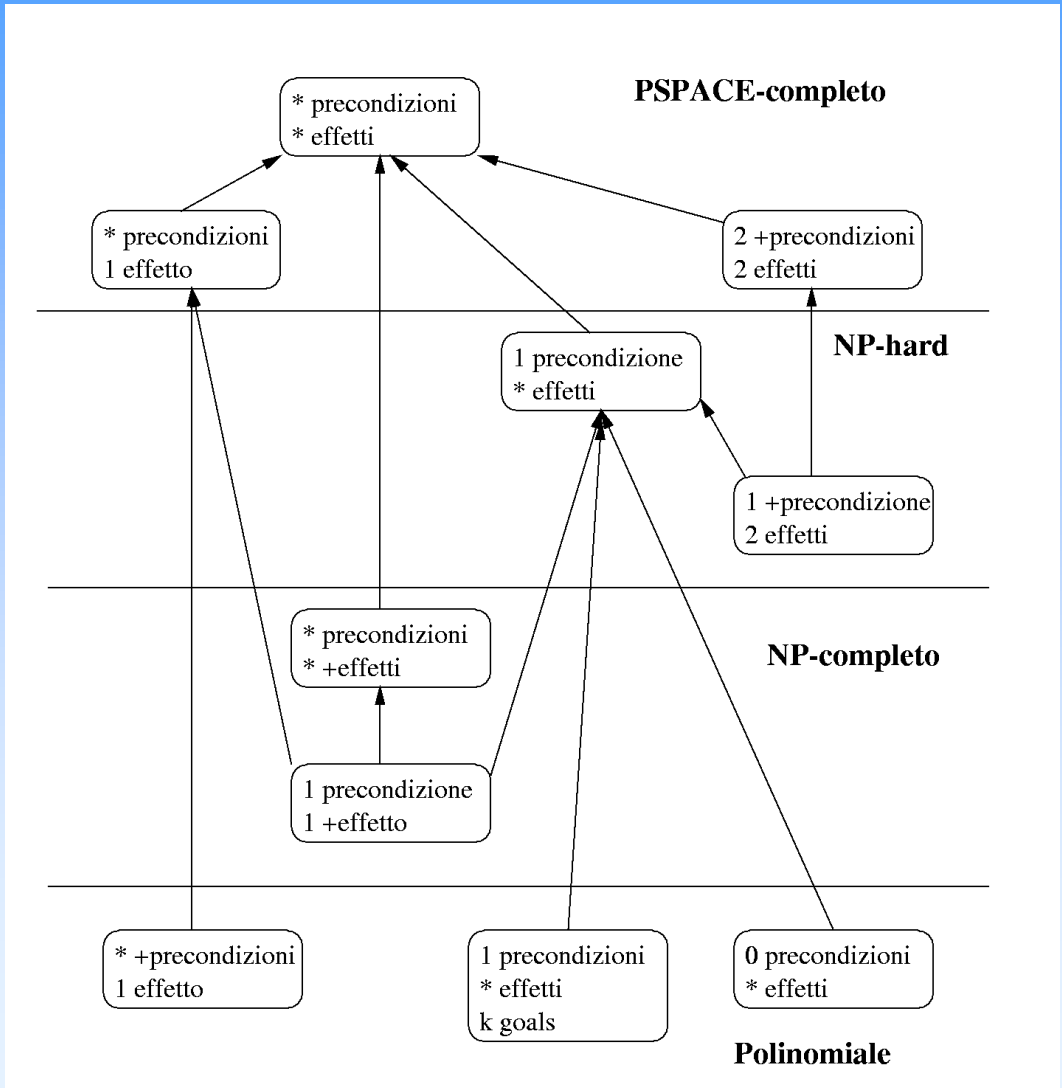
Input: $\boldsymbol{I} + \boldsymbol{G} + \boldsymbol{O} + \pi_0$

Assunzioni

- Tempo Discreto
- Effetti Deterministici
- Unica Causa di Cambiamento

Spazio di Ricerca Esponenziale rispetto ai parametri di Ingresso (PSPACE-Completo)

Complessità computazionale (generazione)



Complessità computazionale (adattamento)

- Se la generazione è PSPACE-completa o NP-hard, il problema di adattamento ottenuto modificando 1 solo goal è PSPACE-completo o NP-hard
- Anche se la generazione (sotto opportune ipotesi) è Polinomiale, il problema di adattamento (sotto il vincolo di riutilizzare almeno k azioni) è NP-Completo

Questi risultati valgono nel caso peggiore ….

Pianificatori

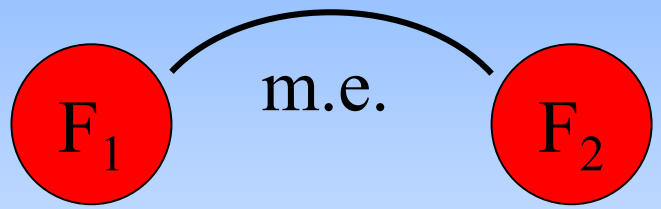
- STRIPS (SRI International '71)
- PRODIGY (Canergie Mellon University '89)
- UCPOP (Washington University '92)

- GRAPHPLAN - IPP
(Canergie Mellon University '95; Freiburg University '97)
- BLACKBOX (AT&T Labs '98)
- FF (Freiburg University 2000)
- **LPG** (Università di Brescia '02)

Adattamento di Piani con ADJ

Relazioni Mutex

- Due fatti F_1 e F_2 risultano mutuamente esclusivi se non possono essere contemporaneamente veri



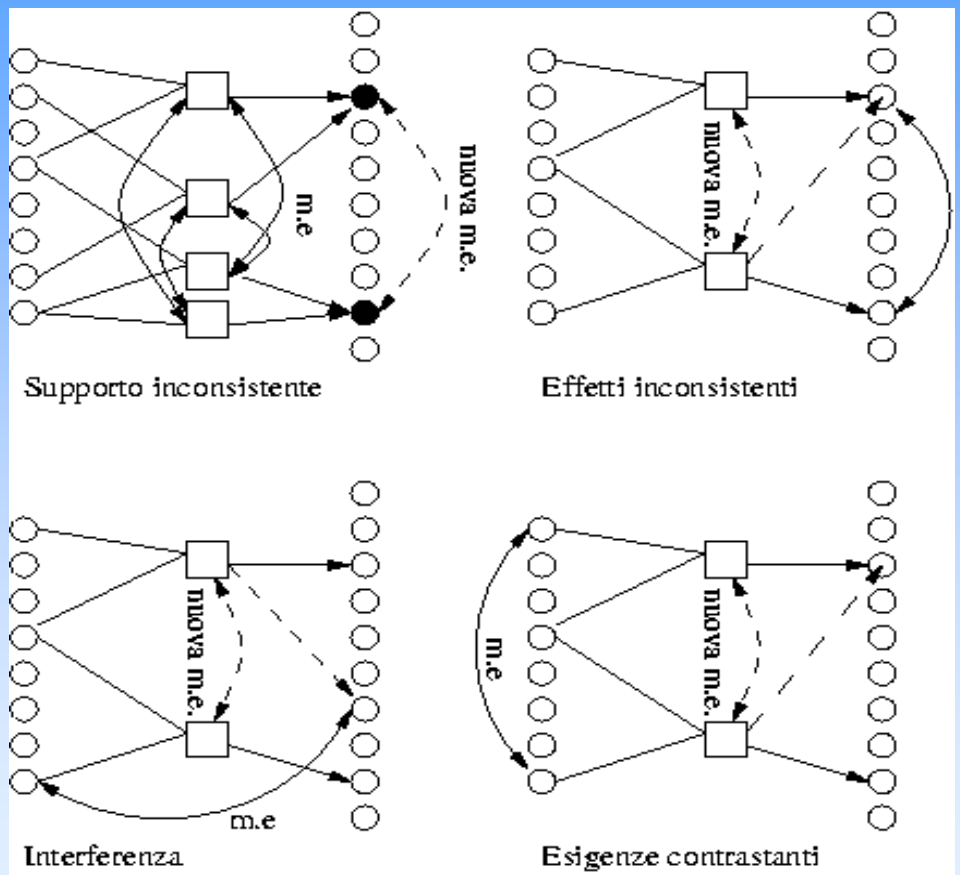
- ADJ sfrutta due metodi per calcolare automaticamente le Relazioni Mutex:
  - Metodo del “Fixpoint”
  - Metodo degli Invarianti (DISCOPLAN)

Relazioni Mutex (2)

- Costruzione dei **Mutex Pairs**
  - Tutte le coppie di fatti mutuamente esclusivi sono generate durante la fase di istanziazione degli operatori (fino al livello in cui il grafo si stabilizza)
  - Vanno **calcolati di volta in volta** per ogni variante del problema considerato

- Metodo degli **Invarianti**
  - Espressi nella logica del primo ordine
  - Calcolati da DISCOPLAN [Gerevini & Schubert, AAAI-98,00] **una sola volta per ogni problema**
  - Stessi risultati del metodo precedente

Relazioni Mutex (3)



Finestre di Ripianificazione (1)

- Una volta identificata un'inconsistenza occorre definire una finestra di ripianificazione **F(i,j)** ove:
  - **i**=livello dell'inconsistenza
  - **j**=livello dei sub-goal
- Inizialmente viene posto j=i+1
- Se l'algoritmo di ricerca non trova una soluzione si **espande la finestra** di ripianificazione, alternativamente:
  - in "avanti" (j=j+1) e
  - in "indietro" (i=i-1)

Finestre di Ripianificazione (2)

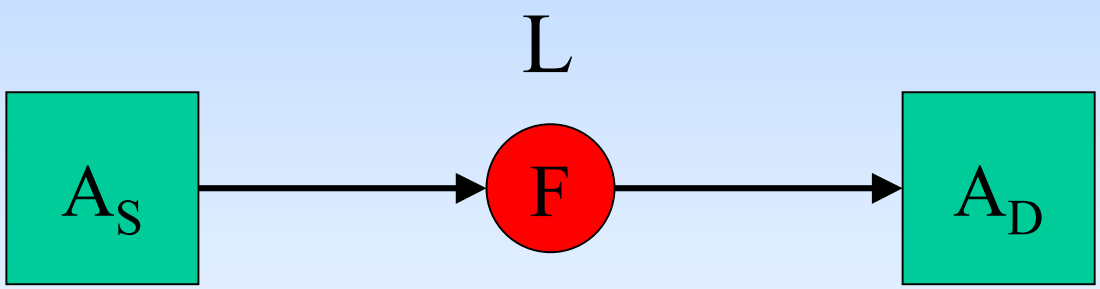
- Occorre imporre dei **limiti alla fase di ricerca** della soluzione. Per ciascuna finestra:
  - Il limite è esprimibile in **Azioni/Livelli/Tempo di CPU**
  - Il limite inizialmente imposto dipende dal **numero delle inconsistenze** presenti nella finestra originale
  - Se non viene trovata una soluzione si espande la finestra e si incrementa il limite
  - Il fattore di espansione dipende dall'algoritmo di ricerca

- Imporre dei limiti consente di ottenere dei piani adattati in tempi brevi e con un numero di livelli contenuto.

Link Causali

Un link causale **L** si esprime con $A_S \xrightarrow{F} A_D$ ove:

- **F** è un fatto
- A_S è l'azione "sorgente" (F = effetto additivo)
- A_D è l'azione "destinazione" (F = precondizione)



Link Causali

- Vengono costruiti in modo **backward** a partire dal livello dei goal basandosi sul piano attuale
- Vanno aggiornati ogni volta che viene trovata una soluzione per una finestra di ripianificazione
- Le azioni che non partecipano a nessun link causale si possono rimuovere dal piano (**irrelevant actions**).

Stima Euristica dei Sotto-Goal

- Sono state ideate ed implementate 3 euristiche:
  - Refined Ω-Goal Set
  - Causal Links
  - Causal Links + Goal Reordering
- Tutte queste euristiche evitano di inserire tra i sotto-goal da realizzare fatti mutuamente esclusivi
- Due fatti F_1 e F_2 risultano mutuamente esclusivi se non possono essere contemporaneamente veri

Stima dei Sotto-Goal (1)

- **Euristica "Refined - Ω Goal Set"**
- Un nuovo fatto viene inserito nell'insieme dei sotto-goal da realizzare se:
  - è una precondizione di qualche azione successiva
  - non sarà aggiunto (affetto additivo) o cancellato (effetto cancellante) da qualche azione successiva
  - non è mutuamente esclusivo con gli altri sotto-goal

- All'ultimo livello tutti i goal del problema vengono marcati come precondizioni di una azione fittizia a_{end} successiva all'ultimo livello

Stima dei Sotto-Goal (2)

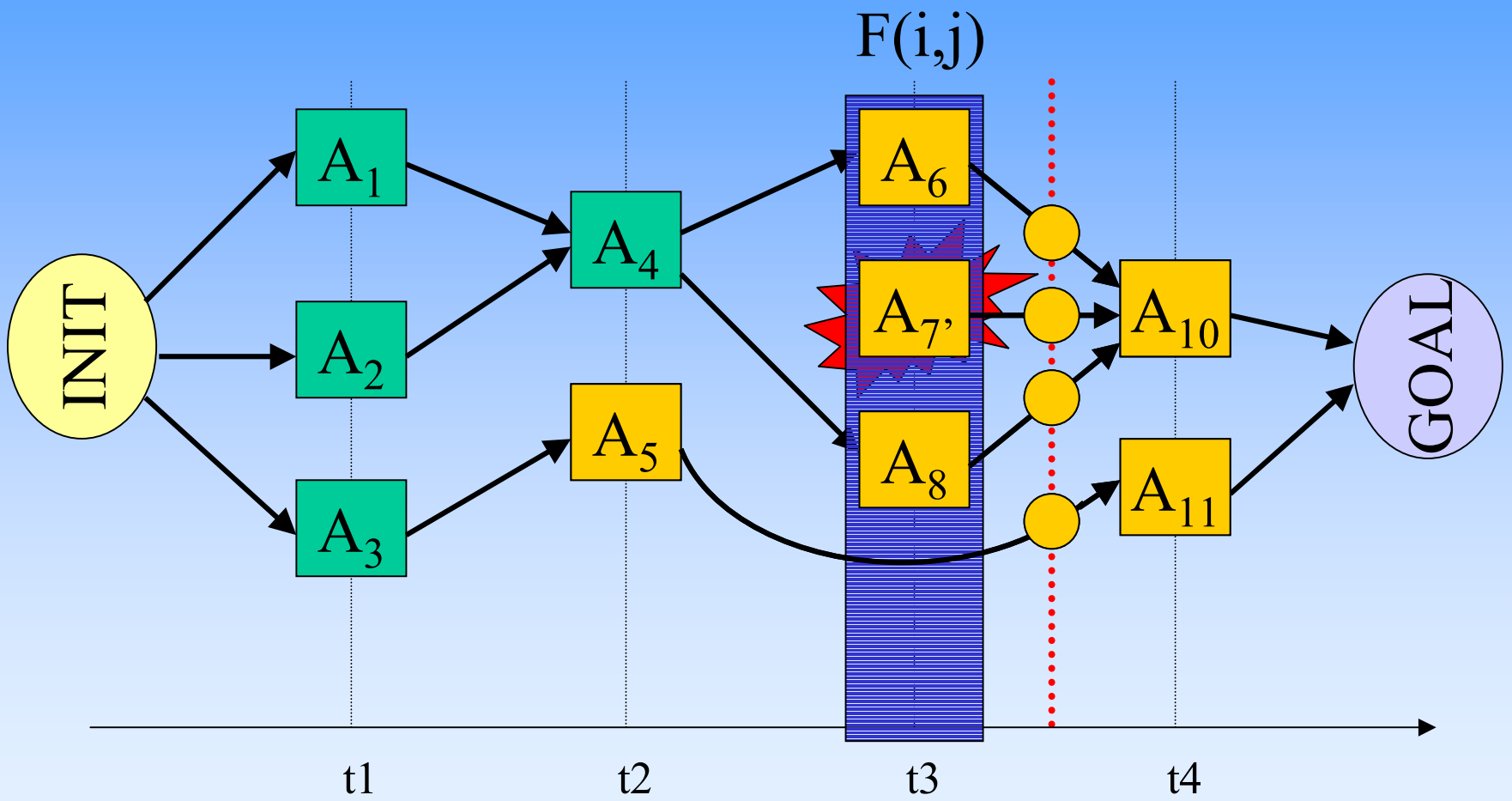
- **Euristica dei "Link Causali"**
- I link causali ottenuti non sono quelli effettivi del piano soluzione del problema, ma solo una stima euristica
- Successivamente si aggiunge un fatto all'insieme dei sotto-goal da realizzare se:
  - esiste un link causale (al livello attuale dei goal) ed il fatto non è mutuamente esclusivo con gli altri fatti goal;

- In caso di mutua esclusione?
  - Si può scegliere quale fatto inserire tra i sotto-goal

Stima dei Sotto-Goal (3)

- Si inserisce il nuovo fatto tra i goal e si elimina il precedente, se:
  - il nuovo fatto è coinvolto in **più link causali** di quelli ai quali partecipa il goal con esso mutuamente esclusivo
  - se i due fatti posseggono lo stesso numero di link causali si è scelto di privilegiare:
    - quello che partecipa al **link causale più prossimo** (cioè più vicino nel tempo)
    - quindi l'euristica sfrutta la **"prossimità temporale"**
  - se i due fatti posseggono lo stesso numero di link causali e coincidono anche i link più prossimi viene scelto uno casualmente

L'euristica "Causal Links"



Ordinamento dei Goal (1)

- In alcuni domini (come Blocksworld) esiste un “naturale” **ordinamento dei goal**; anche in fase di ripianificazione risulta molto utile sfruttare tale ordinamento
- La Lista Consistente H
  - Contiene tutti i **goal del nuovo problema ordinati tra loro** nel rispetto della goal-agenda GA (relazioni mutex)
  - Tale **ordinamento** viene poi **raffinato con il piano di ingresso** (l’ordinamento risultante è quello che comporta i minimi cambiamenti del piano da adattare)
  - $H=\cup H(i)$ è indicizzata per livelli
  - H(i) contiene l’insieme dei goal che vanno realizzati al livello i

Ordinamento dei Goal (2)

- La lista consistente $H=\cup H(i)$ contiene inizialmente l'ordinamento estratto dal piano di ingresso
- Viene scelta una linearizzazione della GA
- $\forall\ g_i \in$ GA, si esamina H(j) (con $0 \leq j \leq$ *endtime*)
  - Se $g_j \in H(j)$ e $g_i <_{GA} g_j$
    - Si eliminano dal piano le azioni che realizzano unicamente g_i
    - Si definisce k, nuovo indice per g_i in H
    - Si inserisce/riposiziona g_i nella lista H(k)
    - Si posticipano di un time-step tutte le azioni dal piano a partire dal livello k e si aggiornano gli indici dei goal $g_l \in H$ con $l>k$
  - Se $g_i \in H(j)$, break (il goal è già ordinato)

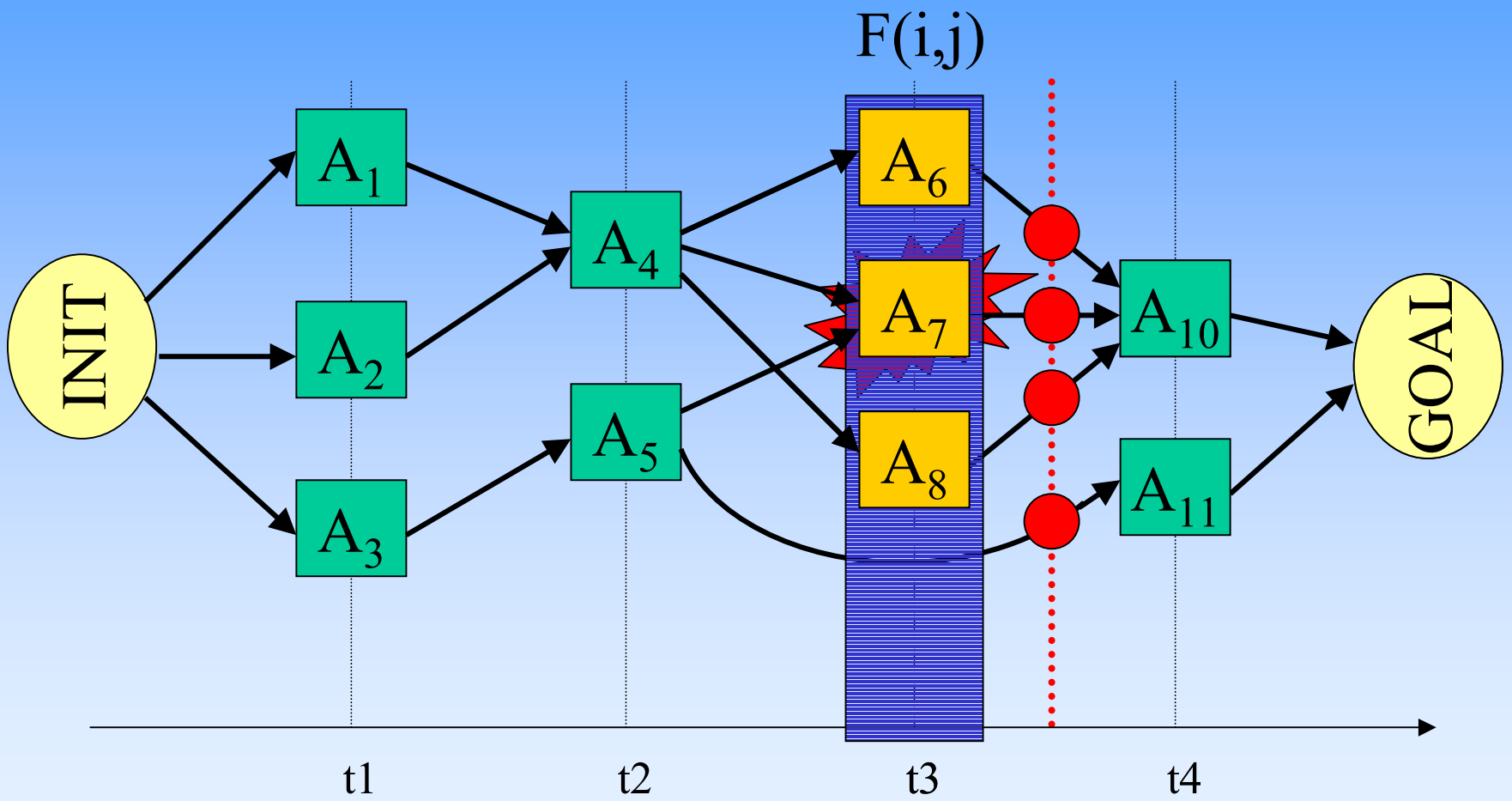
Stima dei Sotto-Goal (4)

- **Euristica "Link Causali + Goal Agenda"**
- Migliora l'euristica precedente, sfruttando le eventuali informazioni di ordinamento tra i goal
- Sia $\mathbf{G_{ORD(j)}}$ l'insieme di tutti i goals per cui esiste un ordinamento rispetto a π prima del timestep j, ove $G_{ORD(j)} = \cup_{i \leq j} H(i)$
- Sia **LC(j)** = $\{f \mid a_i, a_k \in \pi$ con $i \leq j \wedge k \leq j$ tale che $a_i \rightarrow a_j$ in $f\}$ l'insieme dei fatti per cui è stato calcolato un link causale avente come sorgente una azione prima di j e come destinazione una azione dopo j.
- Sia inoltre **A(j)** = $\{g \in G \mid \exists\, g_1 <_{GA} g \notin G_{ORD(j)}\}$ l'insieme dei goal che dovranno essere realizzati dopo l'istante temporale j.
- I goal da realizzare all'istante temporale j sono $\mathbf{G=G_{ORD(j)} + LC(j) - A(j)}$

Ripianificazione sulla Finestra F(i,j)

- E' possibile utilizzare:
  - Un **pianificatore esterno** che colloquia con ADJ tramite file di Input/Output
    - IPP v4.1 [Koehler-et-al 1996]
    - FF v2.3 [Hoffman 2000]
    - Altri pianificatori…
  - L'**algoritmo di ricerca locale** LPG è stato recentemente integrato in ADJ ed è attualmente in fase di testing.

Riparazione del Piano



Algoritmo ADJUST-PLAN (1)

1. Sia π il piano ottenuto da π_0 **rimuovendo le azioni** che non sono più applicabili per risolvere Π
2. Viene determinato l'**ordinamento dei goal** H
3. Viene identificato il più basso **timestep i'** in π in cui si ha un'**inconsistenza**: se tale timestep non esiste, return π
4. Viene identificato il più basso timestep i'' contenente un goal g non ancora realizzato ed ordinato in i'' (cioè $g \in H(i'')$)
5. Si pone **init-time**=i e **goal-time**= i+1 con i=min(i',i'')
6. Se i è l'ultimo timestep di π, allora init-time=i-1 e goal-time=i

Algoritmo ADJUST-PLAN (2)

7. While CPU-time ≤ max-adjust-time

(a) **Ripianifica** usando come fatti iniziali **F(init-time)** e come goal **G(goal-time)**, dove il primo è il set dei fatti che veri all'istante init-time e il secondo è l'insieme dei sotto-goal per π stimati euristicamente al tempo goal-time

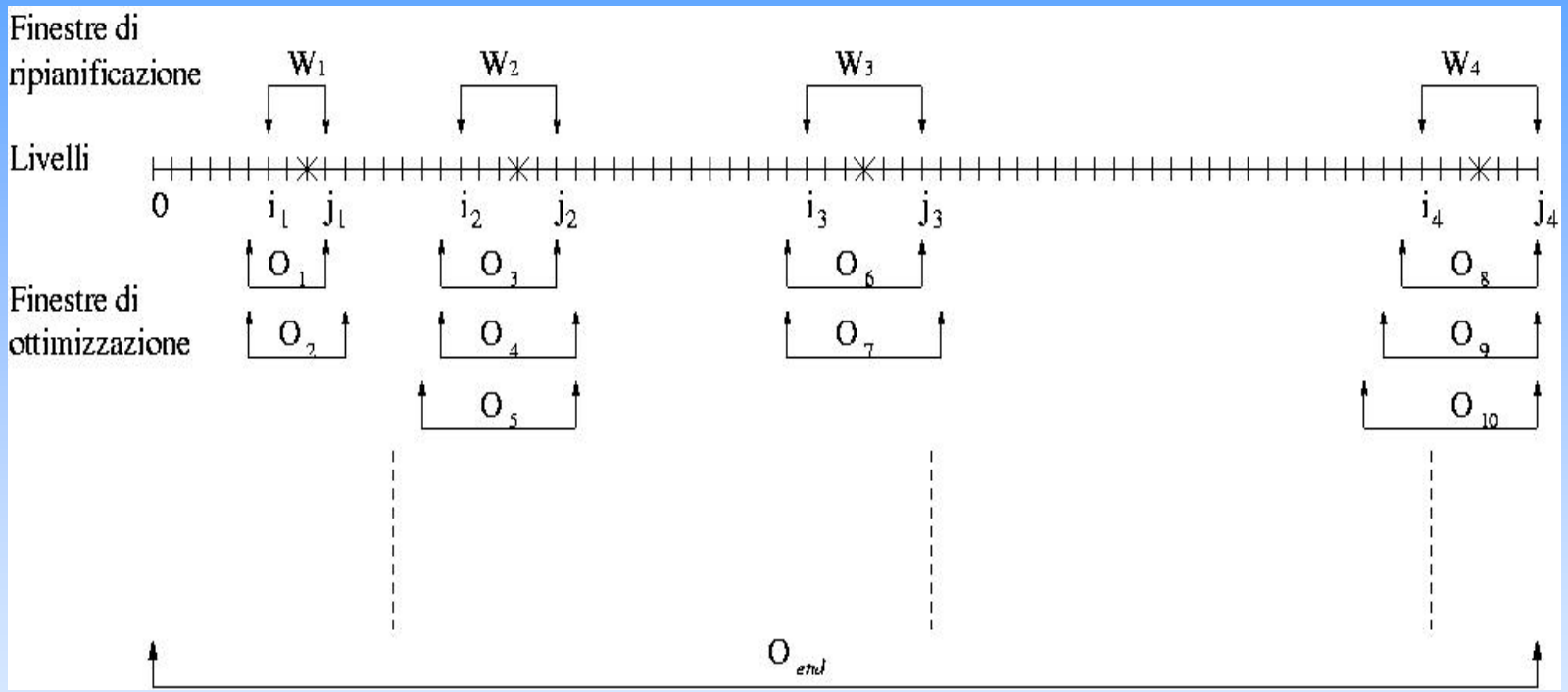
(b) Se non esiste nessun piano da F(init-time) a G(goal-time), oppure si è superato il limite di ricerca imposto, si espande la finestra; altrimenti le azioni esistenti tra init-time e goal-time in π vanno rimpiazzate con il nuovo sottopiano, e goto 3.

8. Return FAIL

Ottimizzazione Iterativa

- Viene prodotta un successione di piani migliorati ognuno dei quali ha un numero di livelli minore del precedente.
- Tali piani sono ottenuti considerando delle finestre di pianificazione (problemi di ripianificazione) *via via più grandi*.
- Se la soluzione di un nuovo problema di ripianificazione comprende un numero di livelli inferiore alla corrispondente finestra di ripianificazione allora *sostituiamo* le azioni della finestra con quelle del sottopiano.

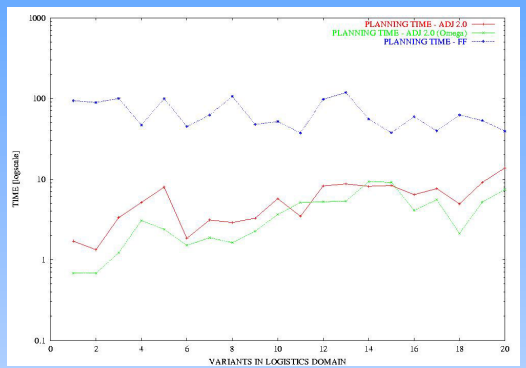
Fase di Ottimizzazione

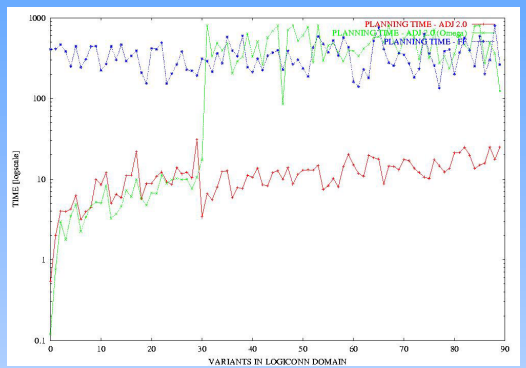


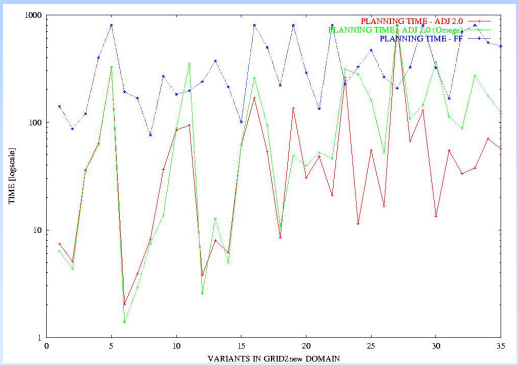
Risultati Sperimentali

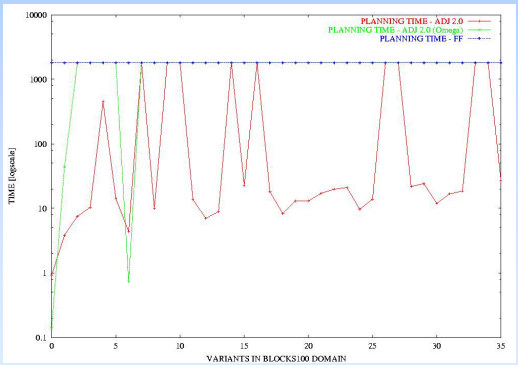
- Si sono utilizzate numerose **varianti dei problemi** utilizzati nelle competizioni AIPS98 e AIPS2000 (generate in maniera casuale)
- ADJ adatta tali varianti a partire dal **piano soluzione del problema originale**
- L'adattamento risulta fino a **2 ordini di grandezza più veloce** della generazione, a scapito di un maggior numero di livelli/azioni nel piano risultante
- Per confronto sono riportati i tempi richiesti da FF, vincitore di AIPS2000

Risultati Sperimentali (2)

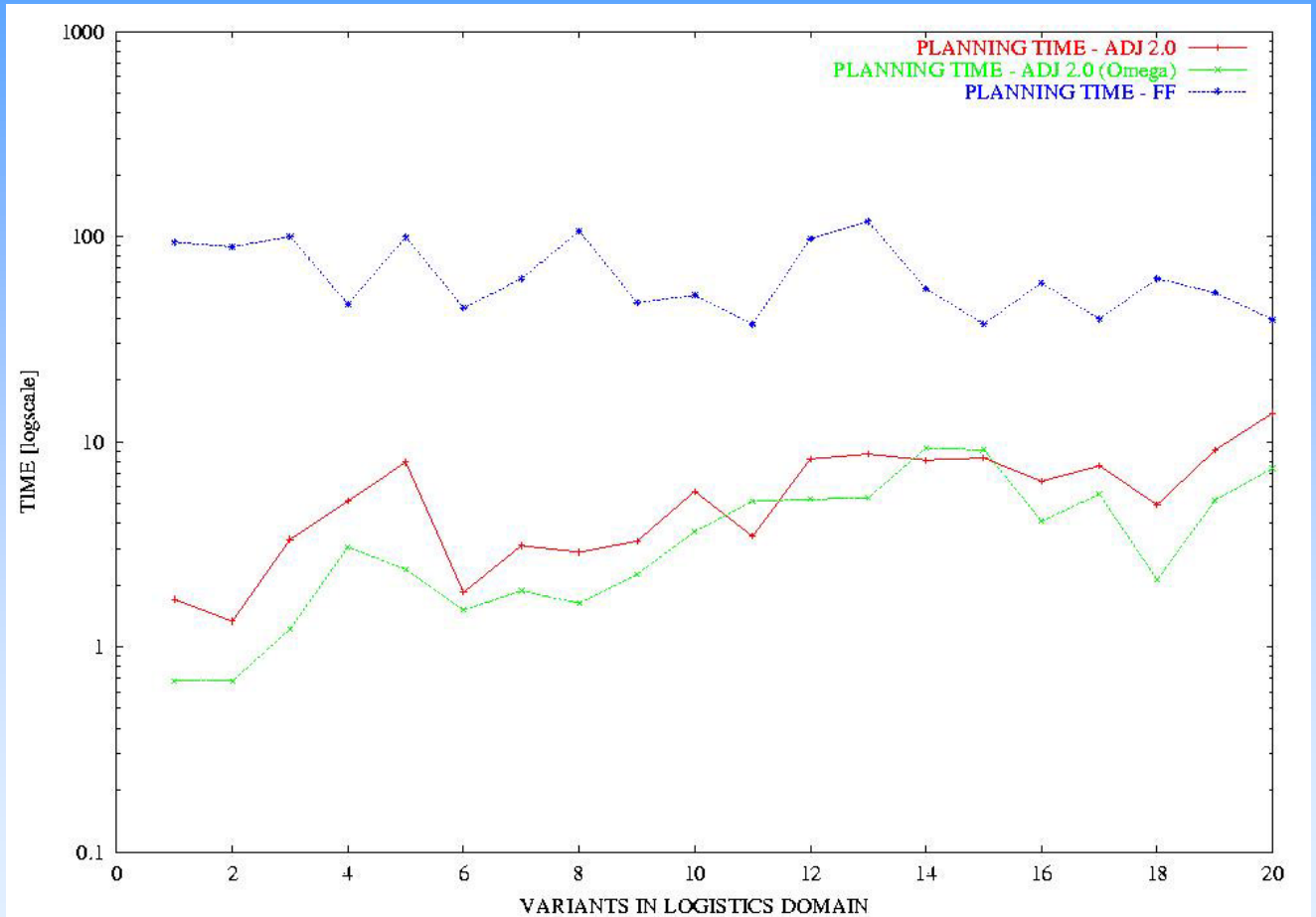






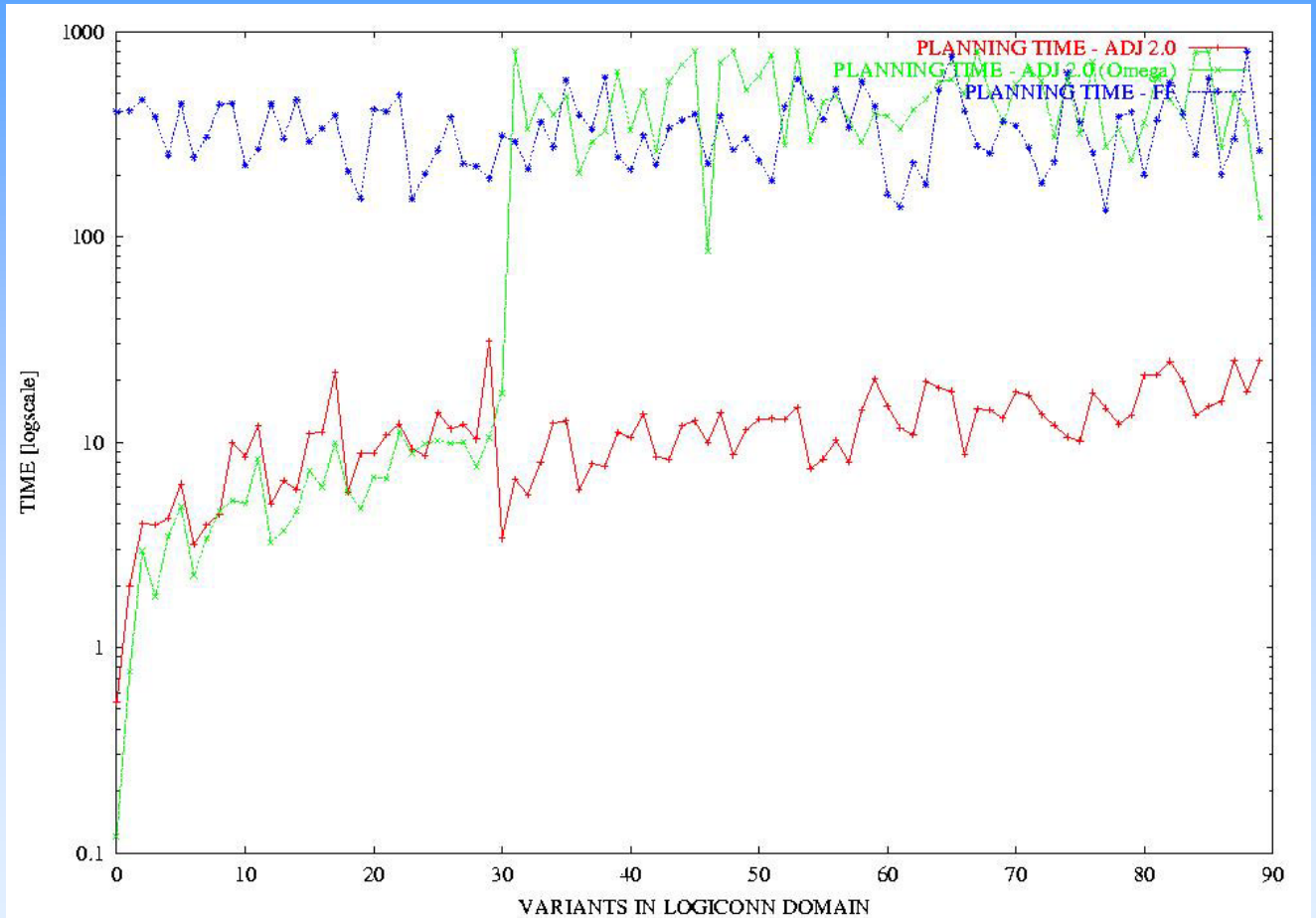


Il Dominio Logistics



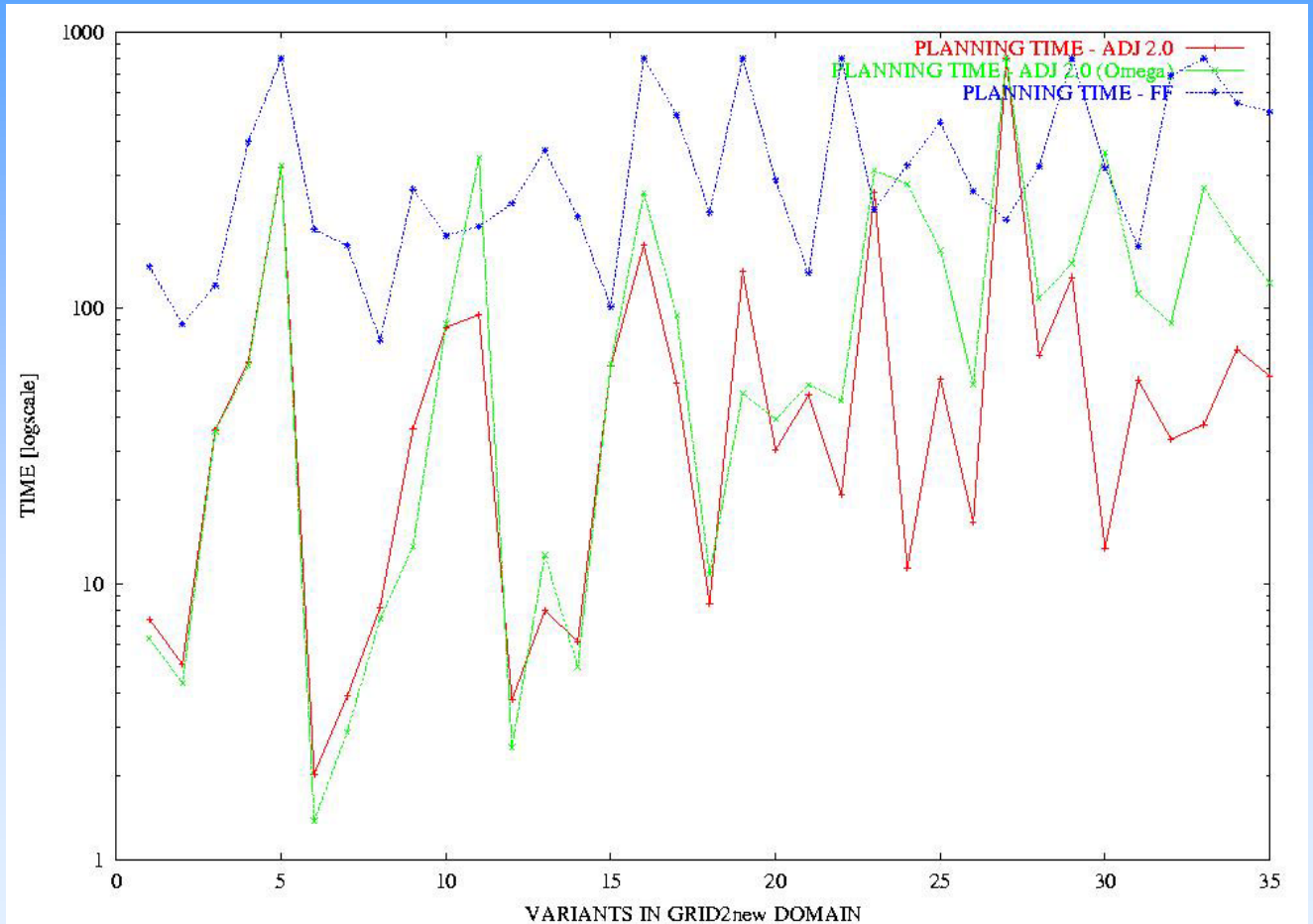
Sistema ADJ - 2000/2002

Il Dominio Logiconn



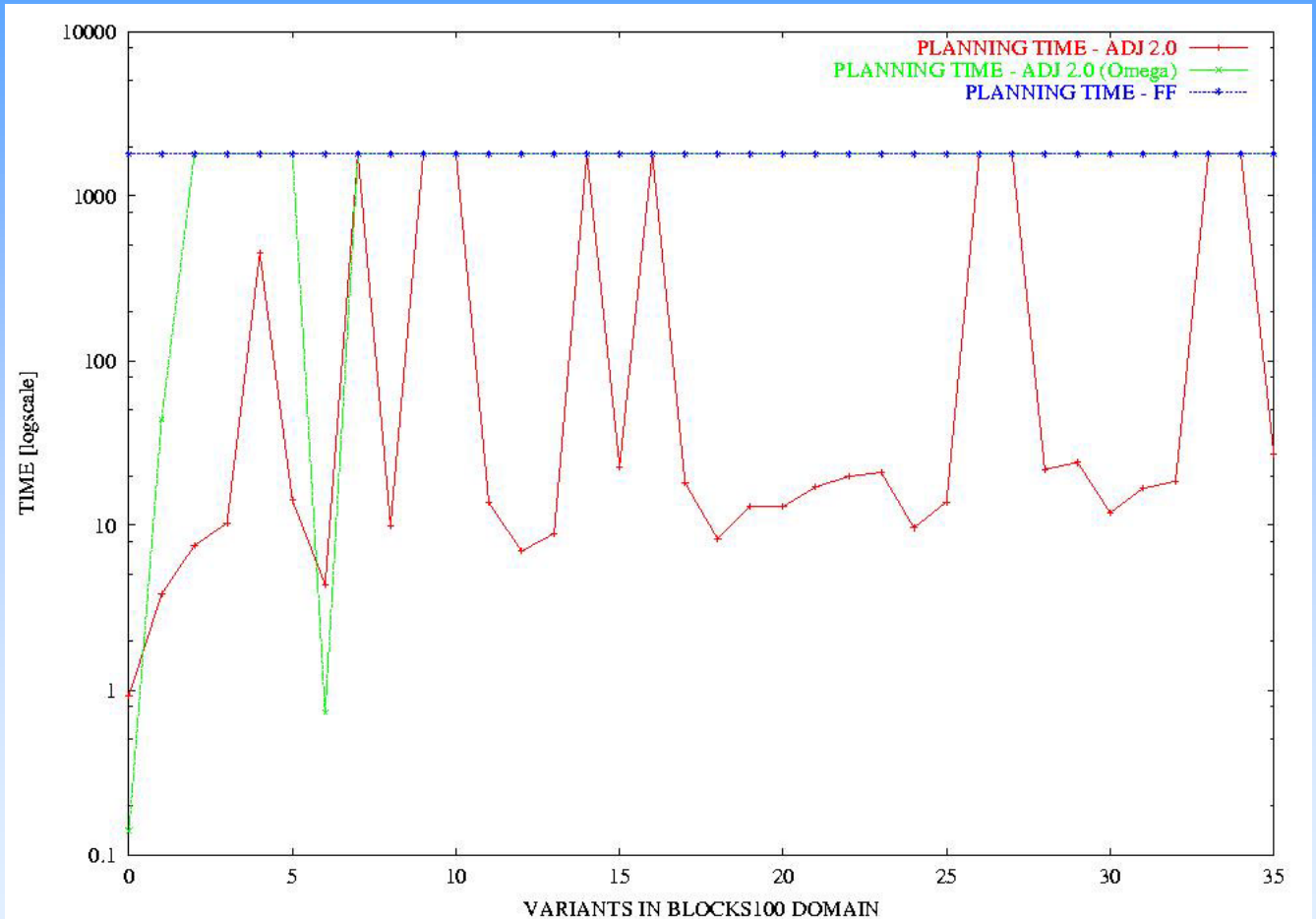
Sistema ADJ - 2000/2002

Il Dominio Grid



Sistema ADJ - 2000/2002

Il Dominio Blocksworld



Sistema ADJ - 2000/2002

Risultati Sperimentali (1)

LOGISTICS	GPG-ADJ 2.0									FF v2.3	
	T1	L1	A1	T2	L2	A2	TO	LO	AO	T	A
prob-logistics41-01	2.18	286	286	29.42	284	284	581.14	261	261	93.78	283
prob-logistics41-02	1.82	279	279	3.16	278	278	773.55	270	270	88.95	275
prob-logistics41-03	3.73	297	297	8.27	296	296	119.35	293	292	99.91	274
prob-logistics41-04	5.52	290	290	18.39	289	289	622.12	279	278	46.75	255
prob-logistics41-05	8.07	297	297	27.62	296	296	727.71	279	279	99.43	275
prob-logistics41-06	2.32	284	284	5.19	282	282	717.48	264	264	44.89	267
prob-logistics41-07	3.55	286	286	30.90	285	285	708.44	259	259	62.14	271
prob-logistics41-08	3.44	292	292	13.03	291	291	658.94	285	283	106.29	280
prob-logistics41-09	3.78	290	290	32.54	289	289	32.54	289	289	47.60	263
prob-logistics41-10	6.11	296	296	17.54	295	295	130.89	289	289	51.68	265
prob-logistics41-11	3.68	288	288	5.71	286	286	339.89	280	280	37.27	256
prob-logistics41-12	7.99	321	321	12.74	319	319	367.90	315	315	97.36	269
prob-logistics41-13	8.59	315	315	14.04	314	314	730.94	306	306	118.62	276
prob-logistics41-14	7.95	325	325	147.61	324	324	614.63	323	323	55.48	258
prob-logistics41-15	8.28	325	325	16.29	323	323	752.02	313	313	37.47	263
prob-logistics41-16	6.51	306	306	527.04	304	304	527.04	304	304	59.32	257
prob-logistics41-17	7.23	312	312	750.63	311	311	750.63	311	311	39.61	244
prob-logistics41-18	5.17	279	279	5.79	276	276	5.79	276	276	62.26	263
prob-logistics41-19	8.68	326	326	18.48	324	324	460.10	313	313	53.19	270
prob-logistics41-20	13.27	298	298	44.60	297	297	162.48	286	286	39.33	245

Risultati Sperimentali (2)

LOGISTICS2	GPG-ADJ 2.0									FF v2.3	
	T1	L1	A1	T2	L2	A2	TO	LO	AO	T	A
prob-logistics70-01	16.82	470	470	105.24	469	469	169.63	468	468	1151.17	449
prob-logistics70-02	16.90	471	471							1189.70	454
prob-logistics70-03	28.92	488	488							1445.50	461
prob-logistics70-04	28.05	488	488	79.42	487	487	79.42	487	487	1151.39	448
prob-logistics70-05	33.95	496	496							—	—
prob-logistics70-06	13.18	455	455	69.12	454	454	69.12	454	454	1149.12	455
prob-logistics70-07	12.29	455	455	75.90	454	454	75.90	454	454	—	—
prob-logistics70-08	27.62	459	459	87.42	458	458	87.42	458	458	502.31	423
prob-logistics70-09	48.97	473	473	197.98	472	472	477.82	471	471	877.96	453
prob-logistics70-10	45.56	473	473	155.35	472	472	155.35	472	472	458.05	462
prob-logistics70-11	14.10	461	461							1707.72	460
prob-logistics70-12	18.02	465	465							1269.76	466
prob-logistics70-13	30.07	476	476	60.38	475	475	1517.78	471	471	1191.12	452
prob-logistics70-14	27.59	481	481	84.23	479	479	323.67	477	477	1545.64	468
prob-logistics70-15	48.19	497	497	178.94	496	496	178.94	496	496	997.39	433

Risultati Sperimentali (3)

GRID	GPG-ADJ 2.0									FF v2.3	
	T1	L1	A1	T2	L2	A2	TO	LO	AO	T	A
prob05-01	5.37	167	167	11.24	166	166	91.66	162	162	73.22	178
prob05-02	6.95	171	171	14.91	170	170	14.91	170	170	87.56	188
prob05-03	2.03	159	159							35.73	178
prob05-04	2.32	161	161	3.06	159	159	509.62	128	128	135.47	141
prob05-05	1.44	154	154							29.32	136
prob05-06	2.52	160	160	440.99	159	159	590.91	134	134	107.06	154
prob05-07	0.33	130	130				309.54	113	112	8.87	112
prob05-08	4.45	167	167	65.52	166	165	101.55	137	137	19.49	140
prob05-09	4.35	170	170	16.25	168	168	44.88	138	138	16.90	134
prob05-10	2.05	157	157	6.49	156	156	264.66	130	130	18.35	143
prob05-11	2.82	161	161							19.76	153
prob05-12	1.38	153	153	53.89	149	149	53.89	149	149	85.02	145
prob05-13	0.95	148	148	1.32	146	146	344.74	131	130	32.89	147
prob05-14	2.05	157	157	6.58	156	156	290.93	136	136	19.33	147
prob05-15	16.11	186	186	39.81	184	183	788.31	177	176	84.03	201
prob05-16	4.59	175	175	6.14	173	173	695.26	162	162	38.02	130
prob05-17	1.37	149	149	2.72	147	147	28.36	130	130	21.10	122
prob05-18	13.88	185	185	37.68	183	182	37.68	183	182	19.13	153

Risultati Sperimentali (4)

GRID2	GPG-ADJ 2.0									FF v2.3	
	T1	L1	A1	T2	L2	A2	TO	LO	AO	T	A
prob-grid8-01	3.11	272	272	5.77	271	271	5.77	271	271	235.83	247
prob-grid8-02	9.04	298	298	9.60	296	296	274.63	268	268	210.48	267
prob-grid8-03	5.37	287	287	5.84	285	285	5.84	285	285	241.71	265
prob-grid8-04	17.59	300	300	76.19	298	298	666.63	259	259	—	—
prob-grid8-05	11.00	291	291	48.06	290	290	48.06	290	290	195.60	252
prob-grid8-06	17.36	300	300	25.85	298	298	88.20	273	273	93.48	215
prob-grid8-07	11.60	317	317	13.52	315	315	14.41	314	314	427.45	302
prob-grid8-08	9.31	309	309	9.88	307	307	23.88	305	305	206.71	259
prob-grid8-09	10.26	317	317	21.01	303	303	21.01	303	303	348.77	238
prob-grid8-10	11.00	306	306	11.68	304	304	194.05	267	266	228.56	285

Risultati Sperimentali (5)

MYSTERY	GPG-ADJ 2.0						FF v2.3		IPP v4.1		
	T1	L1	A1	T2	L2	A2	T	A	T	L	A
prob19-01	0.93	11	13	21.01	10	9	17.37	10	25.54	8	10
prob19-02	9.15	7	7				8.91	8	12.28	6	8
prob19-03	0.95	11	13	20.10	10	9	17.80	10	25.33	8	10
prob19-04							8.61	7	10.86	6	6
prob19-05	0.48	8	8	0.63	6	6	8.36	8	8.20	6	6
prob19-06	0.70	11	11	0.85	9	9	—	—	8.25	6	9
prob19-07	0.77	14	16	0.96	13	15	13.37	15	18.42	7	13
prob19-08	0.79	14	16	0.97	13	15	12.83	15	18.40	7	13
prob19-09	1.56	17	17				5.52	17	18.07	7	16
prob19-10	0.66	10	11	0.83	9	9	7.70	11	12.20	6	11
prob19-11	0.81	11	11				0.44	11	24.00	8	13
prob19-12	1.02	13	13				—	—	24.19	8	17
prob19-13	1.21	13	13				—	—	24.36	8	17
prob19-14							—	—	—	—	—
prob19-15	1.02	13	13				—	—	24.17	8	17
prob19-16	1.02	13	13				—	—	24.23	8	17
prob19-17	1.02	13	13				—	—	24.28	8	17

Combinazione di Ricerca Locale e Sistematica

Combinazione seriale

1 Esegui ricerca sistematica finché

(a) viene individuata una soluzione

(b) si dimostra che il problema non è risolvibile

(c) viene superato un limite di ricerca

2 Se si è verificato (c) allora esegui ricerca locale finché

(a) viene individuata una soluzione

(b) viene superato un limite di ricerca $\Rightarrow$ espandi il grafo di pianificazione e `goto 1`

Combinazione di Ricerca Locale e Sistematica

Ricerca locale + ADJUST-PLAN

- Ricerca locale per calcolare efficientemente un *Quasi Soluzione*
- *Riparare* la quasi soluzione mediante ADJUST-PLAN
- Ottimizzazione della soluzione trovata

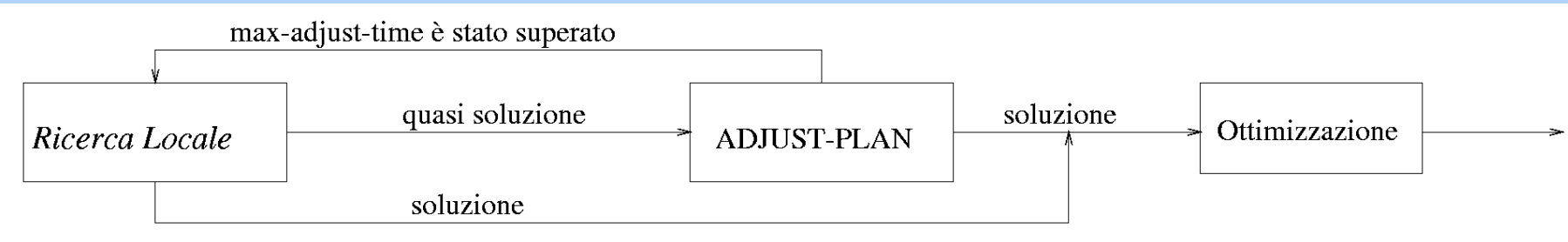
Una Quasi Soluzione corrisponde ad un Sottografo Azione in cui sono presenti un numero limitato di inconsistenze

Ricerca Locale + ADJUST-PLAN

1 Esegui ricerca locale per generare una soluzione o, se viene superato un limite di tempo, una Quasi Soluzione P con un numero predefinito di inconsistenze;

2 Esegui ADJUST-PLAN per riparare P. Se max-adjust-time viene superato `goto 1`

3 Fase di Ottimizzazione



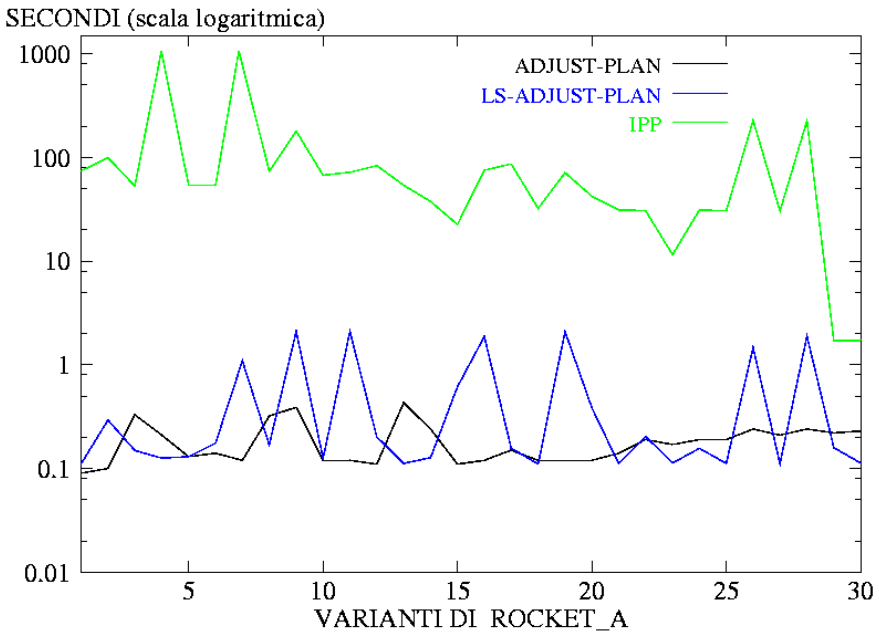
- Alla ricerca locale può essere fornito in input un piano π_0 da adattare.

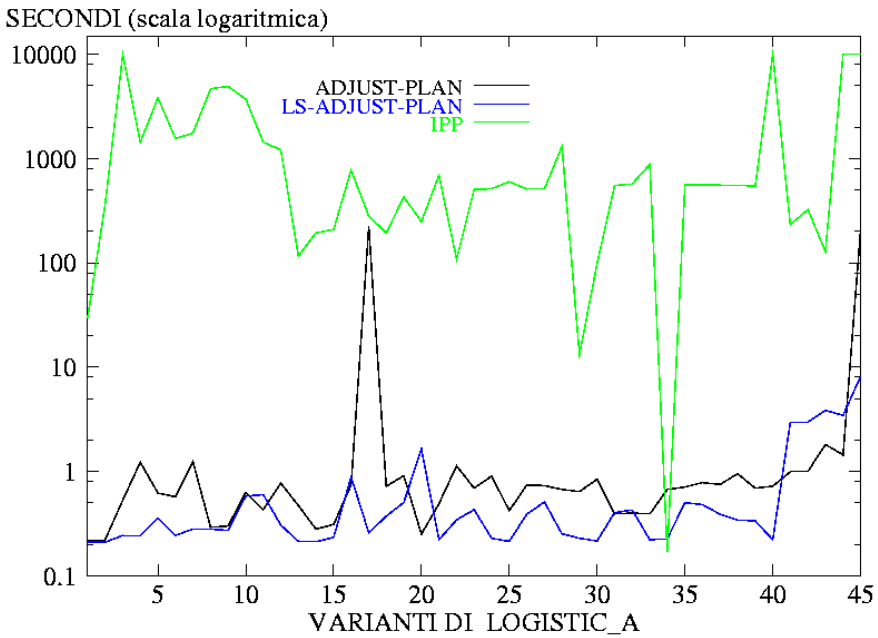
Schema generale di GPG

Input: *I, G, O,* $\underline{\pi_0}$
Output:un piano corretto o `fail`

```
1. goal_reached=Building_Graph(I, G, O);
2. if (goal_reached==FALSE) return "Probl. non risolvibile";
3. switch search
4.         case in LOCAL: found_plan=local_search(lev, π0);
5.                 if(found_plan==TRUE) print plan;
6.         case in ADJUST: found_plan=ADJUST_PLAN(lev, π0);
7.                 if(found_plan==TRUE) print plan;
8.         case in LS_ADJST: found_plan=local_search(lev, π0);
9.                 if(found_plan==TRUE) print plan;
10.                else found_plan=ADJUST_PLAN(lev, quasi_solution);
11.                    if(found_plan==TRUE) print plan;
12. if (found_plan==TRUE)  optimization(plan);
13. else return fail;
```

ADJUST-PLAN e LS-ADJUST-PLAN (Adattamento)





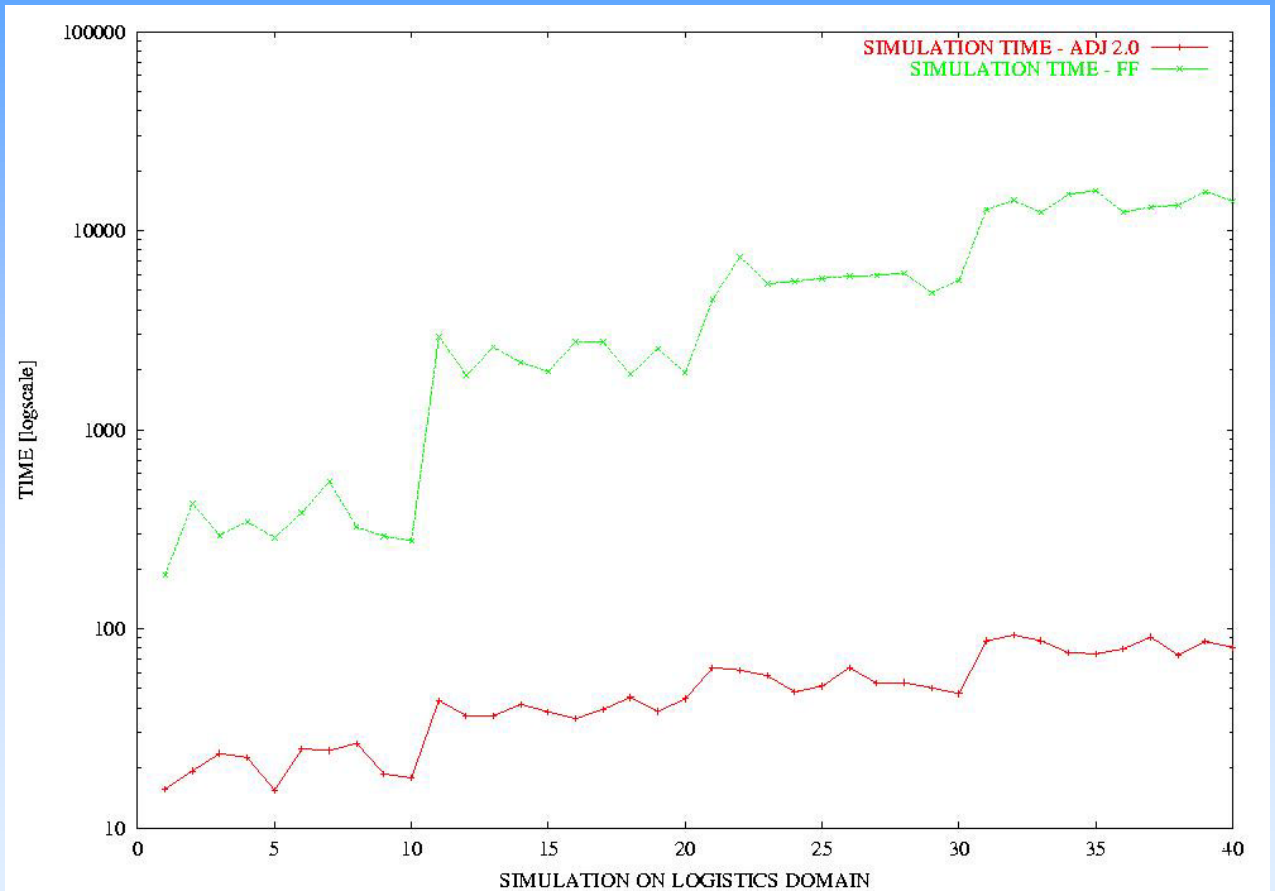
Simulatore “real-time”

- Simulazione = applicazione sequenziale delle azioni contenute nel piano
- Nel dominio sono inclusi degli **operatori di NOISE** opportunamente scritti
- Gli operatori di NOISE vengono selezionati in maniera casuale **(“noise rate”)**
- Il rumore può cambiare fatti iniziali (80%) e/o goal (20%) del problema
- Occorre adattare (ADJ) / generare (FF) un nuovo piano

Simulatore (Algoritmo)

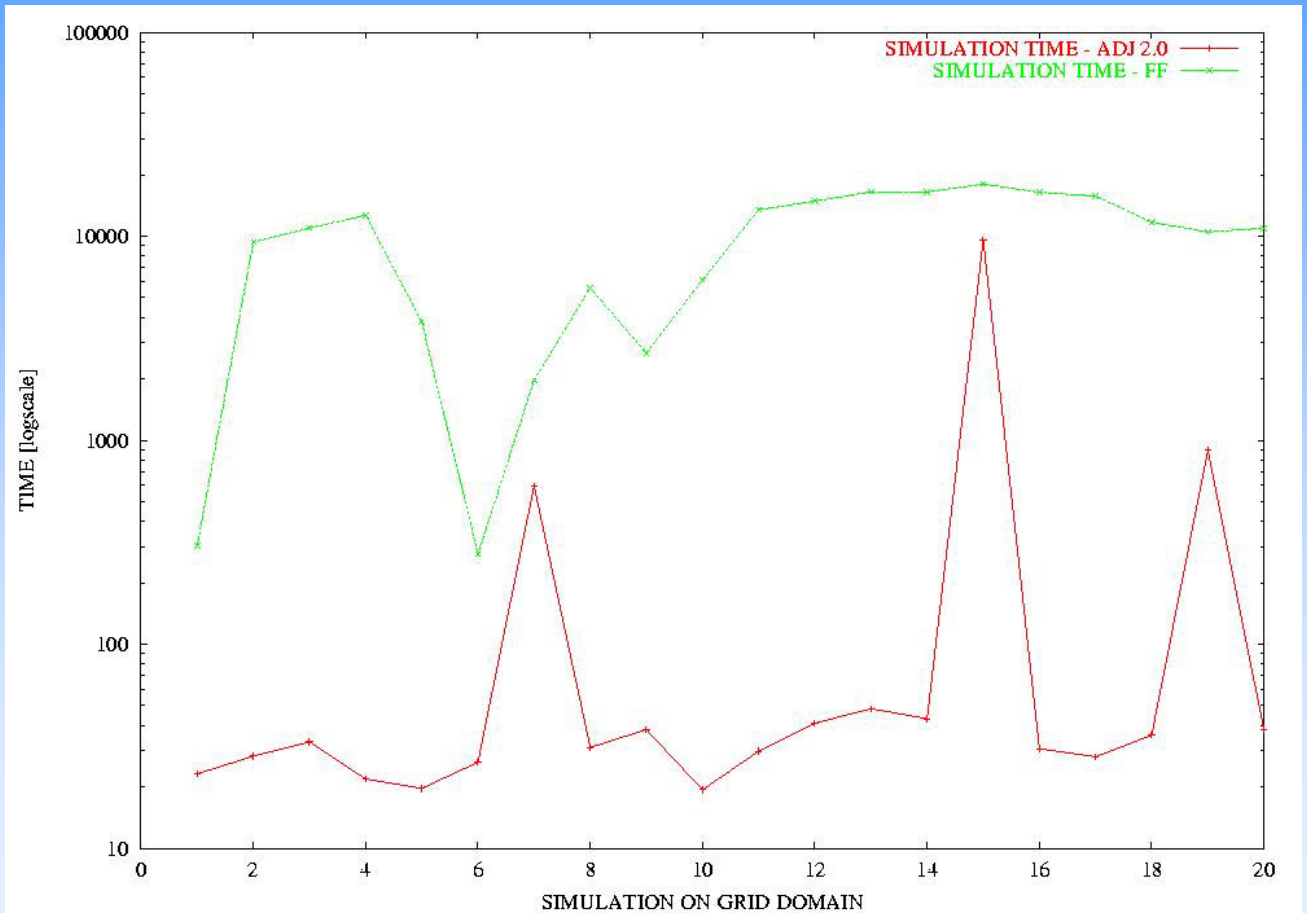
- Viene salvato lo stato corrente (valore di verità dei fatti)
- Si applica il NOISE selezionato
- Si verifica (tramite Forward Propagation) se il NOISE selezionato genera delle inconsistenze nel piano di input
- Si effettua la ripianificazione
- Oppure non si fa nulla e un operatore di NOISE sarà selezionato successivamente
- Quindi si ripristina/aggiorna lo stato attuale
- Si procede nella simulazione del piano di input

Il Dominio Logistics



Sistema ADJ - 2000/2002

Il Dominio Grid (Simulator)

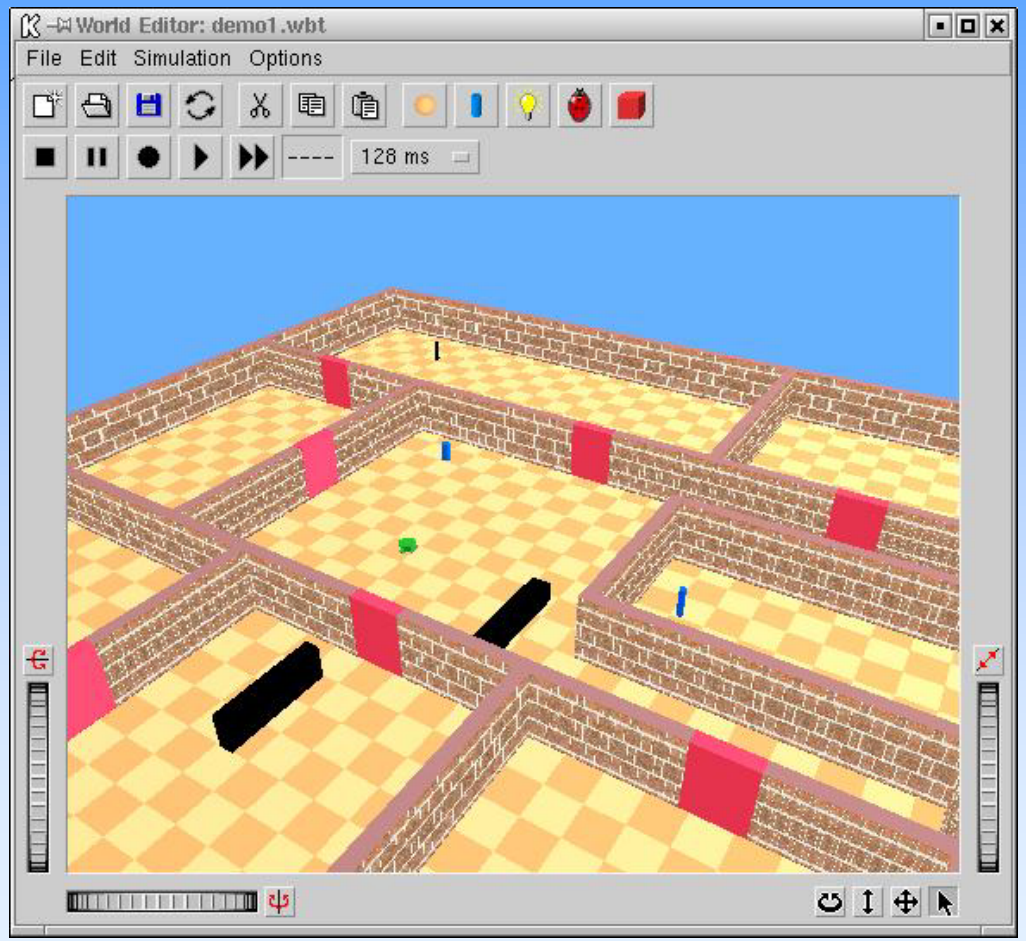


Sistema ADJ - 2000/2002

Il progetto Parsifal (1)

- **Mondo "virtuale"** costruito utilizzando il simulatore robotico Webots della Cyberbotics
- Robot e vari oggetti da muovere/trasportare
- **Robot "pilotati" da un pianificatore** (ADJ, FF)
- Robot appartenenti allo stesso gruppo seguono lo stesso piano (cooperano), mentre gruppi diversi possono avere goal contrastanti
- **Algoritmo di Dijkstra** per la simulazione delle azioni di movimentazione
- E' possibile **interagire con l'ambiente** creando situazioni in cui è necessario ripianificare

Il progetto Parsifal (2)



Conclusioni

- ADJ risulta fino a 2 ordini di grandezza più veloce di FF (il pianificatore attualmente più veloce)
- La struttura di ADJ è aperta e consente l'impiego di numerosi algoritmi di ricerca
- I piani trovato sono di "buona qualità": in genere sono paralleli e non contengono mai azioni "inutili" (link causali)
- In presenza di rumore casuale (simulatore) i risultati sono ancora a favore di ADJ

Sviluppi Futuri

- Estensione al linguaggio PDDL2 (costi e risorse)
- Conseguente revisione delle strutture dati e delle funzioni principali
- Raffinamento delle euristiche
- Ulteriori test sperimentali per validare il sistema in domini più complessi e realistici