Università degli studi di Brescia

Facoltà di Ingegneria

a.a. 1998 – 99

ANALISI E SIMULAZIONE DI TECNICHE DI PIANIFICAZIONE SIMBOLICA PER PROBLEMI DI ROBOTICA

Alessandra Bonzanini **N. matricola 024436**

Denise Salvi **N. matricola 022934**

INTRODUZIONE

La pianificazione simbolica è un problema di intelligenza artificiale che consiste nel determinare una sequenza di azioni che porti da uno stato iniziale ad uno stato finale applicando opportuni operatori definiti in un dominio. In questo lavoro abbiamo considerato due pianificatori indipendenti dal dominio basati su Planning Graph (Blum e Furst 1997) e sulla notazione STRIPS per la definizione dei domini. Tali pianificatori sono:

- IPP, un potente pianificatore sviluppato presso l'università di Friburgo;
- GPG, un pianificatore sviluppato presso l'università degli studi di Brescia che può generare piani ex novo oppure partire da un piano generato precedentemente per un problema simile e adattarlo in modo che risolva il problema posto.

Per i test abbiamo considerato alcuni domini classici (Rocket World, Logistic, Blocks World) e altri due domini orientati alla robotica: il dominio dei robot postini e il dominio del robot guardiano da noi formalizzato.

La relazione è organizzata in sezioni che descrivono:

- il Planning Graph nelle tecniche di pianificazione;
- le tecniche di pianificazione indipendenti dal dominio e in particolare le tecniche di generazione e adattamento di piani di GPG;
- il dominio dei robot guardiani e i risultati dei test delle tecniche di pianificazione nell'ambito di questo dominio;
- il dominio dei robot postini e i risultati dei test delle tecniche di pianificazione nell'ambito di questo dominio.

1. PIANIFICAZIONE MEDIANTE ANALISI DI PLANNING GRAPH

1.1. Introduzione

In questa sezione analizziamo la tecnica di pianificazione basata sui `Planning Graph` che utilizza la notazione STRIPS per la descrizione dei domini. A differenza dei metodi standard di pianificazione, che partono subito con una fase di ricerca, questa tecnica inizia costruendo esplicitamente una struttura compatta che viene detta `Planning Graph`. Un `Planning Graph`, per ridurre il carico di ricerca richiesto, codifica il problema di pianificazione in modo da rendere espliciti molti vincoli associati al problema. Inoltre i `Planning Graph` sono costruiti rapidamente: hanno una dimensione polinomiale (nella lunghezza della descrizione del problema e nel numero di livelli della soluzione) e possono essere costruiti in tempo polinomiale (rispetto agli stessi parametri di cui sopra).

`Graphplan` è un pianificatore che utilizza il `Planning Graph` per guidare la propria ricerca di un piano. La ricerca compiuta combina gli aspetti dei pianificatori total-order e partial-order. Come i tradizionali pianificatori total-order, Graphplan ha forti limiti nella sua ricerca. Quando considera un'azione, la considera in uno specifico istante nel tempo: per esempio, potrebbe considerare di piazzare l'azione "move Rocket1 from London to Paris" in un piano esattamente al time-step 2. D'altro canto, come i pianificatori partial-order [Chapman 1987][McAllester and Rosenblitt, 1991][Barrett and Weld, 1994][Weld, 1994], `Graphplan` genera piani parzialmente ordinati. Per esempio, nel mondo "RocketWorld" di Veloso (Figura 1), il piano che Graphplan genera è della forma:

"al time-step 1, carica appropriatamente tutti gli oggetti nei rocket"

"al time-step 2 muove i rocket"

"al time-step 3 scarica gli oggetti dai rocket".

La semantica di un piano di questo tipo è che tutte le azioni in un dato time-step possono essere svolte in qualsiasi ordine desiderato. Concettualmente è un tipo di piano "parallelo" [Knoblock, 1994], da cui qualcuno può immaginare di eseguire le azioni in tre time-step se ha a disposizione tanti operatori quanti ne servono per caricare, scaricare e far volare i rocket.

Una caratteristica importante di quest'algoritmo è la garanzia di trovare il piano più breve tra quelli in cui azioni indipendenti possono essere svolte nello stesso livello. Per esempio, nel mondo "flat-tire world" di Stuart Russel (l'obiettivo è riparare una ruota sgonfia e quindi portare indietro tutti gli attrezzi da dove provenivano), il piano prodotto da Graphplan apre il bagagliaio nel passo 1, prende tutti gli attrezzi e la ruota di scorta al passo 2, gonfia la ruota di scorta e allenta i bulloni al passo 3 e così via finché chiude il bagagliaio al passo 12 (vedi figura 2). Un'altra caratteristica rilevante dell'algoritmo è quella di non essere particolarmente sensibile all'ordine degli obiettivi nel planning task, contrariamente agli approcci tradizionali. Ulteriori discussioni di questa caratteristica sono date nel paragrafo 3.2.

Il dominio RocketWorld ha tre operatori: Load, Unload e Move. Un oggetto può essere caricato in un rocket se è nella sua stessa locazione, un rocket può muoversi se ha carburante, e quando si muove lo utilizza tutto. In STRIPS, gli operatori sono:

```
load
:v ?object cargo ?rocket rocket ?place place
:p at(?rocket ?place)  at(?object ?place)
:e ADD in(?object ?rocket)
   DEL at(?object ?place).

unload
:v ?object cargo ?rocket rocket ?place place
:p at(?rocket ?place)  in(?object ?rocket)
:e ADD at(?object ?place)
   DEL in(?object ?rocket).

move
:v ?rocket rocket ?from place ?to place
:p has-fuel(?rocket) at(?rocket ?from)
:e ADD at(?rocket ?to) no-fuel(?rocket)
   DEL has-fuel(?rocket) at(?rocket ?from).
```

Figura 1: gli operatori del dominio “RocketWorld”

```
Step 1:      open boot
Step 2: fetch wrench boot
        fetch pump boot
        fetch jack boot
        fetch wheel2 boot
Step 3: inflate wheel2
        loosen nuts the-hub
Step 4: put-away pump boot
        jack-up the-hub
Step 5: undo nuts the-hub
Step 6: remove-wheel wheel1 the-hub
Step 7: put-on-wheel wheel2 the-hub
        put-away wheel1 boot
Step 8: do-up nuts the-hub
Step 9: jack-down the-hub
Step 10:     put-away jack boot
        tighten nuts the-hub
Step 11:     put-away wrench boot
Step 12:     close boot
```

Figura 2: piano di Graphplan per il problema “fixit” di Russell

1.1.1. Definizioni e Notazioni

La PGA (Planning Graph Analysis) si applica a domini di pianificazione di tipo STRIPS [Fikes and Nilsson, 1971]. In questi domini, gli operatori hanno precondizioni, effetti additivi ed effetti cancellanti specificati tramite proposizioni ed hanno parametri che possono essere istanziati come oggetti del mondo. Gli operatori non creano né distruggono oggetti e il tempo può essere rappresentato in modo discreto. Un esempio è fornito in Figura 1.

Precisamente un problema di pianificazione è definito da:

- un dominio di tipo STRIPS (un insieme di operatori)
- un insieme di oggetti
- un insieme di proposizioni (letterali) chiamate condizioni iniziali
- un insieme di goal del problema che si richiede siano veri alla fine del piano.

Per `azione` si intende un operatore completamente istanziato. Per esempio, l'operatore "`put ?x into ?y`" può essere istanziato nell'azione specifica "`put object1 into container2`". Un'azione svolta al tempo t aggiunge al mondo tutte le propozioni che erano tra i suoi effetti additivi e cancella tutte le proposizioni che erano tra i suoi effetti cancellanti. Potrebbe essere utile propagare il valore di verità di una proposizione da un time-step al successivo; si introduce allora un tipo speciale di azione che chiamiamo `no-op` o `frame` action.

1.2. Piani Validi e Planning Graph

Un piano valido per un problema di pianificazione consiste in un insieme di azioni che devono essere eseguite in tempi specificati. Ci saranno azioni al tempo 0, azioni al tempo 1 e così via. Nello stesso time-step possono essere eseguite più azioni, a patto che non interferiscano tra loro. Precisamente, diciamo che due azioni interferiscono se una cancella una precondizione o un effetto additivo di un'altra. In un piano lineare queste azioni parallele e indipendenti possono essere svolte in qualsiasi ordine ottenendo esattamente lo stesso output. Un piano valido può svolgere un'azione al tempo `t > 1` se il piano rende vere tutte le sue precondizioni al tempo `t`. Dato che le azioni no-op propagano i fatti veri in avanti nel tempo, possiamo definire una proposizione vera al tempo `t > 1` se e soltanto se è un effetto additivo di qualche azione svolta al tempo `t − 1`. Infine, un piano valido deve rendere veri tutti i goal al livello finale.

1.2.1. Planning Graph

Un `Planning Graph` è simile ad un piano valido, ma senza il requisito che le azioni ad un dato time-step non interferiscano. In pratica è un tipo di grafo con vincoli che codifica il problema di pianificazione.

Più precisamente, un `Planning Graph` è un grafo diretto aciclico a livelli con due tipi di nodi e tre tipi di archi. I livelli si alternano tra livelli delle proposizioni che contengono nodi delle proposizioni (ognuno assegnato ad una specifica proposizione) e livelli delle azioni che contengono i nodi delle azioni (ognuno assegnato ad una specifica azione). Il primo livello del `Planning Graph` è un livello di proposizioni ed è formato da un nodo per ogni proposizione presente nelle condizioni iniziali. I livelli in un `Planning Graph` sono: proposizioni vere al tempo 0, possibili azioni al tempo 0, proposizioni che possono essere vere al tempo 1, possibili azioni al tempo 1, proposizioni che possono essere vere al tempo 2 e così via.

Gli archi in un Planning Graph rappresentano esplicitamente le relazioni tra le azioni e le proposizioni. I nodi delle azioni nel livello `i` sono connessi con archi di precondizione alle loro

precondizioni nel livello delle proposizioni `i`, con archi additivi ai loro effetti additivi nel livello delle proposizioni `i+1`, con archi cancellanti ai loro effetti cancellanti nel livello delle proposizioni `i+1`.

Le condizioni imposte ad un `Planning Graph` sono molto più deboli di quelle imposte ad un piano valido. Le azioni possono esistere al livello delle azioni `i` se tutte le loro precondizioni esistono al livello delle proposizioni `i` ma non c'è nessun requisito d'indipendenza. In particolare, il livello delle azioni `i` contiene tutte le possibili azioni le cui precondizioni esistono tutte al livello delle proposizioni `i`. Una proposizione può esistere al livello delle proposizioni `i + 1` se è un effetto additivo o cancellante di qualche azione al livello delle azioni `i`.

Un esempio di `Planning Graph` è dato in Figura 3.

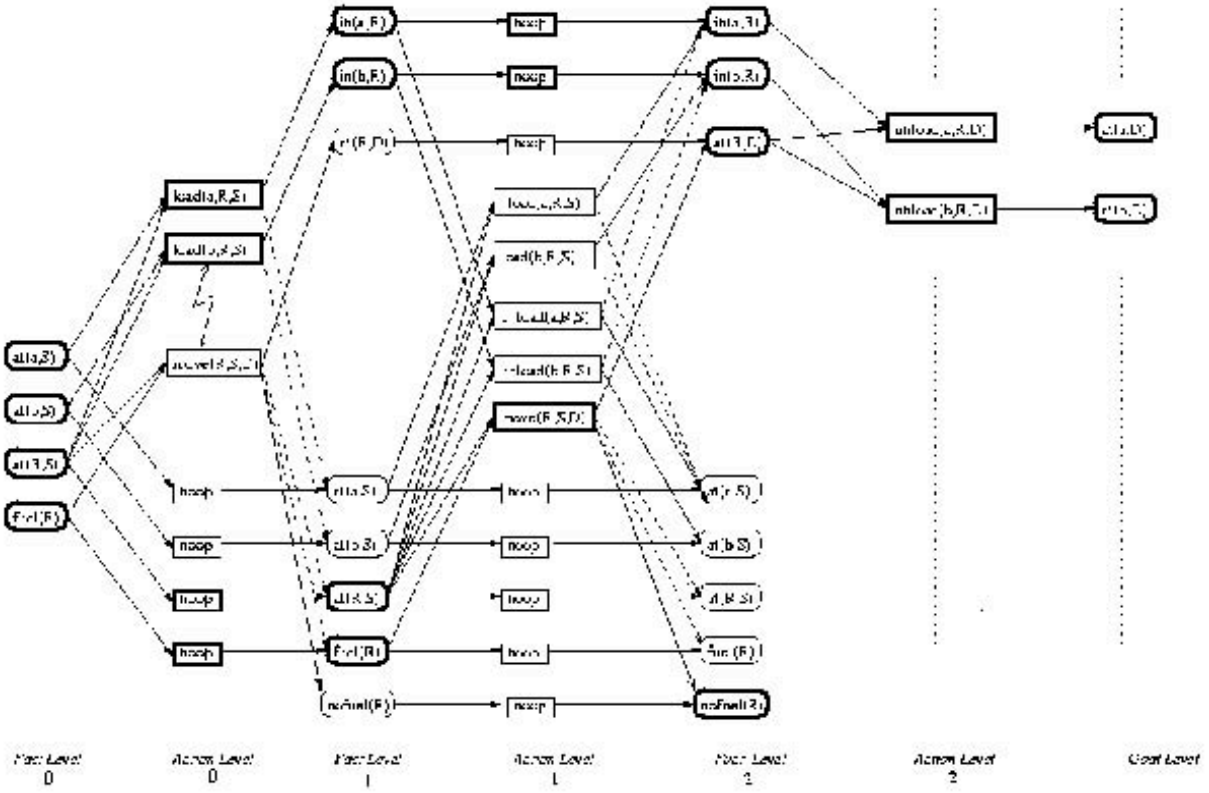


Figura 3: Esempio grafico di un Planning Graph nel dominio Rocket.

Dato che i requisiti sul `Planning Graph` sono così deboli, è facile crearlo. Nel paragrafo 1.3 viene descritto come Graphplan costruisce il `Planning Graph` partendo dai domini e dai problemi. In particolare, ogni `Planning Graph` con `t` livelli delle azioni che Graphplan crea avrà le seguenti proprietà:

- se esiste un piano valido usando `t` o meno time-step, allora quel piano esiste come sottografo del `Planning Graph`
- non è garantito tuttavia che il `Planning Graph` sia di dimensioni contenute. Vedi teorema 1.

1.2.2. Relazioni d'Esclusione tra i Nodi del Planning Graph

Due azioni in un dato livello delle azioni sono mutuamente esclusive se nessun piano valido le può contenere contemporaneamente. Analogamente, due proposizioni ad un dato livello delle proposizioni sono mutuamente esclusive se nessun piano valido può renderle vere entrambe nello stesso istante temporale. L'identificazione delle relazioni di mutua esclusione può essere di enorme aiuto nel ridurre la ricerca di un sottografo in un `Planning Graph` che possa corrispondere ad un piano valido.

Graphplan individua e registra le relazioni di mutua esclusione propagandole lungo il `Planning Graph` usando poche semplici regole, che però non garantiscono di trovare tutte le relazioni di mutua esclusione. Precisamente, vi sono due casi in cui due azioni `a` e `b`, ad un dato livello delle azioni, sono marcate da Graphplan come mutuamente esclusive:

- **interferenza**: si ha quando una delle azioni cancella una precondizione o un effetto additivo dell'altra.
- **esigenze contrastanti**: si ha quando una precondizione dell'azione `a` e una precondizione dell'azione `b` sono marcate come mutuamente esclusive nel precedente livello delle proposizioni.

Due proposizioni `p` e `q` nel livello delle proposizioni sono marcate come mutuamente esclusive se tutti i modi per rendere vera la proposizione `p` sono esclusivi con tutti i modi di rendere vera la proposizione `q`.

Per esempio, nel dominio dei rocket (vedi figura 3) con "`R at S`" e "`has-fuel R`" nelle condizioni iniziali, le azioni "`move R from S to D`" e "`Load a into R in S`" al tempo 1 sono esclusive perché la prima cancella la proposizione "`R at S`" che è una precondizione della seconda. Le proposizioni "`R at S`" e "`R at D`" sono esclusive al tempo 2 perché tutti i modi per generare la prima (ce n'è uno solo: no-op) sono esclusive con tutti i modi per generare la seconda (ce n'è uno solo: move). L'azione "`load a into R in S`" e "`load b into R in D`" (assumendo di aver definito nelle condizioni iniziali che b sia in D) al tempo 2 sono esclusive perché hanno esigenze contrastanti, cioè le proposizioni "`R at S`" e "`R at D`".

Due proposizioni possono diventare mutuamente esclusive in ogni livello di un `Planning Graph` oppure possono essere esclusive nei primi livelli e poi non esserlo più nei livelli successivi. Per esempio, se iniziamo con l'oggetto a e il rocket R nella città S (e non sono in nessun altro posto al tempo 1) allora "`a in R`" e "`R in D`" sono esclusive al tempo 2, e non lo sono più al tempo 3.

1.2.2.1. L'utilità delle relazioni di mutua esclusione

Si noti che la nozione d'esigenze contrastanti e di mutua esclusione tra proposizioni non sono solo proprietà logiche degli operatori. Piuttosto, esse dipendono dalle relazioni di causa-effetto tra operatori e condizioni iniziali.

Si consideri, per esempio, un dominio come quello dei Rocket avente un operatore "`move`". L'utile concetto che un oggetto non può essere in due posti nello stesso tempo non è solo una funzione degli operatori; se le condizioni iniziali specificano che l'oggetto sta inizialmente in due posti differenti, allora esso potrebbe continuare ad essere in due posti contemporanemente. Invece questo concetto dipende sia dalla nozione di "move" che dal fatto che l'oggetto sia all'inizio in un solo posto. Le regole di mutua esclusione forniscono un meccanismo per la propagazione di questo concetto lungo il grafo. La ragione è che se al tempo `t - 1` l'oggetto può essere in un solo posto, allora due qualsiasi azioni "`move`" sullo stesso rocket eseguibili al tempo `t - 1` risulteranno esclusive (due qualsiasi azioni "`move`" da diverse posizioni di partenza sono esclusive per esigenze contrastanti e due azioni "`move`" dalla stessa posizione di partenza sono esclusive perché cancellano l'una le precondizioni dell'altra) e quindi l'oggetto può essere in un solo posto al tempo `t`.

Più in generale, in molti domini differenti le relazioni d'esclusione sembrano propagare attraverso il grafo una varietà di fatti utili ed intuitivi che riguardano il problema.

1.3. Descrizione dell'Algoritmo

La descrizione ad alto livello dell'algoritmo di Graphplan è la seguente. Partendo da un Planning Graph che possiede un solo livello delle proposizioni contenente le condizioni iniziali, Graphplan procede considerando un livello per volta. Nel livello `i` Graphplan considera il Planning Graph del livello `i - 1`, lo estende di un time-step (definendo il prossimo livello delle azioni e delle proposizioni), e quindi cerca nel Planning Graph espanso un piano valido di lunghezza `i`. Se la tecnica di ricerca di Graphplan non trova un piano valido (nel qual caso si blocca), significa che gli obiettivi non sono raggiungibili al tempo `i` (nel qual caso si espande di un altro livello). Così ad ogni iterazione di questo ciclo d'estensione/ricerca l'algoritmo o scopre un piano oppure prova che non è possibile avere un piano con quel numero o un numero inferiore di passi.

Si dimostra che l'algoritmo di Graphplan è robusto e completo: ogni piano che l'algoritmo trova è un piano valido, e se esiste un piano valido allora Graphplan lo trova. Nel paragrafo 1.4 viene descritto come questo algoritmo possa essere migliorato in modo che sia garantito che il pianificatore si fermi con un fallimento in un tempo finito nel caso in cui i goal del problema non possano essere soddisfatti da nessun piano valido. Questo non è garantito da molti pianificatori ad ordinamento parziale.

1.3.1. Espansione dei Planning Graph

Le condizioni iniziali del problema costituiscono il primo livello delle proposizioni del grafo. Per creare un generico livello delle azioni, si opera nel seguente modo. Per ogni operatore ed ogni modo per istanziare le precondizioni di tale operatore con proposizioni del livello precedente, si inserisce un nodo delle azioni se questo non ha precondizioni marcate come mutuamente esclusive. Si inseriscono inoltre tutte le azioni no-op e si inseriscono gli archi delle precondizioni. Si controllano poi i nodi delle azioni per le relazioni di mutua esclusione come descritto nel paragrafo 1.2.2 e si crea una lista di "azioni con le quali sono mutuamente esclusive" per ogni azione.

Per creare un generico livello delle proposizioni, si guardano semplicemente tutti gli effetti additivi delle azioni nel precedente livello (incluse le no-op) e si posizionano nel prossimo livello come proposizioni, connettendole mediante gli appropriati archi additivi o cancellanti. Si marcano due proposizioni come esclusive se tutti i modi di generare la prima sono esclusivi con tutti i modi di generare la seconda.

Come dimostrato nel seguente teorema, il tempo che occorre all'algoritmo per creare questa struttura a grafo è polinomiale rispetto alla lunghezza della descrizione del problema e al numero di livelli.

> **Teorema 1.** *Si consideri un problema di pianificazione con n oggetti, p proposizioni nelle condizioni iniziali e m operatori STRIPS ognuno dei quali possiede un numero costante di parametri formali. Sia l la lunghezza della più lunga lista additiva di ogni operatore. Allora, la dimensione di un grafo di pianificazione a t livelli creata da Graphplan, e il tempo necessario per creare il grafo, sono polinomiali in n , m, p, l e t.*
>
> *Dimostrazione.* Sia k il più grande numero di parametri formali in ogni operatore. Dal momento che gli operatori non possono creare nuovi oggetti, il numero delle diverse proposizioni che possono essere create istanziando un operatore sono O(lnk). Quindi, il massimo numero di nodi in ogni livello delle proposizioni del grafo di

pianificazione è O(p+mlnk). Dal momento che ogni operatore può essere istanziato in al massimo O(nk) modi distinti, il massimo numero di nodi in ogni livello delle azioni del grafo di pianificazione è O(mnk). Quindi la dimensione totale del grafo di pianificazione è polinomiale in n, m, p, l e t, dato che k è costante.
Il tempo necessario per creare un nuovo livello delle azioni o proposizioni del grafo può essere ricondotto a:
(A) il tempo per istanziare gli operatori in tutti i possibili modi dalle precondizioni nel precedente livello delle precondizioni;
(B) il tempo per determinare le relazioni di mutua esclusione tra le azioni;
(C) il tempo per determinare le relazioni di mutua esclusione nel prossimo livello delle proposizioni.
E' chiaro che questo tempo risulta polinomiale nel numero di nodi presenti nel livello corrente del grafo. □

Sperimentalmente si trova che la fase di creazione del grafo che occupa più tempo è determinare le relazioni di mutua esclusione. Tuttavia, sperimentalmente, la sola creazione del grafo occupa una parte significativa del tempo d'esecuzione di Graphplan nei problemi più semplici, dove il tempo totale d'esecuzione non è comunque molto alto.

Un ovvio miglioramento all'algoritmo base descritto sopra (che è implementato in Graphplan) è quello di evitare la ricerca fino a che non sia stato creato un livello delle proposizioni nel quale appaiano tutti i goal del problema e nessuna coppia di goal sia mutuamente esclusiva.

1.3.2. La ricerca di un piano

Dato un Planning Graph, Graphplan cerca un piano valido usando la strategia backward-chaining. Al contrario di molti altri pianificatori, tuttavia, utilizza un approccio level-by-level, per utilizzare al meglio i vincoli di mutua esclusione. In particolare, dato un insieme di goal al tempo t, cerca di trovare un'insieme di azioni (no-op incluse) al tempo t – 1 che hanno questi goal come effetti additivi. Le precondizioni per queste azioni formano un'insieme di sotto obiettivi al tempo t – 1 che hanno la proprietà che se questi obiettivi possono essere raggiunti in t – 1 passi, allora gli obiettivi originali possono essere raggiunti in t passi. Se scopre che l'insieme degli obietti al tempo t – 1 non è raggiungibile, Graphplan cerca di trovare un'insieme diverso di azioni, continuando fino a che o ha successo oppure riesce a dimostrare che l'insieme originale di goal non è raggiungibile al tempo t.

Per implementare questa strategia, Graphplan utilizza il seguente metodo di ricerca ricorsivo:

1) per ogni goal al tempo t in un ordine arbitrario, seleziona un'azione al tempo t – 1 che raggiunga quel goal e non sia esclusiva con nessuna azione precedentemente selezionata. Continua ricorsivamente con il prossimo goal al tempo t (naturalmente, se un obiettivo è già stato raggiunto con un'azione selezionata precedentemente, non c'è bisogno di selezionare una nuova azione per esso);
2) se la chiamata ricorsiva restituisce un fallimento, allora tenta una diversa azione per raggiungere l'obiettivo corrente, e così via ritornando fallimento una volta che sono state tentate tutte le azioni. Una volta finito con tutti i goal al tempo t, le precondizioni alle azioni selezionate diventano il nuovo insieme di goal al tempo t – 1.

Un miglioramento di quest'approccio con un "controllo in avanti" (che è implementato in Graphplan) è assicurarsi dopo ogni azione considerata che non venga reso falso nessun goal non ancora considerato nella lista. In altre parole, Graphplan controlla se per qualche goal ancora in lista, tutte le azioni che lo creano sono esclusive con le azioni correntemente selezionate. Se esiste un tale goal, allora Graphplan sa che deve subito fare backtracking.

1.3.2.1. Memoization

Un altro aspetto della ricerca di Graphplan è che, quando si determina che un insieme di (sotto)obiettivi al tempo `t` non è risolvibile, allora prima di tornare indietro nella ricorsione si "memoizza" ciò che si è imparato, immagazzinando l'insieme di obiettivi e il tempo `t` in una tabella hash. Similmente, quando si crea un insieme di goal al tempo `t`, prima di cercarli si controlla la tabella hash per vedere se l'insieme è già stato determinato come non risolvibile. Se è così, il programma effettua backtracking senza ulteriori ricerche. Questo passo di "memoizing", oltre che essere utilizzato per velocizzare la ricerca, è utile per in controllo di terminazione descritto nel paragrafo 1.4.

1.3.2.2. L'effetto limitato dell'ordinamento degli obiettivi

La strategia di lavorare sui sotto obiettivi in un modo breadth-first rende Graphplan abbastanza insensibile all'ordinamento degli obiettivi. Descriviamo ora un'ultima modifica della strategia di ricerca di Graphplan. Sia `G` un insieme d'obiettivi al tempo `t`. Diciamo che un insieme di azioni `A` non esclusive al tempo `t − 1` è un insieme minimo di azioni che soddisfano `G` se

(1) ogni obiettivo in `G` è un'effetto additivo di qualche azione in `A`,
(2) nessuna azione può essere rimossa da `A` in modo che gli effetti additivi delle azioni rimanenti contengano ancora `G`.

La modifica della strategia di Graphplan è di ricorrere solo ad insiemi minimi d'azioni. Se l'insieme d'azioni `A` scelto da Graphplan per ottenere alcuni insiemi d'obiettivi `G` non è minimo, si fa subito backtracking. (Per esempio, supponiamo che i nostri obiettivi siano g_1 e g_2; scegliamo una qualche azione per ottenere g_1 e poi l'azione che scegliamo successivamente per raggiungere g_2 raggiunge anche g_1. Questo non sarebbe minimo). Questa modifica permette di stabilire chiaramente quali insiemi d'obiettivi Graphplan considera. Più precisamente, si può enunciare il seguente teorema.

> **Teorema 2.** *Sia G un insieme di obiettivi al tempo `t` che non è raggiungibile in `t` passi. Allora, indipendentemente da quale sia l'ordine degli obiettivi in G, l'insieme di obiettivi al tempo `t − 1` che Graphplan considera quando cerca di ottenere G sono esattamente le precondizioni di tutti gli insiemi minimi di azioni al tempo `t − 1` che ottengono G. (Se G è risolvibile in `t` passi, allora Graphplan può fermarsi prima di considerare tutti questi insiemi di obiettivi).*
>
> *Dimostrazione.* Si è fatto in modo che Graphplan consideri solo insiemi minimi di azioni; bisogna dimostrare che tutti questi insiemi vengono esaminati. Sia `A` uno di tali insiemi e si consideri un ordinamento arbitrario di `G`. Sia a_1 un'azione in `A` che permette di ottenere il primo obiettivo in `G` (e chiamiamo tale obiettivo g_{a1}). Sia a_2 l'azione che permette di ottenere il primo obiettivo di `G` non ottenuto con a_1 (e sia tale obiettivo g_{a2}). Più in generale, sia a_i l'azione in `A` che permette di ottenere il primo obiettivo in `G` non ottenuto con nessuno degli $\{a_1,\ldots a_{i-1}\}$, e chiamiamo tale obiettivo g_{ai}. Si noti che tutte le azioni in `A` sono fornite di un indice in questo modo poiché `A` è minimo. Quest'ordine delle azioni implica che, ad un certo punto della ricorsione, a_1 sarà l'azione scelta da Graphplan per ottenere l'obiettivo g_{a1}; a_2 sarà l'azione scelta da Graphplan per ottenere l'obiettivo g_{a2} e così via. Quindi tutte le azioni in `A` vengono esaminate. □

Possiamo quantificare il limitato effetto dell'ordinamento degli obiettivi come segue. Supponiamo che Graphplan stia attualmente tentando di risolvere gli obiettivi del problema ad un dato tempo `T` e non abbia successo. Allora il numero totale d'insiemi obiettivo esaminati nella

ricerca è completamente indipendente dall'ordinamento degli obiettivi. L'effetto dell'ordinamento degli obiettivi è limitato a:

(A) la quantità di tempo che impiega in media per esaminare un nuovo insieme di obiettivi (il passo di creazione di un insieme di obiettivi) e

(B) la quantità di lavoro fatta nell'ultimo passaggio nel quale gli obiettivi del problema sono stati risolti (in quanto l'ordinamento dei goal può influenzare l'ordine in cui gli insiemi obiettivo sono esaminati). In aggiunta a quest'affermazione teorica, sperimentalmente la dipendenza di Graphplan dall'ordinamento degli obiettivi sembra essere molto piccola: decisamente minore rispetto a quella di altri pianificatori come Prodigy e UCPOP.

1.4. Terminazione di problemi non risolvibili

In prima approssimazione, Graphplan effettua qualcosa di simile ad una ricerca iterativa in profondità. Nell'i-mo passo l'algoritmo controlla se c'è un piano parallelo valido di lunghezza inferiore od uguale ad i. Come descritto finora, se non esiste un piano valido nulla previene l'algoritmo dal passare stupidamente attraverso un numero infinito di passi.

Segue un semplice ed efficiente test, che può essere aggiunto dopo ogni passo che non ha avuto successo, in modo che se il problema non ha soluzione Graphplan si fermi e dichiari "No Plan Exists".

1.4.1. Planning Graphs stabilizzati

Supponiamo che un problema non abbia un piano valido. Per prima cosa si osserva che nella sequenza del `Planning Graph` creato ci sarà ad un certo punto un livello di proposizioni P tale che tutti i livelli di proposizioni futuri siano esattamente uguali a P, in altre parole essi contengono lo stesso insieme di proposizioni ed hanno le stesse relazioni di mutua esclusione. La ragione di ciò è la seguente. A causa delle azioni no-op, se una proposizione appare in qualche livello delle proposizioni allora essa apparirà in tutti i futuri livelli delle proposizioni. Dato che solo un insieme finito di proposizioni può essere creato dagli operatori di tipo STRIPS (quando sono applicati ad un insieme finito di condizioni iniziali) ci deve essere un qualche livello delle proposizioni Q tale che tutti i futuri livelli abbiano esattamente lo stesso insieme di proposizioni di Q. Inoltre, ancora a causa delle azioni no-op, se le proposizioni p e q appaiono insieme in un dato livello e non sono marcate come mutuamente esclusive, allora non saranno più marcate come mutuamente esclusive in nessun livello futuro. Così ci deve essere un livello delle proposizioni P dopo Q tale che tutti i futuri livelli delle proposizioni abbiano esattamente lo stesso insieme di relazioni di mutua esclusione di P. Quindi non è difficile vedere che una volta che due livelli adiacenti P_n e P_{n+1} sono uguali, allora tutti i futuri livelli saranno uguali a P_n. A questo punto, si dice che il grafo è stabilizzato.

1.4.2. Un test facile e veloce

Sia P_n il primo livello delle proposizioni in cui il grafo è stabilizzato. Se qualche obiettivo del problema non appare in questo livello, o se due obiettivi del problema sono marcati come mutuamente esclusivi in tale livello, allora Graphplan può dire immediatamente che il piano non esiste. Si noti che in questo caso Graphplan è in grado di fermarsi senza alcuna ulteriore ricerca. Tuttavia, in alcuni casi può succedere che non esista un piano e questo semplice test non lo rilevi. Un esempio è il "mondo dei blocchi" con tre blocchi, in cui l'obiettivo è che il blocco A sia sopra il blocco B, e che il blocco B sia sopra il blocco C e il blocco C sopra il blocco A; ogni coppia di questi obiettivi è raggiungibile ma non lo sono tutti e tre contemporaneamente. Quindi abbiamo bisogno di qualcosa di più raffinato per garantire la terminazione in ogni caso.

1.4.3. Un test di terminazione garantito

Come menzionato sopra, Graphplan memorizza l'insieme d'obiettivi che ha considerato ad un dato livello e che ha deciso essere non risolvibili. Sia S_i^t la collezione di tutti questi insiemi immagazzinati per ogni livello i dopo una fase t senza successo. In altre parole, dopo una fase senza successo t, Graphplan ha determinato due cose:

(1) ogni piano di t o meno passi deve rendere vero al tempo i uno degli insiemi obiettivo in S_i^t;

(2) nessuno degli insiemi obiettivo in S_i^t è raggiungibile in t passi. La modifica a Graphplan che assicura la terminazione è descritta dal seguente teorema:

Teorema 3. S*e il grafo è stabilizzato in qualche livello n e c'è stata una fase t nella quale* $|S_i^{t-1}| = |S_i^t|$*, allora Graphplan dichiara "No Plan Exists", e ciò accade se e soltanto se il problema è irrisolvibile.*

Dimostrazione. Un modo facile per dimostrare ciò è osservare che, se il problema non è risolvibile, allora ad un certo punto Graphplan dichiarerà che non esiste un piano. La ragione di ciò è che il numero d'insiemi in S_n^t non è mai più piccolo del numero di insiemi di S_n^{t-1}, e che c'è un limite massimo finito (benché esponenziale nel numero di nodi del livello n). Per dimostrarlo, si supponga che il grafo sia stabilizzato al livello n e che Graphplan abbia completato una fase $t > n$ senza successo. Ogni piano per raggiungere un insieme di S_{n+1}^t deve, un passo prima, raggiungere un insieme di S_n^t. Questo a causa del modo in cui Graphplan opera: determina che ogni insieme in S_{n+1}^t è irrisolvibile mappandolo negli insiemi al time-step n e determinando che essi sono irrisolvibili. Dato che il grafo è "leveled off", $S_{n+1}^t = S_n^t$. Questo perché gli ultimi t-n livelli del grafo sono uguali, indipendentemente da quanti altri livelli in più ha il grafo.

Si supponga ora che dopo una fase t senza successo, sia $|S_n^{t-1}| = |S_n^t|$ (che implica che $S_n^{t-1} = S_n^t$). Ciò significa che $S_{n+1}^t = S_n^t$. Quindi per raggiungere ogni insieme in S_{n+1}^t si deve aver raggiunto qualche altro insieme in S_{n+1}^t. Quindi nessuno degli insiemi in S_{n+1}^t è contenuto nelle condizioni iniziali, e il problema è irrisolvibile. □

2. ESPLORAZIONE DI PLANNING GRAPH PER L'ADATTAMENTO DI PIANI

2.1. Introduzione

Un tipico problema d'adattamento di piani consiste nel modificare un piano, precedentemente generato, per risolvere un nuovo problema "simile" all'originale. Questo processo può avvenire sia off-line (per esempio, adattare un piano ricavato da una libreria di piani prima della sua esecuzione), che on-line (per esempio, adattare un piano durante una costruzione "mixed-iniative" o durante la sua esecuzione). Una pianificazione off-line veloce è importante per esempio quando, durante l'esecuzione di un piano alcune azioni falliscono, o l'acquisizione di nuove informazioni modifica la descrizione del mondo o degli obiettivi del piano, rendendo il piano corrente non più valido.

Da un punto di vista teorico, nel peggiore dei casi l'adattamento di piani non è più efficiente di una completa rigenerazione del piano. Tuttavia, in pratica l'adattamento di un piano esistente per risolvere un nuovo problema può essere molto più efficiente che generare completamente un nuovo piano, specialmente quando i cambiamenti del piano richiesti riguardano solo alcune parti circoscritte del piano.

GPG (Greedy Planning Graph) è un sistema di pianificazione indipendente dal dominio che si basa sui Planning Graph. Utilizza una collezione di tecniche di ricerca locale e sistematica per risolvere sia problemi di generazione sia d'adattamento di piani. Questo paragrafo si occupa dell'adattamento di piani, dove come input si ha: un dominio di pianificazione (un insieme di operatori) D; un piano valido per il problema P in D, specificato da una lista di azioni; un problema modificato P' in D specificato da un insieme di cambiamenti allo stato iniziale o allo stato finale di P. L'output è un piano valido per P'.

Di seguito presentiamo tre metodi per l'adattamento di piani che sono implementati in GPG. Il primo metodo modifica il piano originale utilizzando una tecnica di ricerca locale; il secondo utilizza una ricerca sistematica per correggere i difetti nel piano originale ripianificando all'interno di una finestra temporale fissata; la terza è una combinazione dei due metodi precedenti.

In generale, si nota che adattare un piano utilizzando le nostre tecniche è in pratica più efficiente di una ripianificazione completa. Questo è vero soprattutto per le modifiche che richiedono cambiamenti che tendono ad essere localizzati in una specifica parte del piano. Il metodo di ricerca locale è particolarmente efficiente quando il problema modificato ammette una soluzione che non richiede più livelli rispetto al piano originale, ma le sue prestazioni dipendono dai valori assegnati a certi parametri di una funzione euristica. La tecnica di ricerca sistematica può efficientemente risolvere problemi d'adattamento che richiedono un numero più elevato di livelli, ma può determinare piani adattati che richiedono più livelli di quelli necessari (specialmente quando esiste una soluzione con lo stesso numero di livelli del piano originale). Il metodo combinato utilizza sia la ricerca locale sia la ricerca sistematica per combinare i vantaggi dei due metodi precedenti e superare i loro limiti: può efficientemente risolvere l'adattamento di piani utilizzando dei valori di default per i parametri euristici, e in generale riesce a trovare un piano con un numero di livelli minore del metodo sistematico.

Infine, presentiamo brevemente il problema della qualità di un piano adattato nel contesto della rappresentazione con Planning Graph, e un semplice processo d'ottimizzazione del numero di livelli del piano adattato attraverso il metodo sistematico, che calcola una successione di piani migliorati.

2.2. Adattamento di piani mediante ricerca locale

Il nostro metodo di ricerca locale per un problema dato *P* in un Planning Graph *G* è un processo che, partendo da un sottografo iniziale *G'* di *G*, trasforma *G'* in una soluzione di *P*. Ciò è realizzato mediante l'applicazione iterativa d'alcune modifiche al grafo volte a trovare, tra le possibili in uno stesso livello, quella con le "qualità" migliori. Ogni modifica è o un'estensione del sottografo per includere un nuovo nodo azione di *G*, o una riduzione del sottografo per rimuovere un nodo azione (e gli archi ad esso connessi).

Aggiungere un nodo azione al sottografo corrisponde ad inserire un'azione nel piano parziale rappresentato dal sottografo (e analogamente per la rimozione di un nodo azione). In qualsiasi passo del processo di ricerca l'insieme di azioni che possono essere aggiunte o rimosse viene determinato dal numero di violazioni di vincoli che sono presenti nel sottografo corrente di *G*. Tali violazioni corrispondono a:

- relazioni di mutua esclusione che coinvolgono nodi azione nel sottografo corrente;
- fatti non supportati che sono o precondizioni di azioni nel piano parziale corrente, o nodi goal nell'ultimo livello del grafo.

Più precisamente, lo spazio di ricerca è formato da un *sottografo d'azione* di *G*, definito come:

Definizione 1 Un **sottografo d'azione** A di un Planning Graph G è un sottografo di G tale che se [a] è un nodo azione di G in A, allora i nodi fatto di G corrispondenti alle precondizioni e agli effetti additivi di [a] sono anch'essi in A insieme agli archi di G che li connettono con [a].

Definizione 2 Un **sottografo soluzione** di un Planning Graph G è un sottografo d'azione A_s contenente i nodi goal di G e tale che:
- tutti i nodi goal e i nodi fatto corrispondenti alle precondizioni di azioni di A_s siano supportati;
- non vi siano relazioni di mutua esclusione tra nodi azione.

Lo schema generale per la ricerca della soluzione consiste in due fasi principali:
- fase di **inizializzazione** della ricerca con la costruzione di un sottografo d'azione iniziale;
- fase di **ricerca locale** nello spazio di tutti i sottografi azione, partendo dal sottografo d'azione iniziale.

Nel contesto della ricerca locale la fase di inizializzazione è un passo importante che può influenzare significativamente le prestazioni della fase di ricerca (Minton 1992). In GPG è stato scelto un sottografo d'azione generato casualmente.

Quando l'inizializzazione é stata completata, la fase di ricerca inizia con la scelta casuale di una violazione di vincoli nel sottografo d'azione corrente. Se si tratta di un nodo fatto non supportato, per eliminare la violazione di vincoli è possibile sia aggiungere un nodo azione che la supporti, sia rimuovere un nodo azione che è connesso al nodo fatto mediante un arco di precondizione.

Se la violazione di vincoli scelta è una relazione di mutua esclusione, possiamo rimuovere uno dei nodi azione della relazione di mutua esclusione. Si noti che l'eliminazione di un singolo nodo azione può portare all'eliminazione di più violazioni di vincoli (per esempio tutte quelle corrispondenti all'insieme di relazioni di esclusione coinvolgenti il nodo azione eliminato). D'altro canto, l'aggiunta di un nodo azione può introdurre più violazioni di vincoli. Inoltre, quando si aggiunge (rimuove) un nodo per soddisfare una violazione di vincoli, si aggiungono (rimuovono) anche tutti gli archi che connettono il nodo azione con le corrispondenti precondizioni e nodi effetto

nel Planning Graph; questo assicura che ogni modifica al sottografo d'azione corrente sia ancora un sottografo d'azione.

La scelta delle violazioni di vincoli può essere guidata da una funzione obiettivo che viene definita nel seguente modo:

Definizione 3 Dato un piano parziale π rappresentato da un sottografo d'azione A, la funzione obiettivo f(n) di π è definita come:

$$f(\pi) = g(A) + \sum_{a \in A} [me(a, A) + p(a, A)]$$

dove a è un'azione di A, me(a, A) è il numero di nodi azione di A che sono mutuamente esclusivi con a, p(a, A) è il numero di fatti precondizione di a che non sono supportati, e g(A) é il numero di nodi goal di A che non sono supportati.

Dato un problema di pianificazione è semplice verificare che il valore di questa funzione è nullo per qualsiasi piano valido. Questa funzione può essere usata anche nel processo di ricerca ad ogni passo per discriminare tra possibili differenti modifiche del grafo, scegliendo quella che minimizza la funzione obiettivo.

2.2.1. Euristica di ricerca

L'utilizzo della funzione obiettivo per guidare la ricerca locale può essere efficiente per alcuni problemi, ma può condurre la ricerca in un minimo locale dal quale non si riesce ad uscire. Per questa ragione, invece di usare la funzione obiettivo, si utilizza una funzione di costo azione F. Questa funzione definisce il costo di inserimento ($F([a], \pi)^i$) e di rimozione ($F([a], \pi)^r$) di una azione $[a]$ nel sottopiano π rappresentato dal sottografo d'azione corrente A. $F([a], \pi)$ è definito nel seguente modo:

$$\begin{cases} F([a], \pi)^i = \alpha^i \cdot p(a, A) + \beta^i \cdot me(a, A) + \gamma^i \cdot unsup(a, A) \\ F([a], \pi)^r = \alpha^r \cdot p(a, A) + \beta^r \cdot me(a, A) + \gamma^r \cdot sup(a, A) \end{cases}$$

dove $me(a,A)$ e $p(a,A)$ sono definite nella Definizione 3, $unsup(a,A)$ è il numero di fatti precondizione non supportati in A che diventano supportati aggiungendo a ad A, e $sup(a, A)$ è il numero di fatti in A che non vengono più supportati rimuovendo a da A.

Scegliendo appropriatamente i coefficienti α, β e γ si possono implementare varie euristiche mirate a rendere la ricerca meno suscettibile ai minimi locali, e la rendeno meno dipendente dal gradiente della funzione obiettivo. Tali valori vengono fissati prima della ricerca e devono soddisfare le seguenti limitazioni:

$$F^i : \alpha^i, \beta^i > 0 \quad \gamma^i \leq 0$$

$$F^{ir} : \alpha^r, \beta^r > 0 \quad \gamma^r \leq 0$$

Si noti che i coefficienti positivi di F (α^i, β^i e γ^r) determinano un incremento di F che è collegato ad un incremento del numero delle violazioni dei vincoli. Analogamente, i coefficienti non positivi di F (α^r, β^r e γ^i) determinano un decremento di F che è collegato ad un decremento del numero di violazioni dei vincoli.

2.2.2. Risultati sperimentali

Il metodo di ricerca locale per l'adattamento di piani è stato testato nel contesto di due domini che sono difficilmente risolvibili per IPP: Logistic e Rocket [1, 10]. I risultati sperimentali mostrano che adattare un piano utilizzando le tecniche di ricerca locale può essere molto più efficiente che generare di un nuovo piano. Le tabelle 1 e 2 mostrano i tempi di CPU per adattare un piano utilizzando la ricerca locale (Walkplan, Tabuplan, e T-Walkplan), e per rigenerarlo (IPP). In particolare la tabella 1 fornisce i tempi di adattamento e generazione per alcune modifiche di Logistic_a e Logistic_b, due problemi del dominio Logistic; mentre la tabella 2 riguarda i risultati di alcune modifiche di Rocket_a e Rocket_b, due problemi del dominio Rocket, che ha molte somiglianze con il dominio Logistic.

Nel mondo di Logistic ci sono più città ed ogni città ha molti luoghi (per esempio, ufficio postale e aeroporto). Alcuni camion possono essere utilizzati per trasportare pacchi all'interno della stessa città, e alcuni aereoplani possono essere utilizzati per trasportare i pacchi tra città differenti. L'obiettivo tipico di un problema di pianificazione in questo dominio consiste nel consegnare dei pacchi in alcuni luoghi. I problemi Init_1-5 corrispondono a cinque modifiche di Logistic_a ottenute cambiando un fatto nello stato iniziale; tale cambiamento impedisce l'applicabilità di alcune azioni pianificate. I problemi Goal_1-11 sono modifiche di Logistic_a che corrispondono a cambiamenti (significativi) a fatti dello stato finale. Per esempio, nel problema Init_1 viene modificata la posizione iniziale del "package1" dal luogo "pgh-po" al luogo "la-airport", che richiede molti cambiamenti rispetto al piano originale; nel problema Goal_5 viene cambiata la posizione finale del "package1" da "bos-po" a "la-po", che richiede il cambiamento di almeno quattro azioni nel piano. In generale GPG ha prestazioni più efficienti rispetto ad IPP, in particolare se si utilizza il metodo T-Walkplan per guidare la ricerca. Tuttavia, i risultati di questi esperimenti sono stati ottenuti mediante il settaggio empirico di alcuni parametri della funzione di valutazione (per esempio, la lunghezza della tabu list per Tabuplan o il "noise factor" per Walkplan). Valori differenti per questi parametri possono dare prestazioni differenti. In più, bisogna notare che ognuno dei problemi modificati presi in considerazione in questi esperimenti ammette un piano con lo stesso numero di livelli del piano risolvente il problema originale.

Problema	graph	graph	Walkplan	Tabuplan	T-Walkplan	IPP	Blackbox
Rocket_a	7	0.46	48.57	1.16	0.5	126.67	5.77
Rocket_b	7	0.49	6.5	1.4	2.78	334.51	8.23
Logistic_a	11	1.58	2.5	1.77	1.04	2329.06	4.0
Logistic_b	13	1.15	19.9	85.43	5.25	1033.75	12.83
Logistic_c	13	2.22	19.4	37.63	7.05	> 24 ore	20.91
BwLarge_a	12	0.3	4.04	123	13.1	0.38	705

Tabella 1: Risultati ottenuti su una SUN ultra 300Mhz, specificando il numero di livelli della soluzione.

Problema	Walkplan	Tabuplan	T-Walkplan	IPP
Rocket_a	21.919	22.253	17.727	38.910
Rocket_b	26.219	30.564	18.165	108.9
Logistic_a	31.09	84.659	43.716	672.5
Logistic_b	59.289	107.899	80.227	311.99
Logistic_c	90.418	177.478	127.378	> 6 ore
BwLarge_a	0.126	0.123	0.131	0.13

Tabella 2: Risultati ottenuti su un PC Intel Pentium II a 450 Mhz.

2.3. Tecniche di ricerca locale per l'adattamento di piani

2.3.1. Introduzione

In questa sezione presentiamo i risultati ottenuti dall'analisi delle tecniche di ricerca locale per l'adattamento di piani nell'ambito di alcuni domini noti (Rocket_a, Rocket_b, Logistic_a, Logistic_b, Logistic_c, BwLarge_a e BwLarge_b); tale analisi è volta a determinare, se esistono, dei valori di default per i parametri della funzione euristica utilizzata per guidare la ricerca.

2.3.2. Descrizione delle tecniche testate

Gli algoritmi utilizzati nella fase di ricerca locale per la modifica del piano sono tre:

- Walkplan
- Tabuplan
- T-walkplan

I tre algoritmi si differenziano, come già detto, per i valori assegnati ai parametri della funzione euristica di GPG che definisce il costo d'inserimento o rimozione di un'azione [a] in un piano parziale π rappresentato dal sottografo corrente A. Tale funzione euristica utilizza sei parametri per determinare il peso delle modifiche causate dall'introduzione o rimozione dell'azione [a]; in particolare:

- per l'introduzione di una nuova azione sottografo d'azione si utilizzano i parametri
  - α^i per il numero di precondizioni di [a] non supportate ($\alpha^i > 0$)
  - β^i per il numero di nodi azione di **A** mutuamente esclusivi con [a] ($\beta^i > 0$)
  - γ^i per il numero di precondizioni non supportate di **A** che diventano supportate introducendo [a] ($\gamma^i < 0$)

  da cui l'espressione della funzione euristica usata per l'inserimento
  $F([a], \pi)^i = \alpha^i \cdot pre(a, \mathbf{A}) + \beta^i \cdot me(a, \mathbf{A}) + \gamma^i \cdot unsup(a, \mathbf{A})$;

- per la rimozione di un'azione presente nel sottografo d'azione si utilizzano i parametri
  - α^r per il numero di precondizioni di [a] non supportate ($\alpha^r < 0$)
  - β^r per il numero di nodi azione di A mutuamente esclusivi con [a] ($\beta^r < 0$)
  - γ^r per il numero di precondizioni di A supportate che diventano non supportate rimuovendo [a] ($\gamma^r > 0$)

  da cui l'espressione della funzione euristica usata per la rimozione
  $F([a], \pi)^r = \alpha^r \cdot pre(a, \mathbf{A}) + \beta^r \cdot me(a, \mathbf{A}) + \gamma^r \cdot sup(a, \mathbf{A})$.

Walkplan minimizza la funzione euristica con le seguenti restrizioni:
$\alpha^i > 0$, $\beta^i > 0$, $\gamma^i = 0$, $\alpha^r = 0$, $\beta^r = 0$, $\gamma^r > 0$
Si vuole in questo modo minimizzare il numero di violazioni dei vincoli introdotte dalla modifica del grafo. L'algoritmo che ne deriva tende a rimanere intrappolato nei minimi locali della funzione euristica, e quindi si introduce un fattore di rumore (parametro N) che determina con probabilità N/100 la scelta casuale per gli elementi del vicinato e con probabilità 1-N/100 la scelta migliore che minimizza la funzione euristica degli elementi del vicinato.

I parametri sono scelti come segue:
$\alpha^i = 1,\ \beta^i = 1,\ \gamma^i = 0,\ \alpha^r = 0,\ \beta^r = 0,\ \gamma^r > 1.$

Tabuplan utilizza una lista contenente le modifiche apportate negli ultimi k passi (parametro L) in termini di inserimento o rimozione di azioni. Ad ogni passo della ricerca si sceglie un nuovo sottografo che possa essere generato da quello attuale effettuando modifiche che non annullino quelle contenute nella tabu list e che ovviamente minimizzino la funzione euristica. In questo caso è fondamentale il parametro L che rappresenta la lunghezza della tabu list, oltre ai parametri della funzione euristica che sono scelti come segue:
$\alpha^i = 1,\ \beta^i = 1,\ \gamma^i < 0,\ \alpha^r < 0,\ \beta^r < 0,\ \gamma^r = 1.$

T-walkplan usa una tabu list per aumentare il costo delle modifiche da effettuare se esse sono presenti nella tabu list. In questo metodo, oltre al parametro L che rappresenta la lunghezza della tabu list, si utilizza un fattore di rumore N per ovviare al problema dei minimi locali; i parametri della funzione euristica sono scelti come segue:
$\alpha^i = 1,\ \beta^i = 1,\ \gamma^i < 0,\ \alpha^r < 0,\ \beta^r < 0,\ \gamma^r = 1.$

Adattamento automatico dei parametri: è stata apportata una modifica al pianificatore affinché adatti i parametri L (lunghezza della tabu list) e N (noise) nel seguente modo:
- N e L assumono inizialmente i valori specificati dall'utente (o quelli di default se non vengono specificati);
- durante l'esecuzione del programma ad ogni passo di rimozione o inserimento di un'azione nel grafo viene chiamata una procedura che memorizza in un vettore di lunghezza K predefinita (abbiamo usato K = 100) il numero di precondizioni non soddisfatte;
- quando il vettore è completo, l'aggiornamento dei parametri può avvenire con uno dei seguenti metodi (vedi appendice A per il listato delle procedure relative):
  - viene calcolata la varianza dei valori contenuti nel vettore; se tale varianza è minore di una soglia S (nei test è stato usato S = 3) L e N vengono moltiplicati per un fattore M (abbiamo usato M = 3), altrimenti L e N vengono ripristinati ai valori originariamente specificati dall'utente;
  - viene calcolato il minimo dei valori contenuti nel vettore; il minimo attuale viene confrontato con il minimo calcolato al passo precedente (ciò non viene fatto nella prima iterazione e nella iterazione immediatamente successiva ad ogni variazione dei parametri, in quanto si vogliono confrontare valori di minimo calcolati a parità di valore dei parametri); se il minimo attuale è minore di quello precedente i valori dei parametri vengono ripristinati ai valori originariamente specificati dall'utente, altrimenti vengono moltiplicati per un fattore 2 (ma ciò può avvenire al massimo per 2 volte).
- una volta effettuato l'aggiornamento dei parametri, il vettore viene azzerato e la procedura si ripete.

2.3.3. Descrizione dei domini utilizzati

Per ognuno dei domini utilizzati per i test sono stati utilizzate:
- la soluzione del problema base tramite GPG (contenuta in un file .sol);
- 10 varianti del problema base ottenute modificando alcuni fatti nello stato iniziale (b2v1-10.fct);
- 10 varianti del problema base ottenute modificando alcuni goal nello stato finale (a3v1-10.fct);
- 10 varianti del problema base ottenute aggiungendo alcuni goal nello stato finale (a1v1-10.fct).

Le descrizioni dettagliate degli operatori domini, dei problemi base delle varianti sono riportate in appendice B.

2.3.4. Risultati sperimentali

I dati sperimentali ottenuti sono stati analizzati tramite tabelle e grafici di Excel, ottenuti automaticamente mediante delle macro in VBA (Visual Basic for Applications).

In appendice C sono riportati a titolo d'esempio alcuni di questi grafici; nel CD-ROM allegato si possono trovare tutti i domini di lavoro completi delle varianti ai problemi e dei risultati ottenuti (si rimanda al file readme.txt nella directory principale del CD-ROM).

2.3.4.1. Walkplan

Dipende da N (noise = probabilità di scegliere una mossa in maniera casuale) e da γ^r.

- Per N si nota una dipendenza variabile a seconda del dominio e in minore misura anche a seconda del particolare problema considerato:
  - per Logistic valori medio-alti (da 20 a 55)
  - per Rocket valori medio-piccoli (da 15 a 20)
  - per BwLarge valori piccoli (minori di 15), ma in BLOCKS WORLD A si ha variabilità a seconda del problema considerato

  Questa dipendenza sembra scomparire nei tests del metodo d'adattamento dei parametri basato sulla varianza della funzione costo, dove raramente si riscontrano valori migliori di altri. Nei tests del metodo di adattamento dei parametri basato sul minimo della funzione costo per tutti i domini si nota una dipendenza da N (buoni valori da 10 a 15).
- Per γ^r valgono considerazioni analoghe:
  - per Logistic valori piccoli (da 1.5 a 1.9)
  - per Rocket valori grandi (da 2.2 a 2.5)
  - per BwLarge si ha variabilità a seconda del problema considerato.

  Questa dipendenza viene mantenuta nei tests degli algoritmi di adattamento automatico dei parametri.

Il metodo di adattamento dei parametri basato sul minimo della funzione costo sembra funzionare meglio del metodo basato sulla varianza per Rocket_b, dove si ha un numero maggiore di problemi risolti ad ogni tentativo. I due metodi sembrano invece equivalenti per Rocket_a, Logistic_a e Logistic_b; per Logistic_c il metodo del minimo sembra funzionare in modo peggiore, perché si ha un numero minore di problemi sempre risolti.

2.3.4.2. Tabuplan

Dipende da L (lunghezza della tabu list) e da γ^i, α^r e β^r che vengono fatti variare insieme.

- Per L si notano su tutti i domini buone prestazioni per valori medio-piccoli (da 5 a 20); per Logistic sono migliori valori medi (da 15 a 25). Anche per i metodi d'adattamento automatico dei parametri è opportuno assegnare valori iniziali bassi (L=5).
- Per γ^i, α^r, β^r si rileva una minore dipendenza dei risultati dal valore assegnato; dai tests dei metodi di adattamento automatico dei parametri sembrano adeguati valori alti in modulo (da -0.8 a -1.0).

Il metodo d'adattamento dei parametri basato sul minimo della funzione costo sembra funzionare meglio del metodo basato sulla varianza per Rocket_b, dove si ha un numero maggiore

di problemi risolti ad ogni tentativo. I due metodi sembrano invece equivalenti per Rocket_a, Logistic_a e Logistic_b; per Logistic_c il metodo del minimo funziona in modo peggiore.

2.3.4.3. T-Walkplan

Dipende da L, N e γ^i, α^r e β^r che vengono fatti variare insieme.

- Per L si riscontrano buone prestazioni su tutti i domini per valori non piccoli (> 15), ma questa dipendenza sparisce nei tests del metodo d'adattamento basato sulla varianza della funzione costo; nei tests del metodo di adattamento basato sul minimo della funzione costo sembrano buoni valori piccoli (5 e 10, anche se il valore 10 non è adatto per Logistic_c).
- Per N si nota in generale una scarsa dipendenza, in quanto si ha variabilità dei valori ottimali sia al variare del dominio sia del problema considerato; per Logistic e Rocket sembrano però migliori valori grandi. Anche per N la dipendenza sparisce nei tests del metodo d'adattamento basato sulla varianza della funzione costo e sembrano buoni valori piccoli per il metodo di adattamento basato sul minimo della funzione costo (N=5).
- Per γ^i, α^r, β^r occorre distinguere a seconda del dominio: per Rocket e Logistic sono buoni valori piccoli in modulo (da -0.3 a 0), mentre per BwLarge si ha una dipendenza minore e variabile (soprattutto in BwLarge_b). Per il metodo d'adattamento basato sul minimo della funzione costo, si nota un comportamento migliore su tutti i domini per valori grandi in modulo di questi parametri (-1.0).

Il metodo d'adattamento dei parametri basato sul minimo della funzione costo sembra funzionare meglio del metodo basato sulla varianza per Rocket_a, Logistic_b e Logistic_c, dove si ha un numero maggiore di problemi risolti ad ogni tentativo. I due metodi sembrano equivalenti per Rocket_b e Logistic_a.

2.4. Adattamento di piani mediante ricerca sistematica

2.4.1. Modifica e ottimizzazione sistematica di piani

Dato un piano *P* per un problema di pianificazione P e un problema modificato P', che differisce da P per qualche fatto iniziale o finale, il processo di modifica e ottimizzazione sistematica del piano di GPG consiste di tre parti principali:

(1) identificazione delle inconsistenze che sono presenti in *P* rispetto a P';
(2) revisione delle inconsistenze di *P* per eliminarle e fornire un piano valido *P'* per P';
(3) ottimizzazione di *P'* per produrre un piano più compatto, cioè un piano che utilizza un numero inferiore di livelli nel Planning Graph.

(1) e (2) sono realizzati con l'algoritmo ADJUST-PLAN descritto nelle figure 4 e 5. L'algoritmo in primo luogo identifica il primo livello i di *P* che contiene una inconsistenza (passo 1). Si noti che, dal momento che stiamo considerando un dominio STRIPS, questo può essere realizzato in tempo polinomiale facendo una simulazione di *P* (in modo simile i fatti che sono necessariamente veri ad ogni livello possono essere determinati in un tempo polinomiale).

Algoritmo: ADJUST-PLAN
Input: un piano *P* che contiene alcune inconsistenze `max-adjust-time` tempo limite di CPU.
Output: o un piano corretto o `fallimento`.

1. Si identifica il primo livello i in *P* che contiene una inconsistenza; se non c'è tale livello, allora ritorna *P*;
2. Se i è l'ultimo livello di *P*, allora pone `init-level` a i-1 e `goal-level` ad i, altrimenti si pone `init-level` ad i e `goal-level` a i+1;
3. Mentre il tempo di CPU≤ `max-adjust-time`
    4. Si ripianifica sistematicamente utilizzando come fatti iniziali *F* (`init-level`) e come fatti finali *G* (`goal-level`), dove *F* (`init-level`) è un insieme di fatti che sono verificati al livello `init-level`, e *G* (`goal-level`) è un insieme di precondizioni delle azioni in *P* al livello `goal-level` (includendo le `no-op`);
    5. Se non c'è nessun piano da *F* (`init-level`) a *G* (`goal-level`), oppure si supera il limite di ricerca, allora si decrementa `init-level` o incrementa `goal-level` (cioè, si allarga la finestra di ripianificazione), altrimenti si inserisce il (sotto)piano trovato in *P*, si rimuovono le azioni esistenti tra `init-level` e `goal-level` e si torna ad 1.
6. Ritorna `fallimento`.

Figura 4: descrizione dell'algoritmo ADJUST-PLAN usato da GPG.

Quindi ADJUST-PLAN prova a correggere le inconsistenze contenute al livello i ripianificando dal livello i al livello i+1 utilizzando la ricerca sistematica basata su backtracking. Se non esiste un piano oppure viene superato un fissato limite di ricerca, la finestra di ripianificazione viene allargata (per esempio, si ripianifica dal livello i-1 al livello i+1). Il processo viene iterato fino a che non viene trovato un (sotto)piano, oppure la ricerca ha raggiunto un tempo limite massimo di CPU predefinito (max-adjust-time).

In linea di principio, se max-adjust-time ha un valore abbastanza grande, il passo 5 di adjust-plan può incrementare la finestra di ripianificazione fino a raggiungere i livelli iniziale e finale del Planning Graph originale. Questo determinerebbe una ricerca completa. Perciò questo metodo di adattamento dei piani è completo, nel senso che con un alto valore di max-adjust-time, se il problema modificato è risolvibile, allora il metodo trova un piano, altrimenti il metodo rileva che il problema non è risolvibile.

La ripianificazione sistematica al passo 4 è realizzata prima costruendo il corrispondente Planning Graph, e poi cercando il piano utilizzando la stessa tecnica di backtracking di IPP [13]. Questa ricerca sistematica garantisce che se viene trovato un (sotto)piano, allora esso è ottimo rispetto al numero di livelli in questione [1, 13]. L'appropriato inserimento in *P* del sottopiano trovato al passo 4 non invalida il resto del piano. Al contrario, tale sottopiano può essere utile per raggiungere delle precondizioni non raggiungibili presenti nei livelli successivi rispetto a quello in cui era iniziato il processo al passo 1. Una volta che l'algoritmo ha corretto le inconsistenze in un livello, esso procede considerando il prossimo livello contenente una inconsistenza (se esiste).

Il processo di ottimizzazione iterativa presentato in figura 4 produce una successione di piani migliorati, ognuno dei quali coinvolge un numero minore di livelli rispetto al precedente. Maggiore è il tempo dedicato a questo processo e migliore sarà il piano determinato. In questo senso il processo può essere visto come un "anytime process", ossia può essere interrotto in ogni momento per dare il piano migliore calcolato fino a quel punto. Questa ottimizzazione può essere utile sia per i problemi di adattamento on-line che per i problemi off-line. Nel caso on-line il piano calcolato da ADJUST-PLAN può essere utilizzato per un'identificazione veloce della prossima azione da eseguire, mentre il processo d'ottimizzazione potrebbe essere eseguito in background per migliorare la parte successiva del piano.

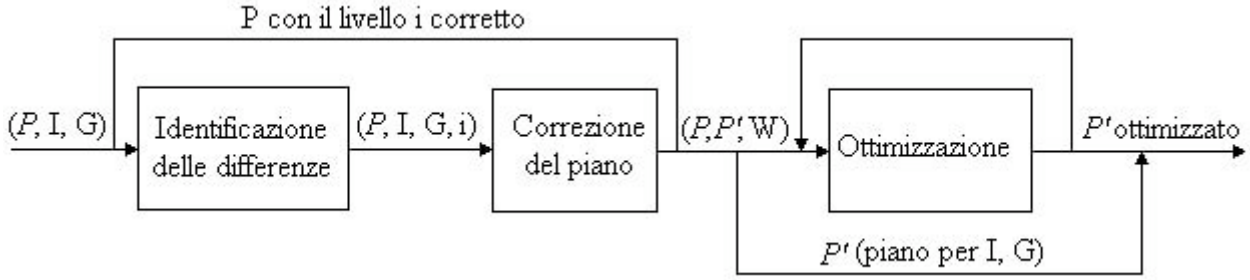


Figura 5: il processo di adattamento sistematico e di ottimizzazione del piano di GPG.

Il modo in cui viene calcolato un piano ottimizzato è basato sulla considerazione di finestre di ripianificazione alternative per i sottopiani che sono stati precedentemente inseriti o da ADJUST-PLAN durante il calcolo del primo piano adattato o dalla precedente ottimizzazione per ottenere un piano più compatto. Per ognuno di questi sottopiani si considera una finestra di ripianificazione più ampia rispetto a quello che era stata usata per produrlo. Se si trova un sottopiano che coinvolge un numero minore di livelli rispetto alla dimensione della finestra di ripianifiazione allora si rimpiazza il corrispondente sottopiano nel piano corrente con quello nuovo. Per esempio, se si è ripianificato da F(i) a G(j), ottenendo un sottopiano con k livelli, allora si ripianifica da F(i') a G(j') con j'>j e i'<=i, e si considerano solo i piani che utilizzano un numero di livelli minore di (j' – j) + k + (i – i'). Se non c'è nessun piano da F(i') a G(j'), allora si diminuisce i' o si aumenta j' (cioè, si allarga la finestra di ripianificazione), altrimenti si inserisce il sottopiano in *P*, e si considera il prossimo sottopiano che sarà inserito da ADJUST-PLAN o dalla precedente ottimizzazione. Ogni ottimizzazione di un piano produce un piano migliorato, e viene iterata fino a quando non si è ripianificato da F(i') = I (lo stato iniziale) a G(j') = G (lo stato finale), che garantisce di aver trovato il piano ottimo.

2.4.2. Risultati sperimentali

Le tabelle 3, 4 e 5 danno i tempi di CPU per adattare un piano e per rigenerarlo (IPP) per molte varianti dei problemi base di Rocket, Logistic e Gripper rispettivamente, che sono stati usati per testare i pianificatori basati sul Planning Graph [1, 10, 12]. In particolare, la tabella 3 mostra i tempi di adattamento e di generazione per Logistic_a e Logistic_b, la tabella 4 per Rocket_a e Rocket_b, e la tabella 5 per Gripper-10 e Gripper-12 come formalizzati in IPP (due problemi in un semplice dominio dove un robot dotato di due braccia deve spostare varie palline da alcune stanze in altre). Le varianti dei problemi che sono state considerate per gripper riguardano la posizione iniziale e finale delle palline, ed è stata utilizzata una stanza aggiuntiva. Le varianti Init sono problemi ottenuti modificando alcuni fatti nello stato iniziale del problema originale; le varianti Goal sono problemi ottenuti modificando alcuni obiettivi nello stato finale; le varianti IG sono problemi ottenuti modificando entrambi gli stati iniziale e finale del problema originale.

Per ogni variante considerata le tabelle presentano: i secondi di CPU richiesti da ADJUST-PLAN per trovare il primo piano valido (T1), e i secondi di CPU utilizzati per trovare un secondo piano ottimizzato (T2); il numero di livelli del primo piano (L1) e del secondo (L2); i secondi di CPU richiesti da IPP per risolvere le varianti da zero, e il numero ottimale di livelli richiesti (L). I risultati delle tabelle mostrano che adattare un piano utilizzando questo metodo è molto più veloce che effettuare una ripianificazione completa (fino a 4 ordini di grandezza). Inoltre molto spesso il primo piano che viene trovato dal processo di ottimizzazione è calcolato abbastanza efficientemente (vedere la colonna T2 delle tabelle). D'altronde, questi primi due piani possono richiedere più livelli rispetto all'ottimo. Infine il processo d'ottimizzazione trova un piano ottimo, ma il tempo di CPU può essere molto più alto rispetto al tempo di CPU richiesto per il calcolo del primo piano adattato.

Roc_a	ADJUST-PLAN				IPP		Roc_b	ADJUST-PLAN				IPP	
	Time1	L1	T2	L2	Time	L		Time	L1	T2	L2	Time	L
Goal_1	0.09	7	18.06	7	74.26	7	Goal_1	0.09	7	18.1	7	48.57	7
Goal_2	0.09	8	0.18	7	99.47	7	Goal_2	0.09	8	7.67	7	26.12	7
Goal_3	0.25	10	0.61	9	53.06	7	Goal_3	0.11	9	0.24	8	163.26	7
Goal_4	0.16	9	0.69	8	---	---	Goal_4	0.12	9	0.24	8	---	---
Goal_5	0.11	9	0.23	8	53.69	7	Goal_5	0.15	9	0.28	7	187.53	7
Goal_6	0.11	9	0.4	8	53.61	7	Goal_6	0.63	10	0.85	9	162.42	7
Goal_7	0.11	9	0.3	8	---	---	Goal_7	0.14	8	0.25	7	---	---
Goal_8	0.24	10	0.38	7	73.7	7	Goal_8	0.33	10	0.49	8	---	---
Goal_9	0.31	10	3.34	9	179.42	7	Goal_9	0.23	10	0.44	9	140.38	7
Goal_10	0.1	9	0.22	8	67.34	7	Goal_10	0.63	10	0.82	9	---	---
Goal_11	0.09	9	0.39	8	72.01	7	Goal_11	0.17	10	0.31	8	140.23	7
Goal_12	0.09	9	0.2	8	83.45	7	Goal_12	0.1	9	0.21	8	113.25	7
Goal_13	0.35	10	0.47	7	54.05	7	Goal_13	0.25	10	0.39	8	63.99	7
Goal_14	0.17	10	0.34	9	37.64	7	Goal_14	0.12	8	0.22	7	88	7
Goal_15	0.1	9	0.34	8	22.54	7	Goal_15	0.1	9	0.27	8	246.94	7
Goal_16	0.1	9	0.21	8	75.47	7	Goal_16	0.1	9	0.27	8	---	---
Goal_17	0.13	8	0.22	7	86.82	7	Goal_17	0.12	8	0.22	7	91.51	7
Goal_18	0.1	9	0.35	8	32.27	7	Goal_18	0.1	9	0.21	8	203.74	7
Goal_19	0.09	9	0.2	8	71.29	7	Goal_19	0.36	10	0.55	9	95.95	7
Goal_20	0.1	9	0.21	8	42.36	7	Goal_20	0.12	8	0.22	7	8.68	6
Init_1	0.14	8	0.25	7	30.66	7	Init_1	0.41	7	7.59	7	85.38	6
Init_2	0.15	9	0.28	8	11.53	7	Init_2	0.25	10	1.01	8	110.7	7
Init_3	0.15	9	0.27	8	31.17	7	Init_3	0.18	9	0.79	8	84.31	7
Init_3	0.15	8	0.26	7	30.52	7	Init_4	0.18	9	1.32	8	223.29	7
Init_4	0.41	7	7.52	7	228.35	7	Init_5	0.17	8	0.32	7	19.28	7
Init_5	0.16	9	0.32	8	30.36	7	Init_5	0.11	9	0.22	7	82.48	7
Init_6	0.4	7	7.36	7	225.44	7	Init_6	0.41	7	7.45	7	---	---
Init_7	0.18	9	0.34	7	1.7	6	Init_7	0.18	9	1.46	8	201.09	7
Init_8	0.18	9	0.38	8	1.72	6	Init_8	0.17	8	0.32	7	90.93	7
							Init_10	0.18	9	0.35	7	77.26	7

Tabella 3: tempi di adattamento e generazione per il dominio Rocket.

Log_a	ADJUST-PLAN				IPP		Log_b	ADJUST-PLAN				IPP	
	Time1	L1	T2	L2	Time	L		Time1	L1	T2	L2	Time	L
Goal_1	0.23	11	18.73	11	29.8	11	Goal_1	0.42	16	0.77	15	4544.06	13
Goal_2	0.22	11	282.6	11	362.6	11	Goal_2	0.63	18	1.1	17	887.65	13
Goal_3	0.43	14	0.96	13	---	---	Goal_3	0.45	17	0.83	16	4409.32	13
Goal_4	0.96	16	1.48	15	1430.8	11	Goal_4	0.85	18	1.36	17	1109.98	13
Goal_5	0.49	15	0.89	14	3845.2	11	Goal_5	0.43	16	0.89	15	4423.25	13
Goal_6	0.45	14	0.82	13	1560.7	11	Goal_6	0.84	18	1.36	17	1108.71	13
Goal_7	0.96	16	1.49	15	1752.7	11	Goal_7	0.47	17	0.88	16	1243.47	13
Goal_8	0.27	13	0.53	12	4686.0	11	Goal_8	0.35	15	0.65	14	5447.56	13
Goal_9	0.28	13	0.54	12	4931.7	11	Goal_9	0.36	16	0.64	15	5.49	11
Goal_10	0.5	15	0.91	14	3712.1	11	Goal_10	0.35	16	0.63	15	5.15	11
Goal_11	0.36	13	0.63	12	1438.6	11	Goal_11	0.52	18	0.89	16	3.58	11
Goal_12	0.63	14	0.94	12	1216.4	11	Goal_12	0.34	15	0.59	13	5.33	11
Goal_13	0.38	14	0.66	12	116.3	11	Goal_13	0.47	16	0.79	15	0.72	11
Goal_14	0.26	12	0.47	11	193.8	11	Goal_14	0.36	16	0.64	15	5.14	11
Goal_15	0.27	13	0.5	11	208.3	11	Goal_15	0.53	18	0.89	16	1.21	11
Goal_16	0.62	14	0.93	12	775.3	11	Goal_16	0.34	15	0.58	13	2.86	11
Goal_17	217.9	15	223.6	13	282.1	11	Goal_17	0.53	18	0.9	16	1.94	11
Goal_18	0.58	14	0.89	12	193.4	11	Goal_18	0.25	14	0.83	13	149.23	13
Goal_18	0.77	14	1.09	12	428.0	11	Init_1	0.45	15	1.52	14	1.09	11
Goal_20	0.23	12	308.7	11	248.8	11	Init_2	0.93	18	160.57	13	129.27	13
Init_1	0.43	13	4.66	12	692.0	11	Init_3	0.61	17	4.44	16	0.73	11
Init_2	0.92	16	65.5	11	107.9	11	Init_4	0.46	15	0.72	13	2.85	11
Init_3	0.53	14	0.86	11	505.1	11	Init_5	0.56	17	1.3	16	233.01	13
Init_4	0.73	15	1.15	13	513.9	11	Init_6	0.56	17	1.29	16	3.56	11
Init_5	0.33	13	0.6	12	600.3	11	Init_7	0.57	17	1.3	16	248.21	13
Init_6	0.55	15	0.98	14	510.7	11	Init_8	0.53	17	1.21	16	3.59	11
Init_7	0.57	15	4.36	14	513.9	11	Init_9	0.64	16	0.92	13	25.21	13
Init_8	0.51	15	5.64	11	1306.0	11	Init_10	0.5	17	0.89	16	239.19	13
Init_9	0.58	14	0.87	11	13.1	11	IG_1	4.79	20	62.27	17	73.11	15
Init_10	0.67	15	51.06	14	98.2	11	IG_2	2.65	20	60.96	17	73.02	15
IG_1	0.68	17	14.44	14	233.0	13	IG_3	2.87	25	75.98	17	50.8	15
IG_2	0.68	17	14.26	14	324.1	13	IG_4	5.00	25	190.86	19	56.05	15
IG_3	1.21	19	14.24	14	128.3	13	IG_5	0.94	15	n.d.	n.d.	4.14	12
IG_4	0.96	17	460.3	14	---	---							
IG_5	521.5	18	73	n.d.	n.d.	12							

Tabella 4: tempi di adattamento e generazione per il dominio Logistic.

Gr_10	ADJUST-PLAN				IPP		Gr_12	ADJUST-PLAN				IPP	
	Time	L1	T2	L2	Time	L		Time	L1	T2	L2	Time	L
Goal_1	0.23	22	1384	20	1006	20	Goal_1	0.34	27	055	26	381	18
Goal_2	0.23	22	>600	---	---	---	Goal_2	0.34	26	514	25	---	---
Goal_3	0.24	23	>600	---	---	---	Goal_3	0.53	25	>600	---	---	---
Goal_4	0.22	22	>600	---	---	---	Goal_4	0.35	26	>600	---	---	---
Goal_5	0.23	20	>600	---	488	17	Goal_5	0.32	28	051	24	914	15
Goal_6	0.23	22	>600	---	---	---	Goal_6	0.48	25	4548	24	---	---
Goal_7	1258	23	>600	---	---	---	Goal_7	0.34	26	>600	---	1007	10
Goal_8	0.22	22	>600	---	---	---	Init_1	50.1	17	629	15	---	---
Init_1	1.01	23	126	22	677	18	Init_2	3.28	29	>600	---	---	---
Init_2	1.78	25	>600	---	---	---	Init_3	3.54	29	>600	---	---	---
Init_3	1.83	25	626	23	---	---							

Tabella 5: tempi di adattamento e generazione per il dominio Gripper.

2.5. Adattamento di piani mediante ricerca locale e sistematica

Come mostrato nel paragrafo precedente, il metodo di ricerca locale è particolarmente efficiente quando il problema modificato ammette una soluzione che non richiede un numero di livelli maggiore rispetto al piano originale. Il metodo di ricerca sistematica riesce efficientemente a risolvere problemi di adattamento che richiedono un più alto numero di livelli, ma può generare piani adattati che richiedono molti più livelli rispetto a quelli necessari (specialmente quando esiste una soluzione con lo stesso numero di livelli rispetto al piano originale). In questo paragrafo proponiamo un terzo metodo d'adattamento, esso unisce la ricerca locale e sistematica in modo da combinarne i vantaggi ed ovviare ai loro limiti. Questo metodo inizialmente prova a adattare il piano utilizzando le tecniche di ricerca locale con parametri di default per la funzione euristica. Se non viene trovata una soluzione (un piano adattato valido) entro un tempo limite predefinito di CPU, allora

- se il piano originale contiene un numero di inconsistenze minore di un numero predefinito *f*, si adatta il piano originale utilizzando ADJUST-PLAN;
- altrimenti la ricerca locale continua finchè non viene trovata una *quasi-soluzione*, cioè finchè la ricerca locale raggiunge un sottografo di azioni che rappresenta un piano con al più *f* inconsistenze. Questo piano viene poi adattato utilizzando ADJUST-PLAN.

ADJUST-PLAN può essere eseguito finché non trova una soluzione oppure finché rileva la non adattabilità del piano. In alternativa, possiamo imporre ad ADJUST-PLAN un tempo limite di CPU (max-adjust-time), dopo il quale ripetere il passo di ricerca locale, sperando di trovare un'altra quasi-soluzione più facile da adattare per ADJUST-PLAN.

Abbiamo testato questo metodo per adattare un piano utilizzando le stesse varianti considerate nei paragrafi precedenti, con in più molte varianti del problema Logistic_c, un altro difficile problema nel dominio Logistic, e Bw_large_b, un problema della formalizzazione blocks world introdotto in [10] (le varianti I riguardano lo stato iniziale, mentre le varianti G riguardano lo stato finale). I risultati sono riportati nelle tabelle 6, 7 8 e 9 ed evidenziano l'efficienza di quest'approccio.

In ogni tabella la seconda colonna indica il tempo totale di CPU mediato su dieci prove richiesto dal metodo combinato; la terza indica il tempo medio dedicato alla ricerca locale (T-L); la quarta il tempo medio dedicato ad ADJUST-PLAN (T-A) mediato sul numero di volte in cui è stato utilizzato tale algoritmo (indicato tra parentesi); la quinta il numero medio di livelli richiesti (LEV); la sesta e la settima rispettivamente il tempo e i livelli (L) richiesti da IPP.

Roc_a	T-Walkplan+ADJUST				IPP		Roc_b	T-Walkplan+ADJUST				IPP	
	T	T-L	T-A	L	T	L		T	T-L	T-A	L	T	L
G_1	0.09	0.09	0	7	74.26	7	G_1	0.10	0.10	0	7	48.57	7
G_2	0.09	0.09	0	7	99.47	7	G_2	0.47	0.37	0.09(10)	8	26.12	7
G_3	0.13	0.13	0	7	53.06	7	G_3	0.14	0.14	0	7	163.2	7
G_4	0.15	0.15	0	7	---	---	G_4	0.48	0.38	0.11(9)	8.8	---	---
G_5	0.13	0.13	0	7	53.69	7	G_5	0.12	0.12	0	7	187.5	7
G_6	0.10	0.10	0	7	53.61	7	G_6	0.12	0.12	0	7	162.4	7
G_7	0.38	0.31	0.10(7)	8.4	---	---	G_7	0.11	0.11	0	7	---	---
G_8	0.11	0.11	0	7	73.7	7	G_8	0.28	0.21	0.32(2)	7.6	---	---
G_9	0.68	0.37	0.31(10)	10	179.4	7	G_9	0.11	0.11	0	7	140.38	7
G_10	0.11	0.11	0	7	67.34	7	G_10	0.27	0.21	0.62(1)	7.3	---	---
G_11	0.42	0.33	0.09(9)	8.8	72.01	7	G_11	0.54	0.37	0.16(10)	10	140.2	7
G_12	0.11	0.11	0	7	83.45	7	G_12	0.10	0.10	0	7	113.2	7
G_13	0.09	0.09	0	7	54.05	7	G_13	0.10	0.10	0	7	63.99	7
G_14	0.10	0.10	0	7	37.64	7	G_14	0.10	0.10	0	7	88	7
G_15	0.27	0.24	0.09(3)	7.6	22.54	7	G_15	0.47	0.37	0.1(10)	9	246.94	7
G_16	0.45	0.35	0.09(10)	9	75.47	7	G_16	0.16	0.16	0	7	---	---
G_17	0.09	0.09	0	7	86.82	7	G_17	0.10	0.10	0	7	91.51	7
G_18	0.10	0.10	0	7	32.27	7	G_18	0.45	0.36	0.1(9)	8.8	203.7	7
G_19	0.44	0.35	0.09(10)	9	71.29	7	G_19	0.34	0.23	0.36(3)	7.9	95.95	7
G_20	0.13	0.13	0	7	42.36	7	G_20	0.10	0.10	0	7	8.68	6
I_1	0.10	0.10	0	7	30.66	7	I_1	0.11	0.11	0	7	85.38	7
I_2	0.10	0.10	0	7	31.17	7	I_2	0.65	0.57	0.11(7)	8.2	110.7	7
I_3	0.09	0.09	0	7	30.52	7	I_3	0.12	0.12	0	7	84.31	7
I_4	0.35	0.29	0.097(6)	8.2	228.3	7	I_4	0.28	0.23	0.18(3)	7.9	223.2	7
I_5	0.11	0.11	0	7	30.36	7	I_5	0.10	0.10	0	7	19.28	7
I_6	0.37	0.30	0.1(7)	8.4	225.4	7	I_6	0.20	0.17	0.18(2)	7.6	---	---
I_7	0.11	0.11	0	7	1.7	6	I_7	0.31	0.24	0.16(4)	8.2	201.0	7
I_8	0.09	0.09	0	7	1.72	6	I_8	0.10	0.10	0	7	90.93	7
							I_10	0.11	0.11	0	7	77.26	7

Tabella 6: tempo di CPU e numero di livelli richiesti da GPG (prima soluzione) e da IPP per alcune modifiche di Rocket_A e Rocket_B. --- indica che IPP non ha trovato la soluzione per "out of memory".

Gr_10	Walkplan+ADJUST				IPP		Gr_12	Walkplan+ADJUST				IPP	
	T	T-L	T-A	L	T	L		T	T-L	T-A	L	T	L
G_1	0.51	0.36	0.14(10)	22	100.6	20	G_1	0.68	0.48	0.20(10)	26	381	18
G_2	0.78	0.55	0.22(10)	22	---	---	G_2	1.11	0.78	0.33(10)	26	---	---
G_3	0.96	0.54	0.41(10)	20.8	---	---	G_3	1.30	0.77	0.53(10)	27	---	---
G_4	0.78	0.56	0.22	22	---	---	G_4	1.12	0.78	0.33(10)	26	---	---
G_5	0.51	0.36	0.16(9)	18.1	48.8	17	G_5	0.66	0.47	0.20(9)	23.9	91.4	15
G_6	0.67	0.50	0.28(6)	18.8	---	---	G_6	1.14	0.79	0.34(10)	26	---	---
G_7	0.97	0.55	0.42(10)	19.1	---	---	G_7	1.33	0.78	0.55(10)	25	---	---
G_8	0.64	0.51	0.21(6)	20.8	---	---	G_8	1.08	0.77	0.33(9)	25.7	10.07	10
I_1	0.81	0.38	0.43(10)	22	67.7	18	I_1	1.00	0.48	0.64(8)	24.7	---	---
I_2	2.50	0.57	1.93(10)	25	---	---	I_2	4.06	0.82	3.23(10)	29	---	---
I_3	2.37	0.58	1.79(10)	25	---	---	I_3	0.78	0.78	0	23	---	---
G_12	0.73	0.36	0.61(6)	12.8	1216	11	I_4	4.32	0.82	3.49(10)	29	---	---

Tabella 7: tempo di CPU e numero di livelli richiesti da GPG (prima soluzione) e da IPP per alcune modifiche di Gripper_10 e Gripper_12. --- indica che IPP non ha trovato la soluzione per "out of memory".

Log_a	T-Walkplan+ADJUST				IPP		Log_b	T-Walkplan+ADJUST				IPP	
	Time	T-L	T-A	Lev	Time	L		Time	T-L	T-A	Lev	Time	L
G_1	0.17	0.17	0	11	29.89	11	G_1	0.43	0.34	0.42(2)	13.6	4544	13
G_2	0.17	0.17	0	11	362.6	11	G_2	0.37	0.31	0.61(1)	13.5	887.6	13
G_3	0.22	0.22	0	11	---	---	G_3	0.37	0.32	0.44(1)	13.4	4409	13
G_4	0.19	0.19	0	11	1430	11	G_4	0.27	0.27	0	13	1109	13
G_5	0.20	0.20	0	11	3845	11	G_5	0.36	0.32	0.43(1)	13.3	4423	13
G_6	0.55	0.33	0.45(5)	12.5	1560	11	G_6	0.27	0.27	0	13	1108	13
G_7	0.35	0.25	0.96(1)	11.5	1752	11	G_7	0.64	0.41	0.46(5)	15	1243	13
G_8	0.20	0.20	0	11	4686	11	G_8	0.61	0.43	0.35(5)	14	5447	13
G_9	0.22	0.22	0	11	4931	11	G_9	0.32	0.29	0.36(1)	13.3	5.49	11
G_10	0.66	0.37	0.49(6)	13.4	3712	11	G_10	0.72	0.43	0.35(8)	15.4	5.15	11
G_11	0.61	0.37	0.34(7)	12.4	1438	11	G_11	0.97	0.45	0.52(10)	18	3.58	11
G_12	0.73	0.36	0.61(6)	12.8	1216	11	G_12	0.25	0.25	0	13	5.33	11
G_13	0.19	0.19	0	11	116.3	11	G_13	0.25	0.25	0	13	0.72	11
G_14	0.17	0.17	0	11	193.8	11	G_14	0.70	0.42	0.35(8)	15.4	5.14	11
G_15	0.19	0.19	0	11	208.3	11	G_15	0.91	0.44	0.52(9)	17.5	1.21	11
G_16	1.00	0.40	0.60(10)	14	775.3	11	G_16	0.24	0.24	0	13	2.86	11
G_17	0.24	0.24	0	11	282.1	11	G_17	0.83	0.42	0.51(8)	17	1.94	11
G_18	0.60	0.32	0.57(5)	12.5	193.4	11	G_18	0.69	0.45	0.24(10)	14	149.2	13
G_18	1.15	0.39	0.75(10)	14	428.0	11	I_1	0.56	0.37	0.39(5)	15	1.09	11
G_20	0.62	0.39	0.22(10)	12	248.8	11	I_2	1.14	0.44	0.70(10)	18	129.2	13
I_1	0.30	0.26	0.38(1)	11.3	692	11	I_3	0.23	0.23	0	13	0.73	11
I_2	1.12	0.39	0.73(10)	16	107.9	11	I_4	0.57	0.38	0.38(5)	15	2.85	11
I_3	0.66	0.35	0.44(7)	13.8	505.1	11	I_5	0.45	0.32	0.67(2)	14	233.0	13
I_4	0.18	0.18	0	11	513.9	11	I_6	1.12	0.45	0.67(10)	18	3.56	11
I_5	0.18	0.18	0	11	600.3	11	I_7	1.13	0.45	0.68(10)	18	248.2	13
I_6	0.92	0.36	0.79(7)	14.5	510.7	11	I_8	1.08	0.45	0.63(10)	18	3.59	11
I_7	0.91	0.37	0.76(7)	14.5	513.9	11	I_9	2.17	0.43	1.74(10)	19	25.21	13
I_8	0.22	0.19	0.23(1)	11.2	1306	11	I_10	0.85	0.39	0.66(7)	16.5	239.1	13
I_9	0.26	0.24	0.2(1)	11.1	13.18	11	IG_1	2.28	0.38	1.89(10)	19	73.11	15
I_10	0.17	0.17	0	11	98.23	11	IG_2	11.2	0.38	10.8(10)	19	73.02	15
IG_1	0.93	0.39	0.54(10)	16	233.0	13	IG_3	0.38	0.38	0	13	50.8	13
IG_2	0.95	0.39	0.55(10)	16	324.1	13	IG_4	1.93	1.67	0.25(10)	17.1	56.05	15
IG_3	1.08	0.39	0.69(10)	15	128.3	13	IG_5	9.46	0.38	9.08(10)	19	4.14	13
IG_4	1.27	0.47	0.80(10)	16	---	---							
IG_5	10.2	9.05	1.26(10)	17.3	n.d.	13							

Tabella 8: tempo di CPU e numero di livelli richiesti da GPG (prima soluzione) e da IPP per alcune modifiche di Logistic_A e Logistic_B. --- indica che IPP non ha trovato la soluzione per "out of memory".

Log_c	T-Walkplan+ADJUST				IPP	
	Time	T-L	T-A	Lev	Time	L
G_1	0.341	0.341	0	13	---	---
G_2	0.406	0.363	0.43(1)	13.3	---	---
G_3	0.549	0.436	1.13(1)	13.3	---	---
G_4	0.718	0.441	0.923(3)	14.5	---	---
G_5	0.551	0.427	0.62(2)	13.8	---	---
G_6	0.373	0.373	0	13	---	---
G_7	0.472	0.414	0.58(1)	13.3	---	---
G_8	0.387	0.387	0	13	---	---
G_9	1.002	0.552	0.643(7)	15.8	---	---
G_10	0.81	0.515	0.492(6)	14.8	---	---
G_11	0.877	0.499	0.63(6)	14.8	1474.8	11
G_12	0.957	0.501	0.651(7)	15.1	511.9	11
G_13	0.452	0.373	0.79(1)	13.5	1870.9	11
G_14	0.625	0.437	0.47(4)	13.8	611.54	11
G_15	0.419	0.348	0.71(1)	13.3	159.57	11
G_16	0.836	0.458	0.63(6)	14.8	644.29	11
G_17	0.687	0.456	0.77(3)	14.5	306.84	11
G_18	0.423	0.376	0.47(1)	13.2	422.57	11
G_19	0.946	0.478	0.78(6)	16	863.61	11
G_20	0.946	0.541	0.405(10)	16	---	---
I_1	0.727	0.449	0.556(5)	15	279.75	11
I_2	3.026	0.541	2.485(10)	18	---	---
I_2	0.489	0.384	1.05(1)	13.5	158.52	11
I_3	0.947	0.516	0.539(8)	16.2	611.98	11
I_4	1.6	0.547	1.053(10)	18	---	---
I_6	0.881	0.455	1.065(4)	15	1888.9	11
I_7	1.564	0.546	1.018(10)	18	---	---
I_8	1.375	0.522	0.948(9)	17.5	1894.3	11
I_9	16.631	0.839	15.79(10)	16.2	12103	13
I_10	1.068	0.468	1(6)	16	---	---
IG_1	12.714	11.06	1.654(10)	18.8	---	---
IG_2	7.981	7.535	0.446(10)	18.4	6963	15
IG_3	0.372	0.332	0.4(1)	13.3	---	---
IG_4	3.681	3.198	0.483(10)	17.5	7414	15
IG_5	10.2	9.05	1.26(10)	17.3	n.d.	13

Bwl_b	T-Walkplan+ADJUST				IPP	
	Time	T-L	T-A	Lev	Time	L
G_1	3.88	1.47	2.4(10)	24.0	62.57	21
G_2	2.63	1.86	0.72(10)	20.0	36.77	19
G_3	3.7	1.86	1.84(10)	24.0	44.1	19
G_4	24.05	1.82	22.23(10)	26.4	162.7	21
G_5	2.96	1.8	1.15(10)	20.0	92.98	19
G_6	66.05	22.5	43.48(10)	24.6	38.94	19
I_1	2.4	1.42	0.98(10)	20.0	34.92	19
I_2	2.18	1.30	1.40(6)	20.4	5.71	17
I_3	8.14	2.28	5.85(10)	22.2	4.72	17
I_4	19.86	1.94	17.92(10)	26.2	29.51	19
IG_1	2.15	1.45	0.7(10)	22.0	24.67	19
IG_2	2.76	1.42	1.33(10)	24.0	23.07	21
IG_3	4.33	1.43	2.9(10)	26.0	27.98	19

Tabella 9: tempo di CPU e numero di livelli richiesti da GPG (prima soluzione) e da IPP per alcune modifiche di Logistic_C e BwLarge_B. --- indica che IPP non ha trovato la soluzione per “out of memory”.

3. DESCRIZIONE DI UN DOMINIO DI LAVORO: IL ROBOT GUARDIANO

3.1. Introduzione

Il dominio di lavoro da noi ipotizzato per un robot guardiano è composto da un robot che si muove in un ambiente formato da corridoi e stanze aventi le seguenti caratteristiche:

- il mondo è composto da: corridoi, ascensori e stanze dotate di porte, luci e finestre;
- esiste un sistema di allarme fisso composto da: telecamere per i corridoi e sensori di movimento per le stanze; questi sensori sono attivabili e disattivabili singolarmente tramite un calcolatore con cui il robot comunica via ponte radio;
- esiste un sistema antincendio con sensori dislocati nei corridoi e nelle stanze;
- esiste un posto di supervisione o centro di controllo con un operatore che controlla le immagini inviate dalle telecamere fisse o dal sistema di visione del robot, ed eventuali segnali di allarme inviati dai sensori di movimento delle stanze; il centro di controllo può interagire con il robot tramite il calcolatore;
- il robot dispone di: sistema di visione, sirena d'allarme, possibilità di aprire le stanze tramite chiave elettronica, di controllare presenza ed integrità di oggetti, di accendere le luci e di chiamare gli ascensori, di riconoscere persone tramite dispositivo opportuno (ad esempio tessera magnetica nominativa) e dispone di un ponte radio per la comunicazione con il calcolatore.

Un esempio grafico di un possibile mondo è presentato in figura 6:

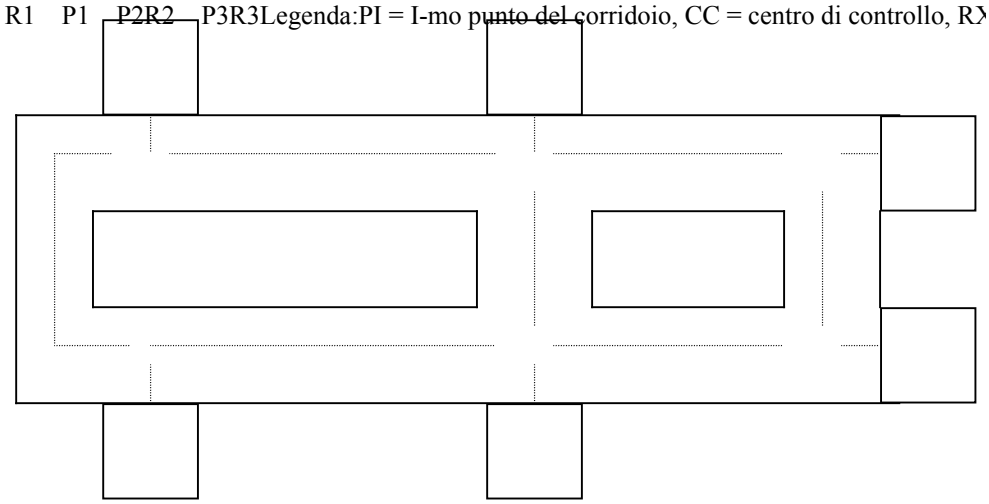


Figura 6: esempio di mondo per il dominio del robot guardiano.

All'interno di questo mondo, il robot si muove seguendo un piano predefinito di controllo durante il quale percorre i corridoi ed entra nelle stanze per controllarle ad una ad una. Ciò accade sia nel caso in cui la porta sia aperta sia nel caso in cui sia chiusa; questo perché se una porta è aperta si potrebbe avere la presenza di un intruso, mentre se la porta è chiusa si rafforza il controllo dei sensori con quello del robot.

Durante l'esecuzione del piano predefinito, il robot deve mandare in tempo reale un rapporto del suo operato al calcolatore, controllare lo stato della batteria e reagire ad imprevisti di vario genere quali ad esempio l'incontro di persone autorizzate o non, l'esecuzione di un ordine impartito dal centro di controllo o l'attivazione della sirena d'allarme in caso di pericolo.

Quando il robot rileva un imprevisto, modifica la propria base di conoscenza aggiungendo o togliendo uno o più fatti , oppure aggiunge nuovi obiettivi. Quindi è necessario modificare il piano predefinito inserito in memoria attraverso gli algoritmi d'adattamento sopra descritti.

La descrizione del mondo è stata suddivisa in due domini separati, il primo dedicato alla navigazione nei corridoi dei vari piani dell'edificio e il secondo dedicato alla navigazione nelle stanze e al controllo degli oggetti in esse contenuti. Questa scelta è stata dettata dall'esigenza di rendere il dominio modulare, modellabile e trattabile.

Specifichiamo di seguito per ognuno dei due domini le azioni e i piani del robot, gli oggetti del dominio e le relazioni e i vincoli che intercorrono tra essi.

3.2. Il dominio dei corridoi

3.2.1. Azioni base del robot

Le azioni principali che il robot deve saper compiere in questo dominio sono:

- muoversi da un punto ad un altro dei corridoi;
- entrare nelle stanze e controllarle (viene indicato un controllo generico, che verrà poi dettagliato nel dominio delle stanze);
- sbloccare porte bloccate tramite un pass-partout;
- utilizzare l'ascensore;
- interagire con l'impianto fisso di allarme e con il centro di controllo;
- interagire con persone autorizzate e non;
- controllare l'operatività delle telecamere fisse dei corridoi;
- controllare i sensori antincendio.

3.2.2. Oggetti del dominio

Il dominio è costituito da oggetti che vengono specificati mediante le seguenti variabili:

- `location`, che rappresenta un punto del corridoio;
- `room`, che rappresenta una stanza;
- `pass-partout`, che rappresenta le chiavi per aprire le stanze bloccate;
- `lift`, che rappresenta un ascensore;
- `robot`;
- `fire`, che rappresenta un sensore antincendio;
- `camera`, che rappresenta una telecamera fissa dei corridoi.

3.2.3. Operatori per l'interazione con gli oggetti del dominio

Gli operatori sono stati suddivisi, a seconda del tipo di interazione che descrivono, in due distinte categorie, quella dell'interazione tra robot e mondo e quella dell'interazione tra robot, allarmi e centro di controllo. Ogni operatore ha la seguente struttura:

`nome_operatore`	
`:v ?nome_variabile tipo_variabile`	(elenco delle variabili utilizzate dall'operatore)
`:p fatto1 fatto2`	(elenco di fatti che devono essere verificati per l'applicabilità dell'operatore, chiamati precondizioni)
`:e ADD fatto1 fatto2`	(elenco di fatti che vengono aggiunti alla base di conoscenza dopo aver applicato l'operatore, chiamati effetti additivi)
`DEL fatto1 fatto2.`	(elenco di fatti che vengono tolti dalla base di conoscenza dopo aver applicato l'operatore, chiamati effetti cancellanti)

I fatti utilizzati dagli operatori del dominio sono:

`at(?r ?s)`	il robot ?r si trova nel punto ?s
`con(?s ?a)`	il punto ?s è connesso con il punto ?a
`has-fuel(?r)`	il robot ?r è rifornito di carburante
`in-front-of(?p ?l)`	il punto ?p si trova di fronte all'ascensore ?l
`closed(?l ?p)`	l'ascensore ?l nel punto ?p è chiuso
`open(?l ?s)`	l'ascensore ?l nel punto ?s è aperto
`sensor-off(?p)`	il sensore della stanza ?p è disattivo
`sensor-on(?p)`	il sensore della stanza ?p è attivo
`open(?p)`	la porta della stanza ?p è aperta
`closed(?p)`	la porta della stanza ?p è chiusa
`locked(?p)`	la porta della stanza ?p è bloccata
`not-locked(?p)`	la posta della stanza ?p non è bloccata
`has-pass-partout(?r ?u)`	il robot ?r ha il pass-partout ?u

Gli operatori del dominio sono:

1) Interazione robot-mondo
   a) `location`: il robot si muove tra due punti interconnessi con l'operatore `move` e si rifornisce di carburante con l'operatore `refuel`:

```
move
:v ?r robot ?s location ?a location
:p at(?r ?s) con(?s ?a) has-fuel(?r)
:e ADD at(?r ?a)
   DEL at(?r ?s).

refuel
:v ?r robot
:p at(?r LP1)
:e ADD has-fuel(?r).
```

   Nota: l'operatore `refuel` non viene mai utilizzato nelle nostre simulazioni, perché si suppone che il problema della ricarica del robot venga risolto da un opportuno agente.

b) `lift`: il robot interagisce con l'ascensore in due modi: lo chiama con l'operatore `call-lift` (se la porta è chiusa) e prende l'ascensore con l'operatore `take-lift` (si suppone che l'ascensore chiuda la porta automaticamente dopo l'uscita del robot):

```
call-lift
:v ?r robot ?p location ?l lift
:p at(?r ?p) in-front-of(?p ?l) closed(?l ?p)
:e ADD open(?l ?p)
   DEL closed(?l ?p).

take-lift
:v ?r robot ?s location ?a location ?l lift
:p at(?r ?s) in-front-of(?s ?l) in-front-of(?a ?l) open(?l ?s) con-
lift(?s ?a)
:e ADD at(?r ?a) closed(?l ?s)
   DEL at(?r ?s) open(?l ?s).
```

c) `room`: il robot interagisce con le stanze in molti modi: apre e chiude la porta della stanza con gli operatori `open-room` e `close-room`, entra nella stanza con l'operatore `enter-room`, esce dalla stanza con l'operatore `exit-room` e controlla la stanza con l'operatore `control-room` (quest'ultimo operatore viene dettagliato nel dominio delle stanze). Nel caso una porta sia bloccata e non si possa aprire con l'operatore `open-room`, è previsto l'utilizzo dell'operatore `unlock-room`. Tale operatore utilizza un pass-partout che si trova in una locazione apposita e che viene recuperato dal robot guardiano mediante l'operatore `take-pass-partout` e riposto in tale locazione mediante l'operatore `putdown-pass-partout`:

```
open-room
:v ?r robot ?l location ?p room
:p at(?r ?l) in-front-of(?l ?p) closed(?p) sensor-off(?p) not-
locked(?p)
:e ADD open(?p)
   DEL closed(?p).

close-room
:v ?r robot ?l location ?p room
:p at(?r ?l) in-front-of(?l ?p) open(?p)
:e ADD closed(?p)
   DEL open(?p).

enter-room
:v ?r robot ?l location ?p room
:p at(?r ?l) in-front-of(?l ?p) open(?p)
:e ADD in(?r ?p)
   DEL at(?r ?l).

exit-room
:v ?r robot ?l location ?p room
:p in(?r ?p) in-front-of(?l ?p) open(?p)
:e ADD at(?r ?l)
   DEL in(?r ?p).

control-room
:v ?r robot ?p room
:p in(?r ?p)
```

```
:e ADD controlled(?p).

unlock-room
:v ?r robot ?l location ?p room ?u pass-partout
:p at(?r ?l) in-front-of(?l ?p) locked(?p) has-pass-partout(?r ?u)
:e ADD not-locked(?p)
   DEL locked(?p).

take-pass-partout
:v ?r robot ?p location ?u pass-partout
:p at(?r ?p) at(?u ?p)
:e ADD has-pass-partout(?r ?u)
   DEL at(?u ?p).

putdown-pass-partout
:v ?r robot ?p location ?u pass-partout
:p at(?r ?p) has-pass-partout(?r ?u)
:e ADD at(?u ?p)
   DEL has-pass-partout(?r ?u).
```

2) Interazione robot-allarme-centro di controllo: il robot può comunicare con il computer tramite un ponte radio; ciò può essere utile in molte situazioni:
   a) quando il robot entra in una stanza per controllarla richiederà che il calcolatore disattivi il sensore di movimento della stanza quando entra e lo riattivi quando esce tramite gli operatori `sensor-off` e `sensor-on`:

```
sensor-off
:v ?r robot ?l location ?p room
:p at(?r ?l) in-front-of(?l ?p) closed(?p)
:e ADD sensor-off(?p)
   DEL sensor-on(?p).

sensor-on
:v ?r robot ?l location ?p room
:p at(?r ?l) in-front-of(?l ?p) closed(?p)
:e ADD sensor-on(?p)
   DEL sensor-off(?p).
```

   b) quando il centro di controllo vuole impartire ordini al robot (ad esempio controllare una stanza specifica);
   c) per trasmettere il rapporto dettagliato del suo operato in tempo reale;
   d) inoltre il robot deve poter attivare il proprio allarme tramite opportuni operatori e comunicare l'evento anomalo al centro di controllo (è previsto un operatore per ogni tipo di situazione anomala, si veda oltre);
   e) quando il centro di controllo ordina al robot guardiano di controllare una telecamera fissa di un corridoio o un sensore antincendio esso lo farà mediante gli operatori `control-fire` e `control-camera`:

```
control-fire
:v ?r robot ?l location ?f fire
:p at(?r ?l) at(?f ?l)
```

```
:e ADD controlled(?f).

control-camera
:v ?r robot ?l location ?c camera
:p at(?r ?l) at(?c ?l)
:e ADD controlled(?c).
```

Gli operatori appena descritti sono contenuti nel file "guard.ops", riportato nel CD-ROM allegato.

3.2.4. Eventi imprevisti

Durante lo svolgimento dei propri compiti, il robot guardiano può incorrere in imprevisti di varia natura. Classifichiamo questi imprevisti di due categorie:

1) imprevisti **non critici**, che si suppone siano risolvibili autonomamente dal robot (al più comunicando con il centro di controllo per ottenere informazioni necessarie); ad esempio:
   a) l'incontro di un oggetto o altro ingombro sul percorso;
   b) l'ordine da parte del centro di controllo di eseguire una specifica azione che esula dal piano in esecuzione (ad esempio il controllo di una telecamera, di un sensore antincendio o di una stanza solitamente ignorata);
   c) la presenza di una porta bloccata;
2) imprevisti **critici**, che non possono essere affrontati autonomamente dal robot ma che richiedono l'intervento di un operatore umano; ad esempio:
   a) l'incontro inaspettato con un intruso (cioè di una persona che non si fa identificare dal robot tramite una tessera magnetica appositamente prevista).

Ci siamo focalizzati, per i test degli algoritmi di adattamento, sugli imprevisti non critici, dei quali abbiamo presentato più sopra gli operatori; gli operatori per trattare gli imprevisti critici sono riportati nel paragrafo 3.4., nel quale viene presentata la formalizzazione di un dominio reale.

In presenza di un imprevisto, il robot deve interrompere il piano attuale e ripianificare le proprie azioni in modo tale da superare l'imprevisto e riprendere l'esecuzione del piano originario.

Nel caso d'incontro con un intruso, il robot deve interrompere il piano che stava eseguendo, andare verso la persona ed avviare la procedura di riconoscimento. Ovviamente in caso di mancato riconoscimento o di fuga dell'intruso il robot deve suonare l'allarme ed eventualmente cercare di seguirlo.

Nel caso di un ingombro sul percorso, il robot deve ripianificare il tracciato da seguire in modo da evitare l'ingombro e portare ugualmente a termine il piano che stava eseguendo.

Nel caso di un ordine da parte del centro di controllo, il robot deve interrompere il piano che stava eseguendo, pianificare un percorso che lo porti all'esecuzione dell'ordine ricevuto e, dopo averlo eseguito, riprendere il piano precedente dal punto d'interruzione o meglio riprenderne la parte non ancora eseguita e ripianificarla in modo che possa essere svolta dal luogo in cui il robot si trova.

Nel caso di una porta bloccata abbiamo ipotizzato due possibilità:

- è disponibile un pass-partout comune a tutti i robot; nel caso una porta sia bloccata un robot deve recarsi nella locazione in cui si trova il pass-partout, prelevarlo e riportarlo a posto dopo averlo utilizzato;
- ogni porta è dotata di un dispositivo elettronico; quando una porta è bloccata il robot agisce su tale dispositivo sbloccandola.

3.2.5. Esempi di domini

Abbiamo ipotizzato due possibili scenari per il dominio dei corridoi del robot guardiano denominandoli “guard1” e “guard2”; diamo di seguito una descrizione di questi due domini.

1) **guard1:** il mondo può essere descritto graficamente come in figura 7:

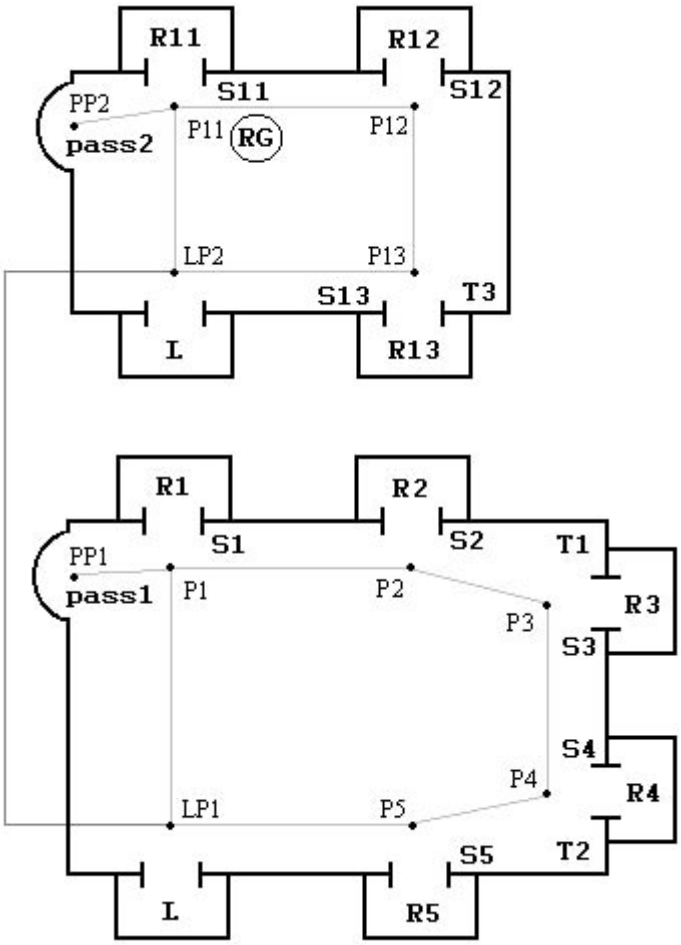


Figura 7: descrizione grafica del problema guard1.

Il mondo è costituito da due piani collegati tramite l’ascensore L. Il primo piano è costituito da cinque stanze (R1 R2 R3 R4 R5), l’ascensore (L), sei punti di collegamento del corridoio (P1 P2 P3 P4 P5 LP1), un punto in cui viene custodito il pass-partout (PP1), due telecamere fisse (T1 e T2) e 5 sensori antincendio (S1 S2 S3 S4 S5). Il secondo piano è costituito da tre stanze (R11 R12 R13), l’ascensore (L), quattro punti di collegamento del corridoio (P11 P12 P13 LP2), un punto in cui viene custodito il pass-partout (PP2), una telecamera fissa (T3) e tre sensori antincendio (S11 S12 S13).

I goal del problema consistono nel controllare tutte le stanze, supposte in uno stato normale (porte chiuse e non bloccate, sensori attivi) percorrendo dal punto P11 i corridoi, supposti anch’essi in uno stato normale (sensori antincendio spenti, telecamere funzionanti, pass-partout nella posizione apposita, connessioni bidirezionali tra i punti come in figura) e posizionandosi alla fine in LP1 dopo aver chiuso tutte le porte e riattivato tutti i sensori delle stanze; tale problema viene descritto nel file “guard1.fct” riportato di seguito:

```
# guard1.fct: ambiente con due piani e un ascensore
```

```
location: P1 P2 P3 P4 P5 P11 P12 P13 LP1 LP2 PP1 PP2;
room: R1 R2 R3 R4 R5 R11 R12 R13;
robot: RG;
lift: L;
pass-partout: PASS1 PASS2;
fire: S1 S2 S3 S4 S5 S11 S12 S13 SL1 SL2;
camera: T1 T2 T3;

initial:

at(RG P11) has-fuel(RG)

con(P1 P2) con(P2 P1) con(P2 P3) con(P3 P2) con(P3 P4) con(P4 P3)
con(P4 P5) con(P5 P4)
con(P5 LP1) con(LP1 P5)  con(P1 LP1) con(LP1 P1)  con(PP1 P1) con(P1
PP1) at(PASS1 PP1)

in-front-of(P1 R1) sensor-on(R1) closed(R1) not-locked(R1)
in-front-of(P2 R2) sensor-on(R2) closed(R2) not-locked(R2)
in-front-of(P3 R3) sensor-on(R3) closed(R3) not-locked(R3)
in-front-of(P4 R4) sensor-on(R4) closed(R4) not-locked(R4)
in-front-of(P5 R5) sensor-on(R5) closed(R5) not-locked(R5)

con-lift(LP1 LP2)
con-lift(LP2 LP1)
in-front-of(LP1 L) closed(L LP1)
in-front-of(LP2 L) closed(L LP2)

con(LP2 P11) con(P11 LP2) con(P11 P12) con(P12 P11) con(P12 P13)
con(P13 P12) con(P13 LP2) con(LP2 P13)
con(PP2 P11) con(P2 PP11) at(PASS2 PP2)

in-front-of(P11 R11) sensor-on(R11) closed(R11) not-locked(R11)
in-front-of(P12 R12) sensor-on(R12) closed(R12) not-locked(R12)
in-front-of(P13 R13) sensor-on(R13) closed(R13) not-locked(R13)

at(S1 P1) at(S2 P2) at(S3 P3) at(S4 P4) at(S5 P5) at(S11 P11) at(S12
P12) at(S13 P13)
at(T1 P3) at(T2 P4) at(T3 P12);

goal:

at(RG LP1)

controlled(R1)  controlled(R2)  controlled(R3)  controlled(R4)
controlled(R5)
controlled(R11) controlled(R12) controlled(R13)

sensor-on(R1) sensor-on(R2) sensor-on(R3) sensor-on(R4) sensor-on(R5)
sensor-on(R11) sensor-on(R12)
sensor-on(R13)

at(PASS1 PP1) at(PASS2 PP2);
```

La soluzione del problema, che viene memorizzata dal robot come piano predefinito nel file “guard1.sol”, è composta da 67 livelli:

```
0: sensor-off_RG_P11_R11
1: open-room_RG_P11_R11
2: enter-room_RG_P11_R11
3: control-room_RG_R11
4: exit-room_RG_P11_R11
5: close-room_RG_P11_R11
6: sensor-on_RG_P11_R11
7: move_RG_P11_P12
8: sensor-off_RG_P12_R12
9: open-room_RG_P12_R12
10: enter-room_RG_P12_R12
11: control-room_RG_R12
12: exit-room_RG_P12_R12
13: close-room_RG_P12_R12
14: sensor-on_RG_P12_R12
15: move_RG_P12_P13
16: sensor-off_RG_P13_R13
17: open-room_RG_P13_R13
18: enter-room_RG_P13_R13
19: control-room_RG_R13
20: exit-room_RG_P13_R13
21: close-room_RG_P13_R13
22: sensor-on_RG_P13_R13
23: move_RG_P13_LP2
24: call-lift_RG_LP2_L
25: take-lift_RG_LP2_LP1_L
26: move_RG_LP1_P1
27: sensor-off_RG_P1_R1
28: open-room_RG_P1_R1
29: enter-room_RG_P1_R1
30: control-room_RG_R1
31: exit-room_RG_P1_R1
32: close-room_RG_P1_R1
33: sensor-on_RG_P1_R1
34: move_RG_P1_P2
35: sensor-off_RG_P2_R2
36: open-room_RG_P2_R2
37: enter-room_RG_P2_R2
38: control-room_RG_R2
39: exit-room_RG_P2_R2
40: close-room_RG_P2_R2
41: sensor-on_RG_P2_R2
42: move_RG_P2_P3
43: sensor-off_RG_P3_R3
44: open-room_RG_P3_R3
45: enter-room_RG_P3_R3
46: control-room_RG_R3
47: exit-room_RG_P3_R3
48: close-room_RG_P3_R3
49: sensor-on_RG_P3_R3
50: move_RG_P3_P4
51: sensor-off_RG_P4_R4
52: open-room_RG_P4_R4
53: enter-room_RG_P4_R4
54: control-room_RG_R4
55: exit-room_RG_P4_R4
```

```
56: close-room_RG_P4_R4
57: sensor-on_RG_P4_R4
58: move_RG_P4_P5
59: sensor-off_RG_P5_R5
60: open-room_RG_P5_R5
61: enter-room_RG_P5_R5
62: control-room_RG_R5
63: exit-room_RG_P5_R5
64: close-room_RG_P5_R5
65: sensor-on_RG_P5_R5
66: move_RG_P5_LP1
```

Per i test sono stati utilizzati dei problemi ottenuti da questo problema base mediante aggiunte di goal o modifiche dello stato iniziale che rappresentano i possibili imprevisti che il robot guardiano può dover affrontare; a fronte di ciò sarà necessario modificare il piano predefinito aggiungendo, togliendo o modificando mosse e aggiungendo o togliendo livelli. Ad esempio, si consideri il problema denominato b2v3.fct, in cui viene eliminata la connessione tra P2 e P3 (vedi figura 7) e vengono aggiunte due connessioni, una tra P2 e P4 e una tra P2 e P5: poiché nel piano predefinito il robot si muove tra P2 e P3 nel livello 42, questa mossa non sarà più applicabile e dovrà essere sostituita con due mosse, una da P2 a P4 e una da P4 a P3; ne consegue che per adattare il piano occorrerà modificare la mossa `42: move_RG_P2_P3` in `42: move_RG_P2_P4`, inserire al livello 43 la mossa `43: move_RG_P4_P3` e spostare tutte le mosse successive (43-66) di un livello. Ovviamente questo è solo uno dei possibili modi di adattare il piano: ad esempio, il robot potrebbe muoversi da P2 a P4 nel livello 42, proseguire controllando la stanza R4 (mosse dal livello 51 al 57) e solo successivamente recarsi in P3 per controllare la stanza R3 (mosse da 44 a 49); in questo caso cambierebbe il modo di spostare le mosse dalla 43 alla 66, che dovrebbero essere scambiate anziché fatte slittare semplicemente di un livello.

I problemi sono elencati e descritti di seguito, (a1 indica l'aggiunta di goal, b2 la modifica dello stato iniziale) riportando per ognuno un indice della difficoltà d'adattamento in termini di numero di mosse da aggiungere, togliere o modificare e numero di livelli da aggiungere o togliere al piano (tralasciando il numero di mosse da spostare dal livello in cui si trovano):

- a1v1.fct: ottenuto da guard1.fct aggiungendo una stanza da controllare; per ottenere un piano soluzione a partire da guard1.sol occorre aggiungere 10 mosse in altrettanti livelli (piano risultante di 77 livelli);

- a1v2.fct: ottenuto da guard1.fct chiedendo come goal aggiuntivo di controllare il sensore S2; per ottenere un piano soluzione a partire da guard1.sol occorre aggiungere una mossa, ma non è necessario aggiungere livelli (piano risultante di 67 livelli);

- a1v3.fct: ottenuto da guard1.fct chiedendo come goal aggiuntivo di controllare la telecamera T2; per ottenere un piano soluzione a partire da guard1.sol occorre aggiungere una mossa, ma non è necessario aggiungere livelli (piano risultante di 67 livelli)

- a1v4.fct: ottenuto combinando le variazioni di a1v2.fct e a1v3.fct; per ottenere un piano soluzione a partire da guard1.sol occorre aggiungere 10 mosse in altrettanti livelli (piano risultante di 77 livelli);

- a1v5.fct: ottenuto da guard1.fct chiedendo di controllare tutte le telecamere; per ottenere un piano soluzione a partire da guard1.sol occorre aggiungere tre mosse, ma non è necessario aggiungere livelli (piano risultante di 67 livelli);

- a1v6.fct: ottenuto da guard1.fct aggiungendo un piano con una stanza da controllare; per ottenere un piano soluzione a partire da guard1.sol occorre aggiungere tredici mosse in tredici livelli (piano risultante di 80 livelli);

- a1v7.fct: ottenuto da guard1.fct chiedendo di controllare tutti i sensori del primo piano; per ottenere un piano soluzione a partire da guard1.sol occorre aggiungere cinque mosse senza aggiungere livelli (piano risultante di 67 livelli);

- b2v1.fct: ottenuto da guard1.fct togliendo la connessione tra P2 e P3 e aggiungendone una tra P2 e P4 e una tra P2 e P5; per ottenere un piano soluzione a partire da guard1.sol occorre modificare una mossa presente e aggiungerne nove in altrettanti livelli (piano risultante di 76 livelli);

- b2v2.fct: ottenuto da guard1.fct togliendo la connessione tra LP1 e LP2 (rendendo quindi inutilizzabile l'ascensore L) e aggiungendo un altro ascensore L1 connesso tramite i punti LP2 e LP3 collegati come LP1 e LP2 al resto dell'ambiente; per ottenere un piano soluzione a partire da guard1.sol occorre modificare 4 mosse presenti (piano risultante di 67 livelli);

- b2v3.fct: ottenuto da guard1.fct togliendo la connessione tra P4 e P5 e aggiungendone una tra P2 e P5 e una tra P2 e P4; per ottenere un piano soluzione a partire da guard1.sol occorre modificare una mossa e aggiungerne una in un livello (piano risultante di 68 livelli);

- b2v4.fct: ottenuto da guard1.fct togliendo la connessione tra LP1 e LP2 (rendendo quindi inutilizzabile l'ascensore L) e aggiungendo un altro ascensore L1 connesso tramite i punti LP2 e LP3 collegati in modo diverso da LP1 e LP2 al resto dell'ambiente; per ottenere un piano soluzione a partire da guard1.sol occorre modificare molte mosse, e possono esistere molti modi di adattare il piano;

- b2v5.fct: ottenuto da guard1.fct fct togliendo la connessione tra P4 e P5 e aggiungendone una tra P2 e P5 e una tra P2 e P4 e togliendo la connessione tra P12 e P13 e aggiungendone una tra P11 e P13; per ottenere un piano soluzione a partire da guard1.sol occorre modificare due mosse e aggiungerne 10 in altrettanti livelli (piano risultante di 77 livelli);

- b2v6.fct: ottenuto da guard1.fct modificando lo stato della porta di R3, che è bloccata; per ottenere un piano soluzione a partire da guard1.sol occorre aggiungere 6 mosse in altrettanti livelli (piano risultante di 73 livelli);

- b2v7.fct: ottenuto da guard1.fct modificando lo stato delle porte di R3 e R13, che sono bloccate; per ottenere un piano soluzione a partire da guard1.sol occorre aggiungere 12 mosse in altrettanti livelli (piano risultante di 79 livelli).

- b2v8.fct: ottenuto da guard1.fct modificando lo stato della porta R1, che è bloccata; per ottenere un piano soluzione a partire da guard1.sol occorre aggiungere 6 mosse in altrettanti livelli (piano risultante di 73 livelli);

- b2v9.fct: ottenuto da guard1.fct togliendo la connessione tra P1 e P2; l'adattamento del piano comporta molti cambiamenti, e il comportamento del pianificatore non è prevedibile;
- b2v10.fct: ottenuto da guard1.fct modificando lo stato delle porte di R1 e R5, che sono bloccate; per ottenere un piano soluzione a partire da guard1.sol occorre aggiungere 6 mosse in altrettanti livelli (piano risultante di 73 livelli);

- b2v11.fct: ottenuto da guard1.fct fct togliendo la connessione tra P13 e LP2; l'adattamento del piano comporta molti cambiamenti, e il comportamento del pianificatore non è prevedibile;

- b2v12.fct: ottenuto da guard1.fct spostando la porta della stanza R11 da P11 a P12; per ottenere un piano soluzione a partire da guard1.sol occorre modificare 6 mosse (esiste una soluzione in 61 livelli);

- b2v13.fct: ottenuto da guard1.fct togliendo la connessione tra P5 e LP1 e aggiungendone una tra P1 e P5; per ottenere un piano soluzione a partire da guard1.sol occorre aggiungere due mosse e toglierne una (piano risultante di 68 livelli).

Tutti i file relativi a questo dominio sono riportati nel CD-ROM allegato.

2) **guard2**: il mondo può essere descritto graficamente come in figura 8:

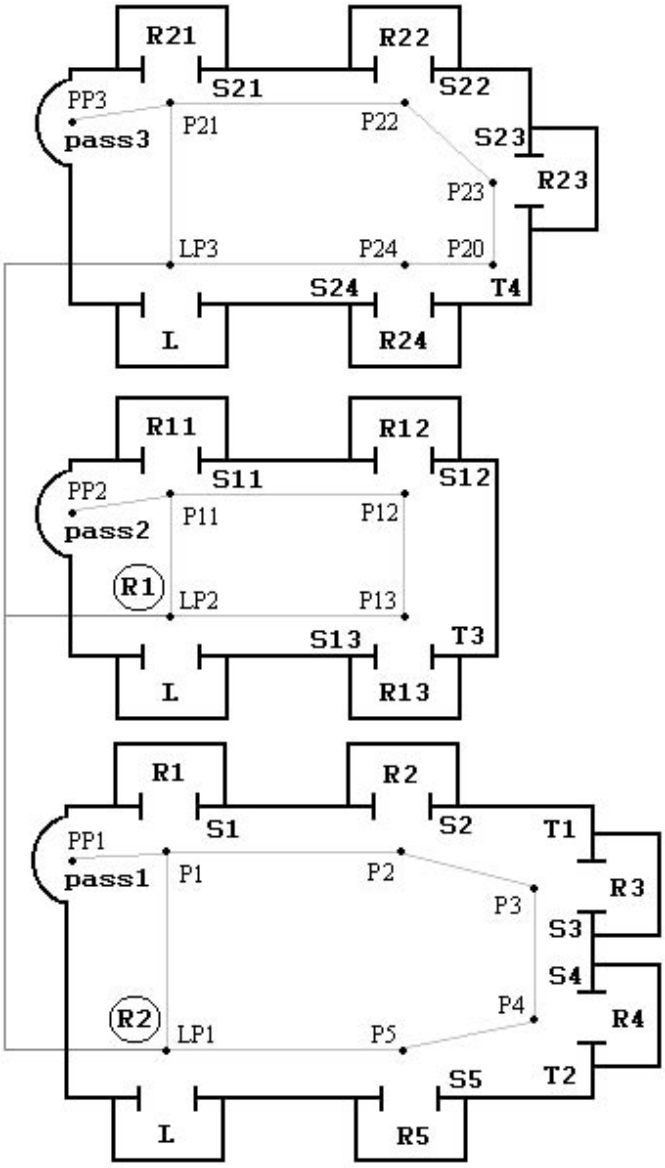


Figura 8: descrizione grafica del problema guard2.

Nel mondo ci sono due robot (R1 R2) in un ambiente costituito da tre piani collegati tramite l'ascensore L. Il primo piano è costituito da cinque stanze (R1 R2 R3 R4 R5), l'ascensore (L), sei punti di collegamento del corridoio (P1 P2 P3 P4 P5 LP1), un punto in cui viene custodito il pass-partout (PP1), due telecamere fisse (T1 e T2) e 5 sensori antincendio (S1 S2 S3 S4 S5). Il secondo

piano è costituito da tre stanze (R11 R12 R13), l'ascensore (L), quattro punti di collegamento del corridoio (P11 P12 P13 LP2), un punto in cui viene custodito il pass-partout (PP2), una telecamera fissa (T3) e tre sensori antincendio (S11 S12 S13). Il terzo piano è costituito da quattro stanze (R21 R22 R23 R24), l'ascensore (L), sei punti di collegamento del corridoio (P20 P21 P22 P23 P24 LP3), un punto in cui viene custodito il pass-partout (PP3), una telecamera fissa (T4) e quattro sensori antincendio (S21 S22 S23 S24).

I goal del problema consistono nel controllare tutte le stanze, supposte in uno stato normale (porte chiuse e non bloccate, sensori attivi) percorrendo dal punto LP1 per il robot R2 e dal punto LP2 per il robot R1 i corridoi, supposti anch'essi in uno stato normale (sensori antincendio spenti, telecamere funzionanti, pass-partout nella posizione apposita, connessioni bidirezionali tra i punti come in figura) e ritornando alla fine nei punti di partenza dopo aver chiuso tutte le porte e riattivato tutti i sensori delle stanze; tale problema viene descritto nel file "guard2.fct" riportato di seguito:

```
# guard2.fct
# ambiente con due robot, tre piani e un ascensore

location: P1 P2 P3 P4 P5 P10 P11 P12 P13 P20 P21 P22 P23 P24 LP1 LP2
LP3 PP1 PP2 PP3;
room: R1 R2 R3 R4 R5 R11 R12 R13 R21 R22 R23 R24;
robot: RG1 RG2;
lift: L;
pass-partout: PASS1 PASS2 PASS3;
fire: S1 S2 S3 S4 S5 S11 S12 S13 S21 S22 S23 S24;
camera: T1 T2 T3 T4;

initial:

at(RG1 LP2) has-fuel(RG1)
at(RG2 LP1) has-fuel(RG2)

con(P1 P2) con(P2 P1) con(P2 P3) con(P3 P2) con(P3 P4) con(P4 P3)
con(P4 P5) con(P5 P4)
con(P5 LP1) con(LP1 P5)  con(P1 LP1) con(LP1 P1)  con(PP1 P1) con(P1
PP1) at(PASS1 PP1)

in-front-of(P1 R1) sensor-on(R1) closed(R1) not-locked(R1)
in-front-of(P2 R2) sensor-on(R2) closed(R2) not-locked(R2)
in-front-of(P3 R3) sensor-on(R3) closed(R3) not-locked(R3)
in-front-of(P4 R4) sensor-on(R4) closed(R4) not-locked(R4)
in-front-of(P5 R5) sensor-on(R5) closed(R5) not-locked(R5)

con-lift(LP1 LP2)
con-lift(LP2 LP1)
in-front-of(LP1 L) closed(L LP1)
in-front-of(LP2 L) closed(L LP2)

con(LP2 P11) con(P11 LP2) con(P11 P12) con(P12 P11) con(P12 P13)
con(P13 P12) con(P13 P10) con(P10 P13) con(P10 LP2) con(LP2 P10)
con(PP2 P11) con(P11 PP2) at(PASS2 PP2)
in-front-of(P11 R11) sensor-on(R11) closed(R11) not-locked(R11)
in-front-of(P12 R12) sensor-on(R12) closed(R12) not-locked(R12)
in-front-of(P13 R13) sensor-on(R13) closed(R13) not-locked(R13)

con-lift(LP2 LP3)
```

```
con-lift(LP3 LP2)
con-lift(LP3 LP1)
con-lift(LP1 LP3)
in-front-of(LP3 L) closed(L LP3)

con(LP3 P21) con(P21 LP3) con(P21 P22) con(P22 P21) con(P22 P23)
con(P23 P22) con(P20 P23) con(P23 P20) con(P20 P24) con(P24 P20)
con(P24 LP3) con(LP3 P24) con(PP3 P21) con(P21 PP3) at(PASS3 PP3)

in-front-of(P21 R21) sensor-on(R21) closed(R21) not-locked(R21)
in-front-of(P22 R22) sensor-on(R22) closed(R22) not-locked(R22)
in-front-of(P23 R23) sensor-on(R23) closed(R23) not-locked(R23)
in-front-of(P24 R24) sensor-on(R24) closed(R24) not-locked(R24)

at(S1 P1) at(S2 P2) at(S3 P3) at(S4 P4) at(S5 P5) at(S11 P11) at(S12
P12) at(S13 P13) at(S21 P21) at(S22 P22) at(S23 P23) at(S24 P24)
at(T1 P3) at(T2 P4) at(T3 P12) at(T4 P20);


goal:

at(RG1 LP2)
at(RG2 LP1)

controlled(R1) controlled(R2) controlled(R3) controlled(R4)
controlled(R5)
controlled(R11) controlled(R12) controlled(R13)
controlled(R21) controlled(R22) controlled(R23) controlled(R24)

sensor-on(R1) sensor-on(R2) sensor-on(R3) sensor-on(R4) sensor-on(R5)
sensor-on(R11) sensor-on(R12)
sensor-on(R13)  sensor-on(R21) sensor-on(R22) sensor-on(R23) sensor-
on(R24)

at(PASS1 PP1) at(PASS2 PP2) at(PASS3 PP3);
```

La soluzione del problema, che viene memorizzata dal robot come piano predefinito nel file "guard2.sol" , è composta da 67 livelli:

```
0: call-lift_RG1_LP2_L
   move_RG2_LP1_P1
1: take-lift_RG1_LP2_LP3_L
   sensor-off_RG2_P1_R1
2: move_RG1_LP3_P21
   open-room_RG2_P1_R1
3: sensor-off_RG1_P21_R21
   enter-room_RG2_P1_R1
4: open-room_RG1_P21_R21
   control-room_RG2_R1
5: enter-room_RG1_P21_R21
   exit-room_RG2_P1_R1
6: control-room_RG1_R21
   close-room_RG2_P1_R1
7: exit-room_RG1_P21_R21
   sensor-on_RG2_P1_R1
8: close-room_RG1_P21_R21
```

```
   move_RG2_P1_P2
9: sensor-on_RG1_P21_R21
   sensor-off_RG2_P2_R2
10: move_RG1_P21_P22
    open-room_RG2_P2_R2
11: sensor-off_RG1_P22_R22
    enter-room_RG2_P2_R2
12: open-room_RG1_P22_R22
    control-room_RG2_R2
13: enter-room_RG1_P22_R22
    exit-room_RG2_P2_R2
14: control-room_RG1_R22
    close-room_RG2_P2_R2
15: exit-room_RG1_P22_R22
    sensor-on_RG2_P2_R2
16: close-room_RG1_P22_R22
    move_RG2_P2_P3
17: sensor-on_RG1_P22_R22
    sensor-off_RG2_P3_R3
18: move_RG1_P22_P23
    open-room_RG2_P3_R3
19: sensor-off_RG1_P23_R23
    enter-room_RG2_P3_R3
20: open-room_RG1_P23_R23
    control-room_RG2_R3
21: enter-room_RG1_P23_R23
    exit-room_RG2_P3_R3
22: control-room_RG1_R23
    close-room_RG2_P3_R3
23: exit-room_RG1_P23_R23
    sensor-on_RG2_P3_R3
24: close-room_RG1_P23_R23
    move_RG2_P3_P4
25: sensor-on_RG1_P23_R23
    sensor-off_RG2_P4_R4
26: move_RG1_P23_P20
    open-room_RG2_P4_R4
27: move_RG1_P20_P24
    enter-room_RG2_P4_R4
28: sensor-off_RG1_P24_R24
    control-room_RG2_R4
29: open-room_RG1_P24_R24
    exit-room_RG2_P4_R4
30: enter-room_RG1_P24_R24
    close-room_RG2_P4_R4
31: control-room_RG1_R24
    sensor-on_RG2_P4_R4
32: exit-room_RG1_P24_R24
    move_RG2_P4_P5
33: close-room_RG1_P24_R24
    sensor-off_RG2_P5_R5
34: sensor-on_RG1_P24_R24
    open-room_RG2_P5_R5
35: move_RG1_P24_LP3
    enter-room_RG2_P5_R5
36: call-lift_RG1_LP3_L
    control-room_RG2_R5
```

```
37: take-lift_RG1_LP3_LP2_L
    exit-room_RG2_P5_R5
38: move_RG1_LP2_P11
    close-room_RG2_P5_R5
39: sensor-off_RG1_P11_R11
    sensor-on_RG2_P5_R5
40: open-room_RG1_P11_R11
    move_RG2_P5_LP1
41: enter-room_RG1_P11_R11
    call-lift_RG2_LP1_L
42: control-room_RG1_R11
    take-lift_RG2_LP1_LP2_L
43: exit-room_RG1_P11_R11
    move_RG2_LP2_P10
44: close-room_RG1_P11_R11
    move_RG2_P10_P13
45: sensor-on_RG1_P11_R11
    sensor-off_RG2_P13_R13
46: move_RG1_P11_P12
    open-room_RG2_P13_R13
47: sensor-off_RG1_P12_R12
    enter-room_RG2_P13_R13
48: open-room_RG1_P12_R12
    control-room_RG2_R13
49: enter-room_RG1_P12_R12
    exit-room_RG2_P13_R13
50: control-room_RG1_R12
    close-room_RG2_P13_R13
51: exit-room_RG1_P12_R12
    sensor-on_RG2_P13_R13
52: close-room_RG1_P12_R12
    move_RG2_P13_P10
53: sensor-on_RG1_P12_R12
    move_RG2_P10_LP2
54: move_RG1_P12_P13
    call-lift_RG2_LP2_L
55: move_RG1_P13_LP2
    take-lift_RG2_LP2_LP1_L
```

Per i test sono stati utilizzati dei problemi ottenuti da questo problema base mediante aggiunte di goal o modifiche dello stato iniziale che rappresentano i possibili imprevisti che il robot guardiano può dover affrontare; a fronte di ciò sarà necessario modificare il piano predefinito aggiungendo, togliendo o modificando mosse e aggiungendo o togliendo livelli. Ad esempio, si consideri il problema denominato a1v4.fct, in cui viene aggiunta una stanza R20 da controllare al secondo piano di fronte al punto P20 (vedi figura 8): poiché il robot RG1 si muove nel livello 24 da P23 a P20, si potrebbe pensare di inserire dal livello 24 al livello 30 le 7 mosse necessarie per controllare la stanza R20 (`sensor-off_RG1_P20_R20`, `open-room_RG1_P20_R20`, `enter-room_RG1_P20_R20`, `control-room_RG1_R20`, `exit-room_RG1_P20_R20`, `close-room_RG1_P20_R20`, `sensor-on_RG1_P20_R20`) e spostare tutte le successive azioni del robot RG1 di 7 livelli. Non è esclusa comunque la possibilità che esista un piano migliore di questo in termini di numero di livelli, in quanto i due robot potrebbero dividersi queste sette azioni (occorrerebbe ovviamente cambiare completamente tutto il piano).

I problemi sono elencati e descritti di seguito, (a1 indica l'aggiunta di goal, b2 la modifica dello stato iniziale) riportando per ognuno un indice della difficoltà di adattamento in termini di numero

di mosse da aggiungere, togliere o modificare e numero di livelli da aggiungere o togliere al piano (tralasciando il numero di mosse da spostare dal livello in cui si trovano):

- a1v1.fct: ottenuto da guard2.fct chiedendo di controllare tutte le telecamere del piano terra; per ottenere un piano soluzione a partire da guard2.sol occorre aggiungere 4 mosse, ma non è necessario aggiungere livelli (piano risultante di 56 livelli);

- a1v2.fct: ottenuto da guard2.fct chiedendo di controllare i sensori antincendio del primo piano; per ottenere un piano soluzione a partire da guard2.sol occorre aggiungere 3 mosse, ma non è necessario aggiungere livelli (piano risultante di 56 livelli);

- a1v3.fct: ottenuto da guard2.fct aggiungendo una stanza (R10) al primo piano; per ottenere un piano soluzione a partire da guard2.sol occorre aggiungere sette mosse e 4 livelli (perché i due robot si dividono i compiti; piano risultante di 60 livelli);

- a1v4.fct: ottenuto da guard2.fct aggiungendo una stanza (R20) al secondo piano; per ottenere un piano soluzione a partire da guard2.sol occorre aggiungere sette mosse in 7 livelli (piano risultante di 63 livelli);

- b2v1.fct: ottenuto da guard2.fct cambiando lo stato della porta di R11 in bloccata; per ottenere un piano soluzione a partire da guard2.sol occorre aggiungere 5 mosse e spostarne molte (piano risultante di 61 livelli);

- b2v2.fct: ottenuto da guard2. cambiando lo stato della porta di R13 in bloccata; per ottenere un piano soluzione a partire da guard2.sol occorre 7 mosse e spostarne parecchie (piano risultante di 63 livelli);

- b2v3.fct: ottenuto da guard2.fct togliendo la connessione tra P10; per ottenere un piano soluzione a partire da guard2.sol occorre modificare 4 mosse e aggiungerne due (piano risultante di 58 livelli);

- b2v4.fct: ottenuto da guard2.fct togliendo la connessione tra P5; per ottenere un piano soluzione a partire da guard2.sol occorre aggiungere 5 mosse e toglierne 1 (piano risultante di 60 livelli);

- b2v5.fct: ottenuto da guard2.fct fct togliendo la connessione tra P20 e P24 e aggiungendone una tra P23 e P24; per ottenere un piano soluzione a partire da guard2.sol togliere una mossa e modificarne una (piano risultante di 55 livelli);

- b2v6.fct: ottenuto da guard2.fct spostando l'ascensore; l'adattamento del piano comporta molti cambiamenti, e il comportamento del pianificatore non è prevedibile.

Tutti i file relativi a questo dominio sono riportati nel CD-ROM allegato.

3) **guard3:** questo dominio è analogo a guard1, ma è stato modificato il metodo per sbloccare le porte; non si ha più la presenza di un pass-partout ma ogni robot ha la capacità di sbloccare le porte tramite un dispositivo elettronico.

Nel file degli operatori (guard.ops) gli operatori sopra descritti per prelevare e riportare il pass-partout sono stati tolti, mentre l'operatore `unlock-room` è stato sostituito dal seguente:

```
unlock-room
:v ?r robot ?l location ?p room
:p at(?r ?l) in-front-of(?l ?p) locked(?p)
:e ADD not-locked(?p)
   DEL locked(?p).
```

Il problema viene descritto nel file "guard3.fct" riportato di seguito:

```
# guard3.fct: ambiente con due piani e un ascensore

location: P1 P2 P3 P4 P5 P11 P12 P13 LP1 LP2;
room: R1 R2 R3 R4 R5 R11 R12 R13;
robot: RG;
lift: L;
fire: S1 S2 S3 S4 S5 S11 S12 S13 SL1 SL2;
camera: T1 T2 T3;

initial:

at(RG P11) has-fuel(RG)

con(P1 P2) con(P2 P1) con(P2 P3) con(P3 P2) con(P3 P4) con(P4 P3)
con(P4 P5) con(P5 P4)
con(P5 LP1) con(LP1 P5)  con(P1 LP1) con(LP1 P1)

in-front-of(P1 R1) sensor-on(R1) closed(R1) not-locked(R1)
in-front-of(P2 R2) sensor-on(R2) closed(R2) not-locked(R2)
in-front-of(P3 R3) sensor-on(R3) closed(R3) not-locked(R3)
in-front-of(P4 R4) sensor-on(R4) closed(R4) not-locked(R4)
in-front-of(P5 R5) sensor-on(R5) closed(R5) not-locked(R5)

con-lift(LP1 LP2)
con-lift(LP2 LP1)
in-front-of(LP1 L) closed(L LP1)
in-front-of(LP2 L) closed(L LP2)

con(LP2 P11) con(P11 LP2) con(P11 P12) con(P12 P11) con(P12 P13)
con(P13 P12) con(P13 LP2) con(LP2 P13)

in-front-of(P11 R11) sensor-on(R11) closed(R11) not-locked(R11)
in-front-of(P12 R12) sensor-on(R12) closed(R12) not-locked(R12)
in-front-of(P13 R13) sensor-on(R13) closed(R13) not-locked(R13)

at(S1 P1) at(S2 P2) at(S3 P3) at(S4 P4) at(S5 P5) at(S11 P11) at(S12
P12) at(S13 P13)
at(T1 P3) at(T2 P4) at(T3 P12);
```

```
goal:

at(RG LP1)

controlled(R1)  controlled(R2)  controlled(R3)  controlled(R4)
controlled(R5)
controlled(R11) controlled(R12) controlled(R13)

sensor-on(R1) sensor-on(R2) sensor-on(R3) sensor-on(R4) sensor-on(R5)
sensor-on(R11) sensor-on(R12)
sensor-on(R13);
```

La soluzione del problema è composta da un piano di 67 livelli contenuta nel file "guard3.sol" identica a "guard1.sol".

Per i test sono stati utilizzate solo le varianti di guard1 nelle quali l'imprevisto consisteva in una o più porte bloccate, in quanto le modifiche apportate agli operatori non influenzano la soluzione degli altri problemi.

- b2v6.fct: ottenuto da guard3.fct modificando lo stato della porta di R3, che è bloccata; per ottenere un piano soluzione a partire da guard3.sol occorre aggiungere 1 mossa (piano risultante di 67 livelli);
- b2v7.fct: ottenuto da guard3.fct modificando lo stato delle porte di R3 e R13, che sono bloccate; per ottenere un piano soluzione a partire da guard3.sol occorre aggiungere 2 mosse (piano risultante di 67 livelli).

- b2v8.fct: ottenuto da guard3.fct modificando lo stato della porta R1, che è bloccata; per ottenere un piano soluzione a partire da guard3.sol occorre aggiungere 1 mossa (piano risultante di 67 livelli);

- b2v10.fct: ottenuto da guard3.fct modificando lo stato delle porte di R1 e R5, che sono bloccate; per ottenere un piano soluzione a partire da guard3.sol occorre aggiungere 2 mosse (piano risultante di 67 livelli).

Tutti i file relativi a questo dominio sono riportati nel CD-ROM allegato.

3.2.6. Risultati sperimentali

Presentiamo di seguito i risultati dei test con il metodo di ricerca sistematica (tabella 10) e con il metodo di ricerca locale e sistematica (tabella 11); tutti i tempi sono indicati in secondi, e in tutte le tabelle L indica il numero di livelli presunti per la soluzione (calcolati a mano e non con un pianificatore ottimo, in quanto IPP non risolve questi problemi per out of memory).

Nella tabella 10 vengono riportati, per ogni problema di ogni dominio: il tempo impiegato per trovare la prima soluzione (Time1) e il relativo numero di livelli (L1), e il tempo impiegato per trovare la seconda soluzione (T2) e il relativo numero di livelli (L2).

Un segno --- per tutti i dati di ADJUST-PLAN indica che il problema non è stato risolto, mentre un segno --- per T2 e L2 indica che un secondo piano non è stato trovato entro 3600 secondi dal momento in cui il programma è stato lanciato.

Guard_1	ADJUST-PLAN				L
	Time1	L1	T2	L2	
a1v1	---	---	---	---	77
a1v2	1.06	72	14.68	70	67
a1v3	1.06	72	2.63	70	67
a1v4	108.01	83	---	---	77
a1v5	25.98	79	---	---	67
a1v6	13.05	80	---	---	80
a1v7	3.32	75	9.96	74	67
b2v1	1.9	68	---	---	68
b2v2	---	---	---	---	67
b2v3	1.03	68	---	---	68
b2v4	---	---	---	---	---
b2v5	2.91	70	---	---	69
b2v6	2.54	79	---	---	73
b2v7	44.29	95	---	---	79
b2v8	1.51	73	---	---	73
b2v9	1.32	71	---	---	---
b2v10	4.55	85	---	---	73
b2v11	1.4	69	---	---	---
b2v12	53.27	75	---	---	61
b2v13	0.69	68	---	---	68

Guard_2	ADJUST-PLAN				L
	Time1	L1	T2	L2	
a1v1	---	---	---	---	56
a1v2	5.86	59	---	---	56
a1v3	7.69	61	---	---	60
a1v4	254.28	65	---	---	63
b2v1	8.12	62	---	---	61
b2v2	16.65	64	---	---	63
b2v3	6.7	63	---	---	58
b2v4	7.63	61	---	---	60
b2v5	9.88	59	---	---	55
b2v6	---	---	---	---	---

Guard_3	ADJUST-PLAN				L
	Time1	L1	T2	L2	
b2v6	0.86	68	1.54	67	67
b2v7	1.29	69	1.99	68	67
b2v8	0.99	68	1.66	67	67
b2v10	1.09	69	1.8	68	67

Tabella 10: risultati dei test del metodo di ricerca sistematica per i domini Guard_1, Guard_2 e Guard_3.

Guard_1	T-Walkplan+ADJUST				L
	Time	T-L	T-A (l)	Lev	
a1v1	---	---	---	---	77
a1v2	2.873	2,873(9)	---	67	67
a1v3	4.395	2,835(2)	1,56(2)	72	67
a1v4	---	---	---	---	77
a1v5	321.817	8,315(4)	313,502(4)	79	67
a1v6	50.924	6,884(9)	44,04(9)	80	80
a1v7	24.196	3,846(9)	91,575(2)	67.2	67
b2v1	291.684	7,57(5)	284,114(5)	68.6	68
b2v2	0.549	0,549(10)	---	67	67
b2v3	24.848	3,174(9)	21,673(9)	68	68
b2v4	0.574	0,574(10)	---	67	---
b2v5	---	---	---	---	69
b2v6	---	---	---	---	73
b2v7	---	---	---	---	79
b2v8	109.83	3,32(10)	106,51(10)	77	73
b2v9	---	---	---	---	---
b2v10	---	---	---	---	73
b2v11	---	---	---	---	---
b2v12	12.153	1,501(10)	53,26(2)	68.6	61
b2v13	2.033	1,336(10)	0,697(10)	68	68

Guard_2	T-Walkplan+ADJUST				L
	Time	T-L	T-A (l)	Lev	
a1v1	---	---	---	---	56
a1v2	17.84	11.91(10)	5.933 (10)	59.4	56
a1v3	20.696	12.98(10)	7.713 (10)	61	60
a1v4	265.72	12.93(10)	252.8 (10)	65	63
b2v1	162.61	15.42 (8)	147.19 (8)	63	61
b2v2	118.80	13.7 (9)	105.1 (9)	64	63
b2v3	56.135	11.67(10)	44.47 (10)	62	58
b2v4	43.578	11.59(10)	31.99 (10)	64	60
b2v5	---	---	---	---	55
b2v6	---	---	---	---	---

Guard_3	T-Walkplan+ADJUST				L
	Time	T-L	T-A (l)	Lev	
b2v6	3.087	1.326	1.761(10)	72	67
b2v7	---	---	---	---	67
b2v8	2.859	1.333	1.526(10)	71	67
b2v10	2.925	1.361	1.564(10)	71	67

Tabella 11: risultati dei test del metodo di ricerca locale e sistematica per i domini Guard_1, Guard_2 e Guard_3.

Nella tabella 11 vengono riportati, per ogni problema di ogni dominio: la media del tempo totale impiegato per trovare la soluzione (Time; dato da T-L + T-A*(l)/10), il tempo medio su 10 prove impiegato dalla ricerca locale per trovare una soluzione definitiva o con un numero di

inconsistenze inferiore a quello limite (T-L), il tempo medio sul numero di prove indicato tra parentesi (l) impiegato dalla ricerca sistematica per trovare una soluzione definitiva partendo dal piano dato dalla ricerca locale (T-A; un segno --- indica che il problema era già stato risolto dalla ricerca locale) e il numero medio di livelli della soluzione finale (Lev). Un segno --- per tutti i dati di T-Walkplan+ADJUST indica che il problema non è stato risolto.

Poiché, come si è detto, IPP non risolve i problemi considerati, è evidente che entrambi i metodi di adattamento (ricerca sistematica e combinazione di ricerca locale e sistematica) risultano essere vantaggiosi rispetto ad una pianificazione completa.

3.3. Il dominio delle stanze

3.3.1. Azioni base del robot

Le azioni principali che il robot deve saper compiere in questo dominio sono:

- accendere le luci e inviare immagini tramite la propria telecamera;
- muoversi da un punto ad un altro di una stanza;
- identificare gli oggetti presenti (oggetti normali, oggetti preziosi, finestre e casseforti);
- controllare la presenza di oggetti;
- controllare la presenza e l'integrità di oggetti preziosi;
- controllare che le casseforti siano chiuse;
- controllare che le finestre siano chiuse;
- controllare lo stato dei sensori antincendio;
- dare l'allarme in caso di imprevisto.

3.3.2. Oggetti del dominio

Il dominio è costituito da oggetti che vengono specificati mediante le seguenti variabili:

- `location`, che rappresenta un punto in una stanza;
- `robot`;
- `object`, che rappresenta in generale un oggetto di cui controllare la presenza;
- `precious-object`, che rappresenta un oggetto prezioso di cui controllare, oltre che la presenza, anche l'integrità;
- `window`, che rappresenta una finestra di cui controllare lo stato (aperta o chiusa);
- `safe-box`, che rappresenta un'eventuale cassaforte da controllare;
- `fire`, che rappresenta un sensore antincendio.

3.3.3. Operatori per l'interazione con gli oggetti del dominio

Gli operatori sono stati suddivisi, a seconda del tipo di interazione che descrivono, in tre distinte categorie, quella dell'interazione tra robot e mondo, quella dell'interazione tra robot e centro di controllo e quella dell'interazione tra robot ed oggetti. Gli operatori sono di seguito descritti seguendo questa suddivisione:

1) Interazione robot-mondo: il robot si muove tra punti interconnessi della stanza con l'operatore `move` e si rifornisce di carburante con l'operatore `refuel`:

```
move
:v ?r robot ?s location ?a location
:p at(?r ?s) con(?s ?a) has-fuel(?r) image-on(?r)
```

```
:e ADD at(?r ?a)
   DEL at(?r ?s).

refuel
:v ?r robot
:p at(?r P0)
:e ADD has-fuel(?r).
```

Nota: l'operatore `refuel` non viene mai utilizzato nelle nostre simulazioni, perché si suppone che il problema della ricarica del robot venga risolto da un opportuno agente.

2) Interazione robot-centro di controllo: il robot invia immagini tramite il proprio sistema di visione al centro di controllo; nel caso le luci siano accese si utilizza l'operatore `send-image`, altrimenti occorre utilizzare l'operatore `light-image-on` per accendere anche le luci; alla fine luci e telecamera vengono disattivate con l'operatore `light-image-off`;

```
light-image-on
:v ?l light ?r robot
:p off(?l)
:e ADD on(?l) image-on(?r)
   DEL off(?l) image-off(?r).

send-image
:v ?l light ?r robot
:p on(?l)
:e ADD image-on(?r)
   DEL image-off(?r).
light-image-off
:v ?l light ?r robot
:p on(?l) image-on(?r)
:e ADD off(?l) image-off(?r) controlled()
   DEL on(?l) image-on(?r).
```

3) Interazione robot-oggetti da controllare: il robot deve identificare gli oggetti e quindi controllarli. L'identificazione degli oggetti avviene tramite un sensore di presenza posseduto dal robot e attivato da uno o più "marker" che sono presenti su ogni oggetto e ne individuano il tipo, la presenza (o assenza) per gli oggetti, la presenza e integrità (o assenza e non integrità) per gli oggetti preziosi, lo stato per le finestre e le casseforti (aperta, chiusa). Queste informazioni vengono aggiunte dal robot alla base di conoscenza sottoforma di fatti (ad esempio, se una finestra è aperta verrà inserito il fatto open(?f), viceversa verrà inserito il fatto closed(?f)). Per ogni tipo d'oggetto esistente sono previsti opportuni operatori per l'identificazione e il controllo, descritti di seguito:

   a) finestre: il robot identifica la presenza di una finestra tramite l'operatore `identify-window`; se la finestra è chiusa l'obiettivo di controllare la finestra viene raggiunto mediante l'operatore `control-window`, altrimenti occorre dare l'allarme tramite l'operatore `alarm-window`:

```
identify-window
:v ?w window ?r robot ?l location
:p at(?w ?l) at(?r ?l)
:e ADD identified(?w).

control-window
```

```
:v ?w window ?r robot ?l location
:p at(?w ?l) at(?r ?l) closed(?w) image-on(?r) identified(?w)
:e ADD controlled(?w).

alarm-window
:v ?w window ?r robot ?l location ?a alarm
:p at(?w ?l) at(?r ?l) open(?w) image-on(?r) identified(?w)
:e ADD controlled(?w) alarm-on(?a).
```

b) oggetti normali: il robot identifica la presenza di un oggetto tramite `l'operatore identify-object`; se l'oggetto è presente, l'obiettivo di controllarlo viene raggiunto tramite l'operatore `control-object`, altrimenti occorre dare l'allarme tramite l'operatore `alarm-object`:

```
identify-object
:v ?o object ?r robot ?l location
:p at(?o ?l) at(?r ?l)
:e ADD identified(?o).

control-object
:v ?o object ?r robot ?l location
:p present(?o ?l) at(?r ?l) image-on(?r) identified(?o)
:e ADD controlled(?o).

alarm-object
:v ?o object ?r robot ?l location ?a alarm
:p not-present(?o ?l) at(?r ?l) image-on(?r)
:e ADD controlled(?o) alarm-on(?a).
```

c) oggetti preziosi: : il robot identifica la presenza di un oggetto prezioso tramite `l'operatore identify-precious-object`; se l'oggetto è presente ed integro l'obiettivo di controllarlo viene raggiunto mediante l'operatore `control-precious-object`, altrimenti occorre dare l'allarme tramite l'operatore `alarm-precious-object`:

```
identify-precious-object
:v ?p precious-object ?r robot ?l location
:p at(?p ?l) at(?r ?l)
:e ADD identified(?p).

control-precious-object
:v ?p precious-object ?r robot ?l location
:p at(?r ?l) present-complete(?p ?l) image-on(?r) identified(?p)
:e ADD controlled(?p).

alarm-precious-object
:v ?p precious-object ?r robot ?l location ?a alarm
:p at(?r ?l) not-present-complete(?p ?l) image-on(?r)
:e ADD controlled(?p) alarm-on(?a).
```

d) casseforti; il robot identifica la presenza di una cassaforte tramite `l'operatore identify-safe-box`; se la cassaforte è chiusa l'obiettivo di controllarla viene raggiunto mediante l'operatore `control-safe-box`, altrimenti occorre dare l'allarme tramite l'operatore `alarm-safe-box`:

```
identify-safe-box
```

```
:v ?s safe-box ?r robot ?l location
:p at(?s ?l) at(?r ?l)
:e ADD identified(?s).

control-safe-box
:v ?s safe-box ?r robot ?l location
:p at(?s ?l) at(?r ?l) locked(?s) image-on(?r) identified(?s)
:e ADD controlled(?s).

alarm-safe-box
:v ?s safe-box ?r robot ?l location ?a alarm
:p at(?s ?l) at(?r ?l) not-locked(?s) image-on(?r) identified(?s)
:e ADD controlled(?s) alarm-on(?a).
```

e) sensori antincendio: vengono controllati dal robot mediante l'operatore `control-fire`:

```
control-fire
:v ?r robot ?l location ?f fire
:p at(?r ?l) at(?f ?l)
:e ADD controlled(?f).
```

Gli operatori sopra descritti sono contenuti nel file "room.ops" riportato nel CD-ROM allegato.

3.3.4. Eventi imprevisti

Durante lo svolgimento dei propri compiti il robot guardiano può incorrere in imprevisti di varia natura. Classifichiamo questi imprevisti di due categorie:

1) imprevisti **non critici**, che si suppone siano risolvibili autonomamente dal robot (al più comunicando con il centro di controllo per ottenere informazioni necessarie); ad esempio:
   a) l'incontro di un oggetto o altro ingombro sul percorso;
   b) l'ordine da parte del centro di controllo di eseguire una specifica azione che esula dal piano in esecuzione (ad esempio il controllo di un sensore antincendio o di un oggetto solitamente ignorato);
   c) oggetti che vengono spostati dalla locazione abituale;
2) imprevisti **critici**, che non possono essere affrontati autonomamente dal robot ma che richiedono l'intervento di un operatore umano; ad esempio:
   a) oggetti che non sono presenti, oggetti preziosi non presenti o non integri, casseforti o finestre non chiuse;
   b) presenza di persone non conosciute (che non si fanno cioè identificare dal robot tramite una tessera magnetica appositamente prevista).

Ci siamo focalizzati, per i test degli algoritmi di adattamento, sugli imprevisti non critici, dei quali abbiamo presentato più sopra gli operatori; gli operatori per trattare gli imprevisti critici sono riportati nel paragrafo 3.4., nel quale viene presentata la formalizzazione di un dominio reale.

Quando avviene un imprevisto il robot deve interrompere il piano attuale e ripianificare le proprie azioni in modo tale da superarlo e riprendere l'esecuzione del piano originario.

Nel caso di oggetti non presenti, o analogamente di oggetti preziosi non presenti o non integri o di casseforti o finestre non chiuse, e nel caso della presenza di persone non identificate, il robot dovrà lanciare l'allarme per avvisare il centro di controllo della situazione anomala; ciò porterà all'intervento di un operatore umano.

Nel caso di un ingombro sul percorso, il robot deve ripianificare il percorso da seguire in modo da evitare l'ingombro e portare ugualmente a termine il piano che stava eseguendo.

Nel caso di un ordine da parte del centro di controllo, il robot deve interrompere il piano che stava eseguendo, pianificare un percorso che lo porti all'esecuzione dell'ordine ricevuto e, dopo averlo eseguito, riprendere il piano precedente dal punto di interruzione o meglio riprenderne la parte non ancora eseguita e ripianificarla in modo che possa essere svolta dal luogo in cui il robot si trova.

Nel caso di un oggetto che viene spostato dalla sua posizione abituale, il robot deve ripianificare il percorso da seguire in modo da controllare l'oggetto nella sua nuova posizione.

3.3.5. Esempi di dominio

Abbiamo ipotizzato due possibili scenari per il dominio delle stanze del robot guardiano denominandoli "room1" e "room2"; diamo di seguito una descrizione di questi due domini.

1) **room1**: il mondo può essere descritto graficamente come in figura 9:

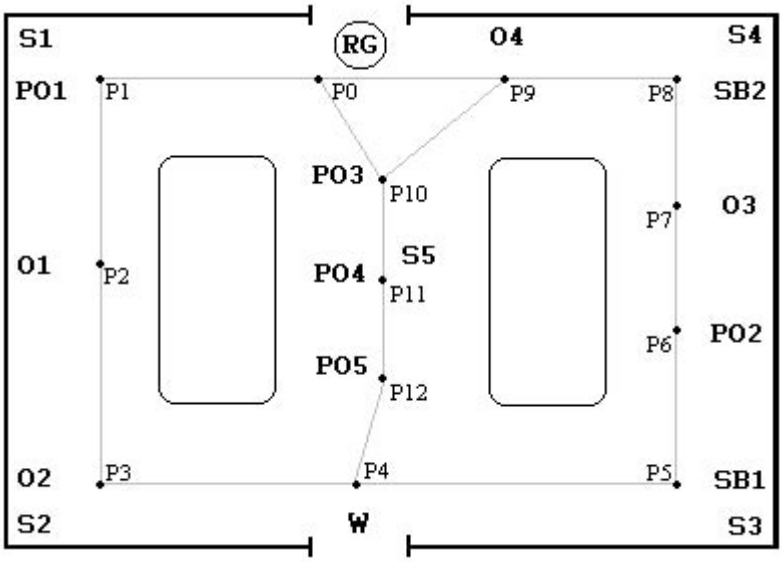


Figura 9: descrizione grafica del problema room1.

Il mondo è costituito da 13 punti collegati per la navigazione nella stanza (P0 … P12), da 5 oggetti (O1 … O4), da 6 oggetti preziosi (PO1 … PO5), da 1 finestra (W), da due casseforti (SB1 SB2) e da 5 sensori antincendio (S1 … S5).

I goal del problema consistono nell'accendere la luce e inviare immagini al centro di controllo, e nel controllare tutti gli oggetti, supposti in uno stato normale (oggetti presenti, oggetti preziosi presenti e integri, casseforti e finestre chiuse) percorrendo dal punto P0 la stanza e ritornando alla fine in P0, spegnendo la luce e smettendo di inviare immagini; ciò è descritto nel file "room1.fct" riportato di seguito:

```
# room1.fct

alarm: A;
robot: RG;
object: O1 O2 O3 O4;
precious-object: PO1 PO2 PO3 PO4 PO5;
window: W;
safe-box: SB1 SB2;
```

```
light: L;
location: P0 P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12;
fire: S1 S2 S3 S4 S5;

initial:

image-off(RG)
off(L)
at(RG P0) has-fuel(RG)

at(O1 P2) present(O1 P2)
at(O2 P3) present(O2 P3)
at(O3 P7) present(O3 P7)
at(O4 P9) present(O4 P9)
at(PO1 P1) present-complete(PO1 P1)
at(PO2 P6) present-complete(PO2 P6)
at(PO3 P10) present-complete(PO3 P10)
at(PO4 P11) present-complete(PO4 P11)
at(PO5 P12) present-complete(PO5 P12)
at(W P4) closed(W)
at(SB1 P5) locked(SB1) at(SB2 P8) locked(SB2)

con(P0 P1) con(P1 P0) con(P0 P10) con(P10 P0) con(P9 P0) con(P0 P9)
con(P9 P10) con(P10 P9)
con(P10 P11) con(P11 P10) con(P11 P12) con(P12 P11) con(P12 P4)
con(P4 P12)
con(P9 P8) con(P8 P9) con(P8 P7) con(P7 P8) con(P7 P6) con(P6 P7)
con(P1 P2) con(P2 P1) con(P2 P3) con(P3 P2)
con(P3 P4) con(P4 P3) con(P4 P5) con(P5 P4)
con(P5 P6) con(P6 P5)

at(S1 P1) at(S2 P3) at(S3 P5) at(S4 P8) at(S5 P11);

goal:

at(RG P0)
controlled(PO1) controlled(PO2) controlled(PO3) controlled(PO4)
controlled(PO5)
controlled(O1) controlled(O2) controlled(O3) controlled(O4)
controlled(W)
controlled(SB1) controlled(SB2)
controlled();
```

La soluzione del problema è composta da un piano di 41 livelli contenuta nel file "room1.sol" e riportata di seguito:

```
0: light-image-on_L_RG
1: move_RG_P0_P10
2: identify-precious-object_PO3_RG_P10
3: control-precious-object_PO3_RG_P10
4: move_RG_P10_P9
5: identify-object_O4_RG_P9
6: control-object_O4_RG_P9
7: move_RG_P9_P8
8: identify-safe-box_SB2_RG_P8
9: control-safe-box_SB2_RG_P8
10: move_RG_P8_P7
```

```
11: identify-object_O3_RG_P7
12: control-object_O3_RG_P7
13: move_RG_P7_P6
14: identify-precious-object_PO2_RG_P6
15: control-precious-object_PO2_RG_P6
16: move_RG_P6_P5
17: identify-safe-box_SB1_RG_P5
18: control-safe-box_SB1_RG_P5
19: move_RG_P5_P4
20: identify-window_W_RG_P4
21: control-window_W_RG_P4
22: move_RG_P4_P12
23: identify-precious-object_PO5_RG_P12
24: control-precious-object_PO5_RG_P12
25: move_RG_P12_P11
26: identify-precious-object_PO4_RG_P11
27: control-precious-object_PO4_RG_P11
28: move_RG_P11_P12
29: move_RG_P12_P4
30: move_RG_P4_P3
31: identify-object_O2_RG_P3
32: control-object_O2_RG_P3
33: move_RG_P3_P2
34: identify-object_O1_RG_P2
35: control-object_O1_RG_P2
36: move_RG_P2_P1
37: identify-precious-object_PO1_RG_P1
38: control-precious-object_PO1_RG_P1
39: move_RG_P1_P0
40: light-image-off_L_RG
```

Per i test sono stati utilizzati dei problemi ottenuti da questo problema base mediante aggiunte di goal o modifiche dello stato iniziale che rappresentano i possibili imprevisti che il robot guardiano può dover affrontare; a fronte di ciò sarà necessario modificare il piano predefinito aggiungendo, togliendo o modificando mosse e aggiungendo o togliendo livelli. Ad esempio, si consideri il problema denominato b2v7.fct, in cui l'oggetto PO2 viene spostato nel punto P5 e la finestra W è aperta (vedi figura 9); poiché PO2 non è più in P6 ma è in P5, occorrerà togliere le mosse `14: identify-precious-object_PO2_RG_P6` e `15: control-precious-object_PO2_RG_P6` e spostarle al livello 17 (che diventerà 16) come `identify-precious-object_PO2_RG_P5` e `control-precious-object_PO2_RG_P5`; in più occorrerà modificare la mossa `control-window_W_RG_P4` in `alarm-window_W_RG_P4`. Il piano risultante avrà quindi un livello in meno rispetto a quello di partenza.

I problemi sono elencati e descritti di seguito, (a1 indica l'aggiunta di goal, b2 la modifica dello stato iniziale) riportando per ognuno un indice della difficoltà di adattamento in termini di numero di mosse da aggiungere, togliere o modificare e numero di livelli da aggiungere o togliere al piano (tralasciando il numero di mosse da spostare dal livello in cui si trovano):

- a1v1.fct: ottenuto da room1.fct aggiungendo un goal, e cioè controllare il sensore antincendio S1; è possibile ottenere un piano soluzione da room1.sol aggiungendo una mossa senza aggiungere un livello (soluzione risultante di 41 livelli);

- a1v2.fct: ottenuto da room1.fct aggiungendo un oggetto prezioso da controllare (PO6 in P7); è possibile ottenere un piano soluzione da room1.sol aggiungendo due mosse senza aggiungere un livello (soluzione risultante di 41 livelli);

- a1v3.fct: ottenuto da room1.fct chiedendo di controllare tutti i sensori; è possibile ottenere un piano soluzione da room1.sol aggiungendo 5 mosse senza aggiungere livelli (soluzione risultante di 41 livelli);

- a1v4.fct: ottenuto da room1.fct aggiungendo due oggetti da controllare (PO6 in P6 e O5 in P5); è possibile ottenere un piano soluzione da room1.sol aggiungendo quattro mosse senza aggiungere livelli (soluzione risultante di 41 livelli);

- a1v5.fct: ottenuto da room1.fct un punto P13 tra P3 e P4 con un oggetto da controllare (O5); è possibile ottenere un piano soluzione da room1.sol modificando una mossa e aggiungendone tre in altrettanti livelli (soluzione risultante di 44 livelli);

- a1v6.fct: combinando le variazioni di a1v3.fct e a1v5.fct; è possibile ottenere un piano soluzione da room1.sol modificando una mossa e aggiungendone otto (soluzione risultante di 44 livelli);

- b2v1.fct: ottenuto da room1.fct togliendo la connessione tra P0 e P10; è possibile trovare un piano soluzione da room1.sol modificando due mosse, togliendone due e aggiungendone tre (soluzione risultante di 42 livelli);

- b2v2.fct: ottenuto da room1.fct modificando lo stato della finestra W in aperta; è possibile trovare un piano soluzione da room1.sol modificando una mossa (soluzione risultante di 41 livelli);

- b2v3.fct: ottenuto da room1.fct modificando lo stato dell'oggetto O3 in assente e lo stato dell'oggetto PO1 in non integro; è possibile trovare un piano soluzione da room1.sol modificando due mosse (soluzione risultante di 41 livelli);

- b2v4.fct: ottenuto da room1.fct togliendo la connessione tra P12 e P4; è possibile trovare un piano soluzione da room1.sol togliendo otto mosse e aggiungendone otto (soluzione risultante di 41 livelli);

- b2v5.fct: ottenuto da room1.fct togliendo la connessione tra P7 e P8 e aggiungendone una tra P7 e P9; è possibile trovare un piano soluzione da room1.sol modificando una mossa e aggiungendone una (soluzione risultante di 42 livelli);

- b2v6.fct: ottenuto da room1.spostando l'oggetto PO2 in P5; è possibile trovare un piano soluzione da room1.sol togliendo una mossa e aggiungendone una (soluzione risultante di 40 livelli);

- b2v7.fct: ottenuto combinando le variazioni di b2v6.fct e b2v2.fct; è possibile trovare un piano soluzione da room1.sol modificando una mossa, aggiungendone una e togliendone una (soluzione risultante di 40 livelli);

- b2v8.fct: ottenuto da room1.fct togliendo la connessione tra P11 e P12; è possibile trovare un piano soluzione da room1.fct togliendo quattro mosse e aggiungendone quattro (soluzione risultante di 41 livelli);

- b2v9.fct: ottenuto combinando le variazioni di b2v1.fct e b2v3.fct; è possibile trovare un piano soluzione da room1.sol modificando tre mosse, aggiungendone tre e togliendone tre (soluzione risultante di 42 livelli).

Tutti i file relativi a questo dominio sono riportati nel CD-ROM allegato.

2) **room2**: il mondo può essere descritto graficamente come in figura 10:

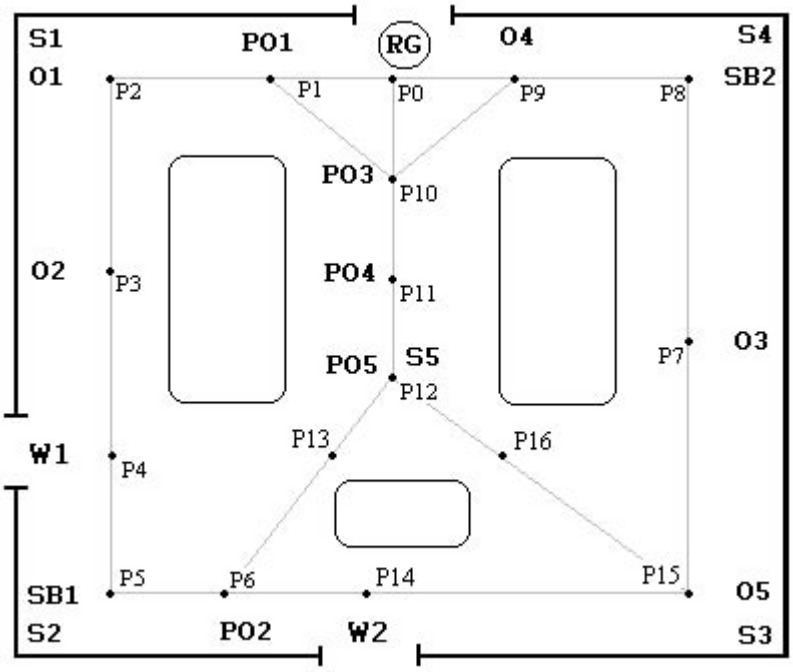


Figura 10: descrizione grafica del problema room2.

Il mondo è costituito da 17 punti collegati per la navigazione nella stanza (P0 … P16), da 6 oggetti (O1 … O5), da 6 oggetti preziosi (PO1 … PO5), da 2 finestre (W1 W2), da due casseforti (SB1 SB2) e da 5 sensori antincendio (S1 … S5).

I goal del problema consistono nell'accendere la luce e inviare immagini al centro di controllo, e nel controllare tutti gli oggetti, supposti in uno stato normale (oggetti presenti, oggetti preziosi presenti e integri, casseforti e finestre chiuse) percorrendo dal punto P0 la stanza e ritornando alla fine in P0, spegnendo la luce e smettendo di inviare immagini; ciò è descritto nel file "room2.fct" riportato di seguito:

```
# room2.fct

alarm: A;
robot: RG;
object: O1 O2 O3 O4 O5;
precious-object: PO1 PO2 PO3 PO4 PO5;
window: W1 W2;
safe-box: SB1 SB2;
light: L;
location: P0 P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12 P13 P14 P15 P16;
fire: S1 S2 S3 S4 S5;



initial:
```

```
image-off(RG)
off(L)
at(RG P0) has-fuel(RG)

at(O1 P2) present(O1 P2)
at(O2 P3) present(O2 P3)
at(O3 P7) present(O3 P7)
at(O4 P9) present(O4 P9)
at(O5 P15) present(O5 P15)
at(PO1 P1) present-complete(PO1 P1)
at(PO2 P6) present-complete(PO2 P6)
at(PO3 P10) present-complete(PO3 P10)
at(PO4 P11) present-complete(PO4 P11)
at(PO5 P12) present-complete(PO5 P12)
at(W1 P4) closed(W1) at(W2 P14) closed(W2)
at(SB1 P5) locked(SB1) at(SB2 P8) locked(SB2)

con(P0 P1) con(P1 P0) con(P10 P1) con(P1 P10) con(P0 P10) con(P10 P0)
con(P9 P0) con(P0 P9) con(P9 P10) con(P10 P9)
con(P10 P11) con(P11 P10) con(P11 P12) con(P12 P11) con(P12 P13)
con(P13 P12) con(P12 P16) con(P16 P12)
con(P9 P8) con(P8 P9) con(P8 P7) con(P7 P8) con(P7 P15) con(P15 P7)
con(P16 P15) con(P15 P16) con(P14 P15) con(P15 P14)
con(P13 P6) con(P6 P13) con(P6 P14) con(P14 P6)
con(P1 P2) con(P2 P1) con(P2 P3) con(P3 P2)
con(P3 P4) con(P4 P3) con(P4 P5) con(P5 P4)
con(P5 P6) con(P6 P5)

at(S1 P2) at(S2 P5) at(S3 P15) at(S4 P8) at(S5 P12);


goal:

at(RG P0)
controlled(PO1) controlled(PO2) controlled(PO3) controlled(PO4)
controlled(PO5)
controlled(O1) controlled(O2) controlled(O3) controlled(O4)
controlled(O5)
controlled(W1) controlled(W2)
controlled(SB1) controlled(SB2)
controlled();
```

La soluzione del problema è composta da un piano di 47 livelli contenuta nel file “room2.sol” e riportata di seguito:

```
0: light-image-on_L_RG
1: move_RG_P0_P10
2: identify-precious-object_PO3_RG_P10
3: control-precious-object_PO3_RG_P10
4: move_RG_P10_P11
5: identify-precious-object_PO4_RG_P11
6: control-precious-object_PO4_RG_P11
7: move_RG_P11_P12
8: identify-precious-object_PO5_RG_P12
9: control-precious-object_PO5_RG_P12
10: move_RG_P12_P11
```

```
11: move_RG_P11_P10
12: move_RG_P10_P9
13: identify-object_O4_RG_P9
14: control-object_O4_RG_P9
15: move_RG_P9_P8
16: identify-safe-box_SB2_RG_P8
17: control-safe-box_SB2_RG_P8
18: move_RG_P8_P7
19: identify-object_O3_RG_P7
20: control-object_O3_RG_P7
21: move_RG_P7_P15
22: identify-object_O5_RG_P15
23: control-object_O5_RG_P15
24: move_RG_P15_P14
25: identify-window_W2_RG_P14
26: control-window_W2_RG_P14
27: move_RG_P14_P6
28: identify-precious-object_PO2_RG_P6
29: control-precious-object_PO2_RG_P6
30: move_RG_P6_P5
31: identify-safe-box_SB1_RG_P5
32: control-safe-box_SB1_RG_P5
33: move_RG_P5_P4
34: identify-window_W1_RG_P4
35: control-window_W1_RG_P4
36: move_RG_P4_P3
37: identify-object_O2_RG_P3
38: control-object_O2_RG_P3
39: move_RG_P3_P2
40: identify-object_O1_RG_P2
41: control-object_O1_RG_P2
42: move_RG_P2_P1
43: identify-precious-object_PO1_RG_P1
44: control-precious-object_PO1_RG_P1
45: move_RG_P1_P0
46: light-image-off_L_RG
```

Per i test sono stati utilizzati dei problemi ottenuti da questo problema base mediante aggiunte di goal o modifiche dello stato iniziale che rappresentano i possibili imprevisti che il robot guardiano può dover affrontare; a fronte di ciò sarà necessario modificare il piano predefinito aggiungendo, togliendo o modificando mosse e aggiungendo o togliendo livelli. Ad esempio, si consideri il problema denominato a1v6.fct, in cui viene aggiunto un punto P17 tra i punti P7 e P15 (vedi figura 10) con un oggetto O6 da controllare; poichè P7 non è più connesso direttamente a P15, ma lo è attraverso P17, occorrerà modificare la mossa `21: move_RG_P7_P15` in `21: move_RG_P7_P17`, aggiungere le mosse `22: identify-object_O6_RG_P17` e `23: control-object_O6_RG_P17`, aggiungere la mossa `24: move_RG_P17_P15` e spostare le mosse del vecchio piano dalla 25 alla 46 di 3 livelli. Il piano risultante sarà quindi composto da 50 livelli.

I problemi sono elencati e descritti di seguito, (a1 indica l'aggiunta di goal, b2 la modifica dello stato iniziale) riportando per ognuno un indice della difficoltà di adattamento in termini di numero di mosse da aggiungere, togliere o modificare e numero di livelli da aggiungere o togliere al piano (tralasciando il numero di mosse da spostare dal livello in cui si trovano):

- a1v1.fct: ottenuto da room2.fct aggiungendo un oggetto prezioso da controllare (O6 in P13); è possibile ottenere un piano soluzione da room2.sol aggiungendo quattro mosse in altrettanti livelli (soluzione risultante di 51 livelli);

- a1v2.fct: ottenuto da room2.fct aggiungendo un goal, e cioè controllare il sensore antincendio S5; è possibile ottenere un piano soluzione da room2.sol aggiungendo una mossa senza aggiungere un livello (soluzione risultante di 47 livelli);

- a1v3.fct: ottenuto da room2.fct chiedendo di controllare tutti i sensori; è possibile ottenere un piano soluzione da room2.sol aggiungendo 5 mosse senza aggiungere livelli (soluzione risultante di 47 livelli);
- a1v4.fct: ottenuto da room2.fct due oggetti da controllare (PO6 in P16 e O6 in P13); è possibile ottenere un piano soluzione da room2.sol aggiungendo quattro mosse senza aggiungere livelli (soluzione risultante di 47 livelli);

- a1v5.fct: ottenuto da a1v4.fct aggiungendo un altro oggetto da controllare (O7 in P5); è possibile ottenere un piano soluzione da room2.sol aggiungendo sei mosse senza aggiungere livelli (soluzione risultante di 47 livelli);

- a1v6.fct: ottenuto da room2.fct aggiungendo il punto P17 tra P7 e P15 con un oggetto da controllare (O6); è possibile ottenere un piano soluzione da room2.sol modificando una mossa aggiungendone tre in altrettanti livelli (soluzione risultante di 50 livelli);

- b2v1.fct: ottenuto da room2.fct togliendo la connessione tra P11 e P12; è possibile trovare un piano soluzione da room2.sol togliendo quattro mosse e aggiungendone sei (soluzione risultante di 49 livelli);

- b2v2.fct: ottenuto da room2.fct togliendo la connessione tra P11 e P0; è possibile trovare un piano soluzione da room2.sol togliendo otto mosse e aggiungendone dieci (soluzione risultante di 51 livelli);

- b2v3.fct: ottenuto da room2.fct togliendo la connessione tra P14 e P6; è possibile trovare un piano soluzione da room2.sol togliendo una mossa e aggiungendone cinque (soluzione risultante di 50 livelli);

- b2v4.fct: ottenuto da room2.fct spostando l'oggetto PO5 in P13; è possibile trovare un piano soluzione da room2.sol togliendo quattro mosse e aggiungendone quattro (soluzione risultante di 47 livelli);

- b2v5.fct: ottenuto da room2.fct spostando PO4 in P16; è possibile trovare un piano soluzione da room2.sol togliendo sei mosse e aggiungendone sei (soluzione risultante di 47 livelli);

- b2v6.fct: ottenuto da room2.fct supponendo che la cassaforte SB1 non sia chiusa; è possibile trovare un piano soluzione da room2.sol modificando una mossa (soluzione risultante di 47 livelli);

- b2v7.fct: ottenuto da b2v6.fct supponendo che la finestra W2 sia aperta; è possibile trovare un piano soluzione da room2.sol modificando due mosse (soluzione risultante di 47 livelli);

- b2v8.fct: ottenuto da room2.fct togliendo la connessione tra P4 e P3; l'adattamento del piano risulta essere difficile;

- b2v9.fct: ottenuto da room2.fct spostando l'oggetto PO3 in P16 e supponendo che l'oggetto PO5 non sia integro; l'adattamento del piano risulta essere difficile.

Tutti i file relativi a questo dominio sono riportati nel CD-ROM allegato.

3.3.6. Risultati sperimentali

Presentiamo di seguito i risultati dei test con il metodo di ricerca sistematica (tabella 12) e con il metodo di ricerca locale e sistematica (tabella 13); tutti i tempi sono indicati in secondi, e in tutte le tabelle L indica il numero di livelli presunti per la soluzione (calcolati a mano e non con un pianificatore ottimo, in quanto IPP non risolve i problemi per out of memory).

Nella tabella 12 vengono riportati, per ogni problema di ogni dominio: il tempo impiegato per trovare la prima soluzione (Time1) e il relativo numero di livelli (L1), e il tempo impiegato per trovare la seconda soluzione (T2) e il relativo numero di livelli (L2).

Un segno --- per tutti i dati di ADJUST-PLAN indica che il problema non è stato risolto, mentre un segno --- per T2 e L2 indica che un secondo piano non è stato trovato entro 3600 secondi dal momento in cui il programma è stato lanciato.

Room_1	**ADJUST-PLAN**				**L**
	Time1	L1	T2	L2	
a1v1	0.29	42	0.5	41	41
a1v2	1.02	49	9.6	48	41
a1v3	---	---	---	---	41
a1v4	---	---	---	---	41
a1v5	0.62	48	---	---	44
a1v6	---	---	---	---	44
b2v1	0.64	44	---	---	42
b2v2	0.47	45	1.06	43	41
b2v3	0.43	46	3.2	45	41
b2v4	0.56	49	9.46	47	41
b2v5	0.51	42	---	---	42
b2v6	0.62	47	1.31	45	40
b2v7	1.35	45	3.77	43	40
b2v8	0.67	49	1.13	47	41
b2v9	4.36	47	---	---	42

Room_2	**ADJUST-PLAN**				**L**
	Time1	L1	T2	L2	
a1v1	2.14	56	10.12	55	51
a1v2	0.69	53	---	---	47
a1v3	---	---	---	---	47
a1v4	---	---	---	---	47
a1v5	---	---	---	---	47
a1v6	3.93	58	---	---	50
b2v1	---	---	---	---	49
b2v2	---	---	---	---	51
b2v3	0.58	51	---	---	51
b2v4	1.92	54	8.99	53	47
b2v5	1.62	54	---	---	47
b2v6	0.81	53	1.7	51	47
b2v7	4.24	55	12.67	53	47
b2v8	0.64	55	---	---	---
b2v9	3.2	54	---	---	---

Tabella 12: risultati dei test del metodo di ricerca sistematica per i domini Room_1 e Room_2.

Nella tabella 13 vengono riportati, per ogni problema di ogni dominio: la media del tempo totale impiegato per trovare la soluzione (Time; dato da T-L + T-A*(l)/(n)), il tempo medio sul numero di prove indicato tra parentesi (n) impiegato dalla ricerca locale per trovare una soluzione definitiva o con un numero di inconsistenze inferiore a quello limite (T-L), il tempo medio sul numero di prove indicato tra parentesi (l) impiegato dalla ricerca sistematica per trovare una soluzione definitiva partendo dal piano dato dalla ricerca locale (T-A; un segno --- indica che il problema era già stato risolto dalla ricerca locale) e il numero medio di livelli della soluzione finale (Lev). Un segno --- per tutti i dati di T-Walkplan+ADJUST indica che il problema non è stato risolto.

Room_1	**T-Walkplan+ADJUST**				**L**
	Time	T-L(n)	T-A(l)	Lev	

Room_2	**T-Walkplan+ADJUST**				**L**
	Time	T-L(n)	T-A(l)	Lev	

a1v1	0.749	0,53(10)	0,73(3)	41.8	41		a1v1	23.313	1,034(9)	22,279(9)	56	51
a1v2	1.239	0,6(10)	1,065(6)	45.8	41		a1v2	1.763	0,983(10)	0,78(10)	53.1	47
a1v3	135.17	1,0687(6)	201,15(4)	47.3	41		a1v3	---	---	---	---	47
a1v4	5.016	0,708(10)	4,308(10)	51	41		a1v4	---	---	---	---	47
a1v5	2.508	1,496(10)	1,012(10)	47.8	44		a1v5	564.42	19,47(1)	544,95(1)	61	47
a1v6	---	---	---	---	44		a1v6	---	---	---	---	50
b2v1	---	---	---	---	42		b2v1	6.646	0,889(10)	5,757(10)	53.4	49
b2v2	0.553	0,504(10)	0,49(1)	41.4	41		b2v2	1.347	0,91(10)	0,437(10)	48.2	51
b2v3	1.057	0,556(10)	0,835(6)	44.4	41		b2v3	---	---	---	---	51
b2v4	3.032	0,634(10)	2,398(10)	43.5	41		b2v4	4.82	0,908(10)	3,912(10)	54.1	47
b2v5	---	---	---	---	42		b2v5	24.402	1,054(9)	23,348(9)	54.1	47
b2v6	1.194	0,586(10)	1,014(6)	44.2	40		b2v6	1.457	0,883(10)	0,82(7)	50.4	47
b2v7	2.029	0,644(10)	1,385(10)	45.2	40		b2v7	4.958	1,184(10)	4,193(9)	54.2	47
b2v8	1.661	0,64(10)	1,021(10)	43.7	41		b2v8	2.435	1,22(2)	1,215(2)	56	---
b2v9	---	---	---	---	42		b2v9	3.652	0,919(10)	2,733(10)	54.2	---

Tabella 13: risultati dei test del metodo di ricerca locale e sistematica per i domini Room_1 e Room_2.

Poiché, come si è detto, IPP non risolve i problemi considerati, è evidente che entrambi i metodi di adattamento (ricerca sistematica e combinazione di ricerca locale e sistematica) risultano essere vantaggiosi rispetto ad una pianificazione completa.

3.4. Formalizzazione di un dominio reale

In questa sezione descriviamo un dominio più complicato ma più vicino alla realtà rispetto ai precedenti, nel quale si prendono in considerazione aspetti finora trascurati che rendono l'analisi impossibile per le tecniche da noi considerate. Più precisamente, vengono considerati:

- operatori con associato un costo di applicazione;
- operatori prioritari rispetto ad altri;
- operatori più complessi;
- ambienti più complessi rispetto ai precedenti per quanto riguarda numero, dislocazione e topologia delle stanze e dei corridoi.

Il dominio rimane comunque diviso in due ambienti, quello dei corridoi e quello delle stanze.

3.4.1. Azioni base del robot

Per entrambi i domini, dei corridoi e delle stanze, le azioni base che il robot può compiere sono quelle presentate nei rispettivi paragrafi precedenti con l'aggiunta delle seguenti:

- riconoscere l'identità di una persona che si muove nell'ambiente;
- ricaricare la batteria presso opportuni punti di ricarica dislocati nell'ambiente.

3.4.2. Oggetti del dominio

Per entrambi i domini, dei corridoi e delle stanze, gli oggetti che li costituiscono sono quelli specificati nei rispettivi paragrafi precedenti con l'aggiunta di:

- `person`, che rappresenta persone che il robot può incontrare;
- `refuel-point`, che rappresenta un punto di ricarica per la batteria del robot.

3.4.3. Operatori per l'interazione con gli oggetti del dominio

Le principali modifiche apportate agli operatori sono:

- l'inserimento di un costo d'esecuzione associato ad ogni operatore;
- l'utilizzo di quantificatori universali che permettono di modellizzare in modo migliore la situazione reale.

Il costo d'esecuzione degli operatori simula il consumo della batteria del robot; tale costo può essere fisso o dipendente da una funzione e viene definito mediante parametri preceduti dal simbolo `$`. Ogni operatore contiene, tra le precondizioni, un fatto che indica quanta carica deve essere presente per effettuare l'azione specificata dall'operatore e, tra gli effetti, un'istruzione che indica il costo d'esecuzione di tale azione. Ad esempio:

```
move
:v ?r robot ?s location ?a location
:p at(?r ?s) con(?s ?a) >($fuel() cost(?s ?a))
```

(la terza precondizione implica che si possa applicare l'operatore solo se la carica delle batterie è superiore al valore restituito da `cost(?s ?a)`)

```
:e ADD at(?r ?a)
   DEL at(?r ?s)
     $fuel() -= cost(?s ?a).
```

(il costo d'applicazione dell'operatore, restituito da `cost(?s ?a)`, viene sottatto al valore di `fuel()`)

La risorsa `fuel()` viene definita ed istanziata nel file .fct nel modo seguente:

```
$fuel(): 0 600;
```

(posta tra le definizioni iniziali delle variabili; fuel() può assumere un valore qualsiasi da 0 a 600)

```
=($fuel() 600)
```

(posta tra i fatti iniziali del file .fct; inizialmente il valore di `fuel()` viene posto a 600)

La funzione `cost(?s ?a)` viene definita, nel file .fct, associando un valore ad ogni connessione presente nel dominio nel modo seguente:

```
database: cost (P1 P2) 3 cost(P2 P1) 3
```

I quantificatori universali sono utilizzati per specificare effetti additivi o cancellanti relativi ad un gruppo di oggetti che soddisfano un insieme di precondizioni. In questo modo si sono potute modellizzare situazioni che non erano affrontabili con i precedenti operatori. Ad esempio aggiungendo tra gli effetti additivi:

```
ALL ?s location in-front-of(?s ?p) !eq(?s ?l) => ADD closed(?s ?p)
```

si vuole indicare che l'effetto additivo `closed(?s ?p)` verrà applicato a tutte le variabili `?s` di tipo `location` che soddisfano le condizioni `in-front-of(?s ?p)` e `!eq(?s ?l)`.

Il dominio dei corridoi è stato modificato aggiungendo un costo ad ogni operatore, inoltre si sono modificati ed aggiunti alcuni operatori come descritto in seguito.

Poiché nel dominio reale ogni stanza può avere più porte, è utile per il robot poter accedere ad una stanza mediante una porta ed uscirne mediante un'altra. Gli operatori da noi utilizzati finora permettevano al robot di uscire da una stanza attraverso una porta diversa da quella di ingresso senza chiudere quest'ultima. Per ovviare a quest'inconveniente si è aggiunto l'operatore `open-door` che utilizza un quantificatore universale:

```
open-door
:v ?r robot ?p room ?l location
:p in(?r ?p) closed(?p ?l) >($fuel() 5)
:e ADD open(?p ?l)
  $fuel() -=1;
   ALL ?s location in-front-of(?s ?p) !eq(?s ?l) => ADD closed(?s ?p).
```

Per evitare che il robot possa aprire contemporaneamente due stanze, situate una di fronte all'altra, per poi controllarne una sola per volta, abbiamo modificato l'operatore sensor-off in modo tale che la disattivazione del sensore di una stanza sia esclusiva con quella di tutti gli altri sensori:

```
sensor-off
:v ?r robot ?l location ?p room
:p at(?r ?l) in-front-of(?l ?p) closed(?p) >($fuel() 13)
:e ADD sensor-off(?p)
   DEL sensor-on(?p)
   $fuel() -=1;
   ALL ?o room in-front-of(?l ?o) !eq(?o ?p) sensor-on(?o) =>
                        ADD sensor-off(?o)
                        DEL sensor-on(?o).
```

Diversamente da quanto ipotizzato in precedenza, il robot non possiede una batteria dotata di carica illimitata, ma deve ricaricarla ogniqualvolta sia necessario con l'operatore `refuel` presso un punto di ricarica:

```
refuel
:v ?r robot ?p refuel-point
:p at(?r ?p)
:e $fuel() :=200.
```

Abbiamo inoltre aggiunto gli operatori necessari per l'interazione e il riconoscimento di una persona; se tale persona dispone di un pass, e quindi è autorizzata a circolare nella struttura, il robot la identifica e ne registra la presenza con l'operatore `person-ok`, altrimenti lancerà l'allarme tramite l'operatore `alarm-person`:

```
person-ok
:v ?r robot ?l location ?p person
:p at(?r ?l) at(?p ?l) pass(?p) >($fuel() 1)
:e ADD identified(?p)
```

```
   $fuel() -=2.

alarm-person
:v ?r robot ?l location ?p person ?a alarm
:p at(?r ?l) at(?p ?l) not-pass(?p) >($fuel() 1)
:e ADD identified(?p) alarm(?a)
   $fuel() -=2.
```

Il dominio delle stanze è stato modificato aggiungendo un costo ad ogni operatore, inoltre si sono ed aggiunti alcuni gli operatori refuel, person-ok e alarm-person già descritti sopra.

Per una descrizione completa di fatti ed operatori si veda l'appendice E oppure il CD-ROM allegato.

.

3.4.4. Eventi imprevisti

Agli eventi imprevisti trattati nei paragrafi precedenti si aggiungono qui alcuni imprevisti:

- l'incontro con persone non autorizzate (ritenuto critico in quanto il robot una volta lanciato l'allarme non può inseguire efficacemente o difendersi dall'intruso);
- la necessità di ricaricare la batteria;
- la necessità di eseguire un obiettivo prioritariamente rispetto ad un altro, ad esempio quando il centro di controllo impartisce l'ordine di eseguire un goal. In tal caso il robot deve eseguire subito tale goal e ripianificare la restante parte del piano originale che stava eseguendo.

3.4.5. Esempi di dominio

Per il dominio reale abbiamo considerato i dipartimenti di Ingegneria Elettronica, Civile e Meccanica della facoltà di Ingegneria dell'Università degli Studi di Brescia.

Nel dominio dei corridoi abbiamo considerato i corridoi presenti nei tre dipartimenti e i due ascesori esterni ai dipartimenti stessi; si è supposto che siano presenti sensori antincendio, telecamere collegate con un ipotetico centro di controllo, punti di ricarica per il robot, stanze dotate di una o più porte apribili dal robot mediante dispositivo elettronico e ascensori di collegamento tra i piani azionabili mediante un apposito dispositivo elettronico.

Nell'ambito del dominio delle stanze abbiamo considerato tre ambienti, uno per dipartimento, che corrispondono a:

- laboratorio di informatica del dipartimento di Ingegneria Elettronica;
- aula CAD del dipartimento di Ingegneria Meccanica;
- aula tesisti del dipartimento di Ingegneria Civile.

La scelta è ricaduta su queste aule in quanto contengono materiale di vario tipo (calcolatori, stampanti e plotter) e sono organizzate in modo tale da poter essere facilmente modellizzate con i nostri formalismi.

4. IL DOMINIO DEI ROBOT POSTINI

Il mondo dei robot postini è costituito da un insieme di uffici e corridoi in cui si muovono uno o più robot aventi il compito di recapitare oggetti quali lettere o pacchi.

Durante l'esecuzione di tale compito si possono verificare eventi imprevisti, ad esempio:

- la porta di un ufficio è chiusa; in tal caso il robot deve prendere le chiavi nella stanza MB (vedi per esempio la figura 11);
- una persona alla quale si deve recapitare un oggetto non è presente; in tal caso l'oggetto viene lasciato in un luogo convenuto nell'ufficio;
- le pile del robot si stanno scaricando; in tal caso il robot deve andare nella stanza RP prima di fermarsi del tutto;
- mentre compie il suo tragitto il robot incontra un ostacolo non previsto; in tal caso deve essere in grado di ripianificare il percorso per raggiungere ugualmente i suoi obiettivi.

Il dominio dei postini utilizza i seguenti operatori (file post.ops):

```
# operatore per appoggiare un oggetto
take-out
:v ?o object ?r robot ?l location
:p on-board(?o ?r) at(?r ?l)
:e ADD at(?o ?l)
   DEL  on-board(?o ?r) .

# operatore per caricare un oggetto
put-in
:v ?o object ?r robot ?l location
:p at(?o ?l) at(?r ?l)
:e ADD on-board(?o ?r)
    DEL at(?o ?l) .

# operatore per muoversi nei corridoi
move
:v  ?r robot ?m location ?l location
:p at(?r ?m) con(?m ?l) has-fuel(?r)
:e ADD at(?r ?l)
   DEL at(?r ?m).

# operatore per ricaricare il robot
refuel
:v ?r robot
:p at(?r RP)
:e ADD has-fuel(?r).

# operatore per aprire un ufficio
open-office
:v ?r robot  ?m location ?off office
:p at(?r ?m) in-front(?m ?off) !open(?off) has-key(?r ?off)
:e ADD  open(?off)
  DEL  closed(?off).

# operatore per chiudere un ufficio
close-office
:v ?r robot  ?m location ?off office
:p at(?r ?m) in-front(?m ?off) open(?off) has-key(?r ?off)
:e ADD  closed(?off)
```

```
   DEL  open(?off).

# operatore per prendere una chiave dalla stanza MB
get-key
:v ?r robot  ?off office
:p at(?r MB) has-key(MB ?off)
:e ADD has-key(?r ?off)
   DEL has-key(MB ?off).

# operatore per restituire una chiave alla stanza MB
putdown-key
:v ?r robot  ?off office
:p at(?r MB) has-key(?r ?off)
:e ADD has-key(MB ?off)
   DEL has-key(?r ?off).

# operatore per entrare in un ufficio
enter-office
:v ?r robot  ?m location ?off office
:p at(?r ?m) in-front(?m ?off) open(?off)
:e ADD  at(?r ?off)
   DEL  at(?r ?m).

# operatore per uscire da un ufficio
exit-office
:v ?r robot ?off office ?m location
:p at(?r ?off) in-front(?m ?off) open(?off)
:e ADD   at(?r ?m)
   DEL   at(?r ?off).

# operatore per dare un oggetto a una persona
give-obj
:v ?o object ?r robot ?p person ?l location
:p on-board(?o ?r) at(?r ?l) at(?p ?l)
:e ADD has(?o ?p)
   DEL   on-board(?o ?r).

# operatore per protocollare un oggetto
protocol-obj
:v ?o object
:p at(?o MB)
:e ADD protocolled(?o).
```

Abbiamo preparato tre problemi per il test delle tecniche di adattamento, denominati post1, post2 e post3. Questi problemi vengono descritti di seguito.

1) **post1**: lo stato iniziale del mondo può essere descritto graficamente come in figura 11:

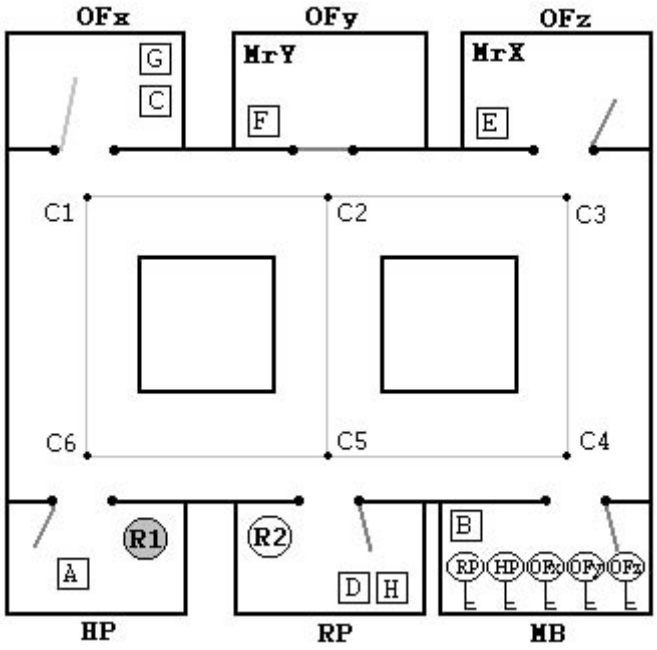


Figura 11: descrizione grafica dello stato iniziale del problema post1.

Il mondo è costituito da 2 robot (R1 R2), da 6 punti collegati per la navigazione nei corridoi (C1 … C6), da 8 oggetti da trasportare (A …H), da 3 uffici (Ofx, Ofy, Ofz), da 2 persone (MrX, MrY), dal punto di ricarica (RP), dalla stanza HP, dalla stanza delle chiavi (MB) e dalle 5 chiavi (3 per gli uffici, una per HP e una per RP).

I goal del problema consistono nel consegnare gli oggetti negli uffici o alle persone specificate nello stato finale, rappresentabile graficamente come in figura 12:

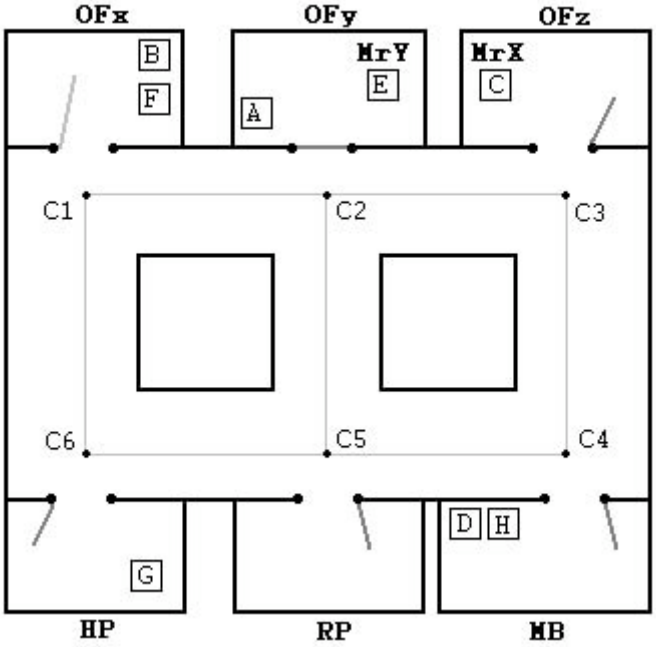


Figura 12: descrizione grafica dello stato finale del problema post1.

Stato iniziale e finale sono specificati nel file post1.fct riportato di seguito:

```
# post1.fct
# 8 objects to be transported
# entspricht rocket_facts4
# 2 robots
# 2 persons


location:  C1 C2 C3 C4 C5 C6 OFx OFy OFz MB RP HP;

office: OFx OFy OFz MB RP HP;

robot: R1 R2;

person: MrX MrY;

object: A B C D E F G H;

initial:
at(A HP) at(B MB) at(C OFx) at(D RP) at(R1 HP) has-fuel(R1)
at(E OFz) at(F OFy) at(G OFx) at(H RP) at(R2 RP)
con(C1 C2) con(C2 C1) con(C2 C3) con(C3 C2) con(C3 C4) con(C4 C3)
con(C4 C5) con(C5 C4) con(C5 C6) con(C6 C5)
con(C6 C1) con(C1 C6) con(C2 C5) con(C5 C2)
in-front(C1 OFx) in-front(C2 OFy) in-front(C3 OFz) in-front(C4 MB)
in-front(C5 RP) in-front(C6 HP)
open(OFx) open(RP) open(MB) open(OFz) open(HP)
#l'ufficio OFy e' chiuso
has-key(MB OFx) has-key(MB OFy) has-key(MB OFz) has-key(MB RP)
has-key(MB HP) at(MrX OFz) at(MrY OFy);

goal:
at(A OFy)
at(B OFx)
has(C MrX)
at(D MB)
has(E MrY)
at(F OFx)
at(G HP)
at(H MB);
```

La soluzione del problema è composta da un piano di 18 livelli contenuta nel file “post1.sol” e riportata di seguito:

```
0: refuel_R2
   put-in_D_R2_RP
   exit-office_R1_HP_C6
   put-in_H_R2_RP
1: move_R1_C6_C1
   exit-office_R2_RP_C5
2: move_R2_C5_C4
   enter-office_R1_C1_OFx
3: put-in_G_R1_OFx
   put-in_C_R1_OFx
   enter-office_R2_C4_MB
```

```
4: put-in_B_R2_MB
   take-out_D_R2_MB
   get-key_R2_OFy
   exit-office_R1_OFx_C1
   take-out_H_R2_MB
5: move_R1_C1_C6
   exit-office_R2_MB_C4
6: move_R2_C4_C3
   enter-office_R1_C6_HP
7: take-out_G_R1_HP
   move_R2_C3_C2
   put-in_A_R1_HP
8: open-office_R2_C2_OFy
   exit-office_R1_HP_C6
9: move_R1_C6_C1
   enter-office_R2_C2_OFy
10: move_R1_C1_C2
    put-in_F_R2_OFy
11: move_R1_C2_C3
    exit-office_R2_OFy_C2
12: move_R2_C2_C1
    enter-office_R1_C3_OFz
13: put-in_E_R1_OFz
    give-obj_C_R1_MrX_OFz
    enter-office_R2_C1_OFx
14: take-out_B_R2_OFx
    exit-office_R1_OFz_C3
    take-out_F_R2_OFx
15: move_R1_C3_C2
16: enter-office_R1_C2_OFy
17: give-obj_E_R1_MrY_OFy
    take-out_A_R1_OFy
```

Per i test sono state utilizzate delle varianti ottenute da questo problema base mediante aggiunte o variazioni di goal o modifiche dello stato iniziale che rappresentano i possibili imprevisti che il robot può dover affrontare; tali problemi sono elencati e descritti di seguito (a1 indica un problema ottenuto per aggiunta di un goal, a3 un problema ottenuto per modifica di un goal e b2 un problema ottenuto per modifica dello stato iniziale):

- a1v1.fct: ottenuto da post1.fct chiedendo che alla fine il robot R1 si trovi al punto di ricarica RP; è possibile ottenere un piano soluzione da post1.sol aggiungendo 4 mosse (piano risultante di 19 livelli);

- a1v2.fct: ottenuto da post1.fct chiedendo che alla fine il robot R2 si trovi al punto di ricarica RP; è possibile ottenere un piano soluzione da post1.sol aggiungendo 3 mosse e 3 livelli (piano risultante di 21 livelli), ma cambiando molto il piano si può trovare una soluzione migliore (di 19 livelli);

- a1v3.fct: ottenuto da post1.fct aggiungendo un oggetto I in MB che deve essere portato in HP; è possibile ottenere un piano soluzione da post1.sol aggiungendo 6 mosse e 3 livelli (piano risultante di 21 livelli), ma cambiando molto il piano si può trovare una soluzione migliore (di 18 livelli);

- a3v1.fct: ottenuto da post1.fct chiedendo che A sia portato a MrY anziché nell'ufficio Ofy; è possibile ottenere un piano soluzione da post1.sol modificando una mossa (piano risultante di 18 livelli);

- a3v2.fct: ottenuto da post1.fct chiedendo che B sia portato in HP anziché nell'ufficio Ofx; è possibile ottenere un piano soluzione da post1.sol togliendo una mossa e aggiungendone quattro (piano risultante di 19 livelli) ma cambiando molto il piano si può trovare una soluzione migliore (di 18 livelli);

- a3v3.fct: ottenuto da post1.fct chiedendo che A sia portato a MrX anziché nell'ufficio Ofy; è possibile ottenere un piano soluzione da post1.sol togliendo una mossa e aggiungendone una (piano risultante di 18 livelli);

- a3v4.fct: ottenuto da post1.fct chiedendo che H sia portato a MrX anziché in MB; per ottenere una soluzione a partire da post1.sol occorre apportare molte modifiche;

- b2v1.fct: ottenuto da post1.fct togliendo il fatto iniziale has-fuel(R1); è possibile trovare un piano soluzione da post1.sol aggiungendo una mossa in un livello (soluzione risultante di 19 livelli);

- b2v2.fct: ottenuto da post1.fct spostando nello stato iniziale l'oggetto C da OfX in HP; è possibile trovare un piano soluzione da post1.sol aggiungendo una mossa e togliendone una (soluzione risultante di 18 livelli), ma cambiando molto il piano si può trovare una soluzione migliore (di 16 livelli);

- b2v3.fct: ottenuto da post1.fct spostando nello stato iniziale l'oggetto B da MB in C5; è possibile trovare un piano soluzione da post1.sol aggiungendo una mossa e togliendone una (soluzione risultante di 18 livelli).

Tutti i file relativi a questo dominio sono riportati nel CD-ROM allegato.

2) **post2**: lo stato iniziale del mondo può essere descritto graficamente come in figura 13:

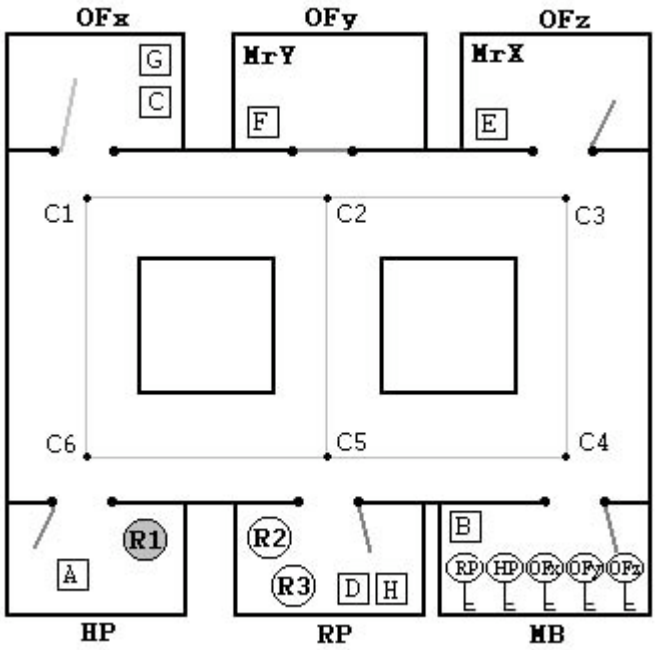


Figura 13: descrizione grafica dello stato iniziale del problema post2.

Il mondo è costituito da 3 robot (R1 R2 R3), da 6 punti collegati per la navigazione nei corridoi (C1 … C6), da 8 oggetti da trasportare (A …H), da 3 uffici (Ofx, Ofy, Ofz), da 2 persone (MrX MrY), dal punto di ricarica (RP), dalla stanza HP, dalla stanza delle chiavi (MB) e dalle 5 chiavi (3 per gli uffici, una per HP e una per RP).

I goal del problema consistono nel consegnare gli oggetti negli uffici o alle persone specificate nello stato finale, rappresentabile graficamente come in figura 14:

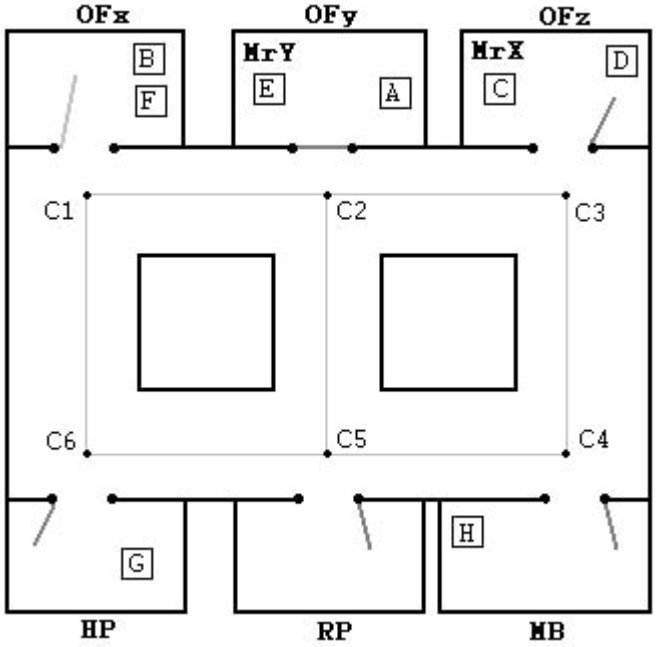


Figura 14: descrizione grafica dello stato finale del problema post2.

Stato iniziale e finale sono specificati nel file post2.fct riportato di seguito:

```
# post2.fct
# 8 objects to be transported
# entspricht rocket_facts4
# 3 robots
# 2 persons


location:  C1 C2 C3 C4 C5 C6 OFx OFy OFz MB RP HP;

office: OFx OFy OFz MB RP HP;

robot: R1 R2 R3;

person: MrX MrY;

object: A B C D E F G H;


initial:
at(A HP) at(B MB) at(C OFx) at(D RP) at(R1 HP) has-fuel(R1)
at(E OFz) at(F OFy) at(G OFx) at(H RP) at(R2 RP) at(R3 RP)
con(C1 C2) con(C2 C1) con(C2 C3) con(C3 C2) con(C3 C4) con(C4 C3)
con(C4 C5) con(C5 C4) con(C5 C6) con(C6 C5)
con(C6 C1) con(C1 C6) con(C2 C5) con(C5 C2)
in-front(C1 OFx) in-front(C2 OFy) in-front(C3 OFz) in-front(C4 MB)
in-front(C5 RP) in-front(C6 HP)
open(OFx) open(RP) open(MB) open(OFz) open(HP)
#l'ufficio OFy e' chiuso
has-key(MB OFx) has-key(MB OFy) has-key(MB OFz) has-key(MB RP)
has-key(MB HP) at(MrX OFz) at(MrY OFy);


goal:
at(A OFy)
at(B OFx)
has(C MrX)
at(D OFz)
has(E MrY)
at(F OFx)
at(G HP)
at(H MB);
```

La soluzione del problema è composta da un piano di 15 livelli contenuta nel file “post2.sol” e riportata di seguito:

```
0: refuel_R2
   put-in_H_R2_RP
   refuel_R3
   put-in_D_R3_RP
   exit-office_R1_HP_C6
1: move_R1_C6_C1
   exit-office_R3_RP_C5
   exit-office_R2_RP_C5
2: move_R2_C5_C4
```

```
   move_R3_C5_C2
   enter-office_R1_C1_OFx
3: put-in_C_R1_OFx
   put-in_G_R1_OFx
   move_R3_C2_C1
   enter-office_R2_C4_MB
4: take-out_H_R2_MB
   get-key_R2_OFy
   take-out_D_R3_C1
   exit-office_R1_OFx_C1
   put-in_B_R2_MB
5: put-in_D_R1_C1
   exit-office_R2_MB_C4
   take-out_G_R1_C1
6: put-in_G_R3_C1
   move_R2_C4_C3
   move_R1_C1_C2
7: move_R3_C1_C6
   move_R1_C2_C3
   move_R2_C3_C2
8: open-office_R2_C2_OFy
   enter-office_R1_C3_OFz
   enter-office_R3_C6_HP
9: put-in_A_R3_HP
   give-obj_C_R1_MrX_OFz
   take-out_D_R1_OFz
   put-in_E_R1_OFz
   enter-office_R2_C2_OFy
   take-out_G_R3_HP
10: put-in_F_R2_OFy
    exit-office_R1_OFz_C3
    exit-office_R3_HP_C6
11: move_R3_C6_C1
    move_R1_C3_C2
    exit-office_R2_OFy_C2
12: move_R2_C2_C1
    enter-office_R1_C2_OFy
    move_R3_C1_C2
13: enter-office_R3_C2_OFy
    give-obj_E_R1_MrY_OFy
    enter-office_R2_C1_OFx
14: take-out_F_R2_OFx
    take-out_B_R2_OFx
    take-out_A_R3_OFy
```

Per i test sono stati utilizzati dei problemi ottenuti da questo problema base mediante aggiunte o variazioni di goal o modifiche dello stato iniziale che rappresentano i possibili imprevisti che il robot può trovarsi ad affrontare; tali problemi sono elencati e descritti di seguito (a1 indica un problema ottenuto per aggiunta di un goal, a3 un problema ottenuto per modifica di un goal e b2 un problema ottenuto per modifica dello stato iniziale):

- a1v1.fct: ottenuto da post2.fct chiedendo che alla fine il robot R1 si trovi al punto di ricarica RP; è possibile ottenere un piano soluzione da post2.sol aggiungendo 3 mosse (piano risultante di 17 livelli), ma cambiando molto il piano si può trovare una soluzione migliore (di 15 livelli);

- a1v2.fct: ottenuto da post2.fct aggiungendo un oggetto I in Ofx che deve essere portato in Ofy; è possibile ottenere un piano soluzione da post2.sol aggiungendo 2 mosse (piano risultante di 15 livelli);

- a1v3.fct: ottenuto da post2.fct chiedendo che l'oggetto B venga protocollato; per ottenere una soluzione a partire da post2.sol occorre modificare molto il piano (esiste una soluzione di 15 livelli);

- a3v1.fct: ottenuto da post2.fct chiedendo che A sia portato a MrY anziché nell'ufficio Ofy; è possibile ottenere un piano soluzione da post2.sol modificando una mossa (piano risultante di 15 livelli);

- a3v2.fct: ottenuto da post2.fct chiedendo che D sia portato a MrY anziché nell'ufficio Ofz; è possibile ottenere un piano soluzione da post2.sol togliendo una mossa e aggiungendone una (piano risultante di 15 livelli);

- a3v3.fct: ottenuto da post2.fct chiedendo che F sia portato in HP anziché nell'ufficio Ofx; è possibile ottenere un piano soluzione da post2.sol togliendo una mossa e aggiungendone quattro (piano risultante di 19 livelli), ma cambiando molto il piano è possibile trovare una soluzione migliore (di 16 livelli);

- a3v4.fct: ottenuto da post2.fct chiedendo che H sia portato a MrX anziché in MB; per ottenere una soluzione a partire da post2.sol occorre apportare molte modifiche (esiste una soluzione in 15 livelli);

- b2v1.fct: ottenuto da post2.fct chiudendo la porta di Ofz nello stato iniziale; per ottenere una soluzione a partire da post2.sol occorre apportare molte modifiche (esiste una soluzione in 15 livelli);

- b2v2.fct: ottenuto da post2.fct togliendo il fatto has-fuel(R1) nello stato iniziale; è possibile trovare un piano soluzione da post2.sol aggiungendo una mossa (soluzione risultante di 16 livelli);

- b2v3.fct: ottenuto da post2.fct spostando nello stato iniziale l'oggetto D da RP in Ofx; è possibile trovare un piano soluzione da post2.sol aggiungendo una mossa e togliendone tre (soluzione risultante di 15 livelli).

Tutti i file relativi a questo dominio sono riportati nel CD-ROM allegato.

3) **post3**: lo stato iniziale del mondo può essere descritto graficamente come in figura 15:

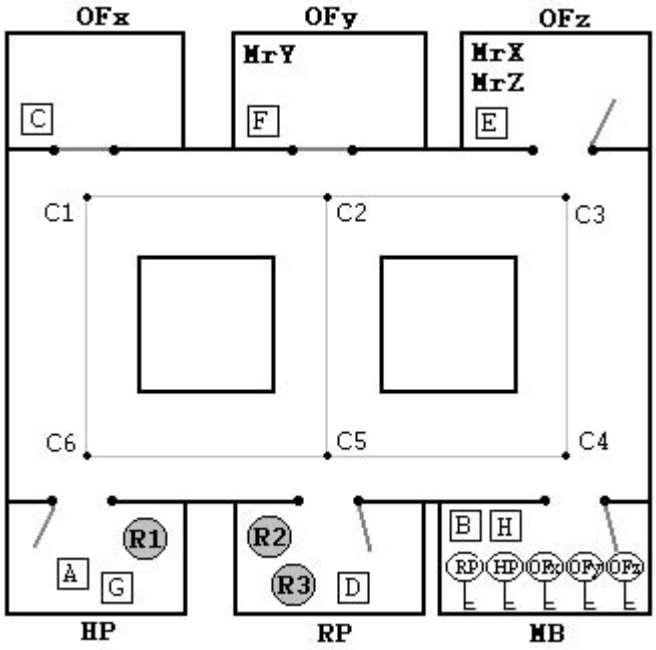


Figura 15: descrizione grafica dello stato iniziale del problema post3.

Il mondo è costituito da 3 robot (R1 R2 R3), da 6 punti collegati per la navigazione nei corridoi (C1 … C6), da 8 oggetti da trasportare (A …H), da 3 uffici (Ofx, Ofy, Ofz), da 3 persone (MrX MrY MrZ), dal punto di ricarica (RP), dalla stanza HP, dalla stanza delle chiavi (MB) e dalle 5 chiavi (3 per gli uffici, una per HP e una per RP).

I goal del problema consistono nel consegnare gli oggetti negli uffici o alle persone specificate nello stato finale, rappresentabile graficamente come in figura 16:

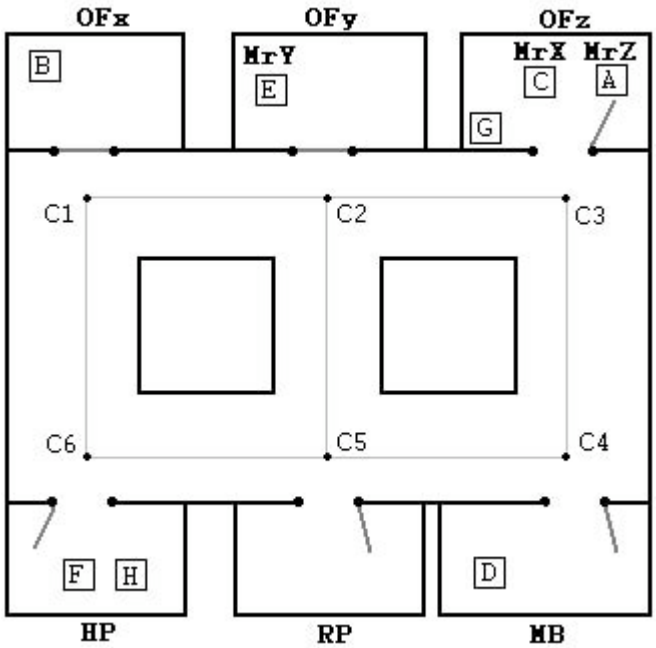


Figura 16: descrizione grafica dello stato finale del problema post3.

Stato iniziale e finale sono specificati nel file post3.fct riportato di seguito:

```
# post3.fct
# 8 objects to be transported
# entspricht rocket_facts4
# 3 robots
# 3 persons

location:  C1 C2 C3 C4 C5 C6 OFx OFy OFz MB RP HP;

office: OFx OFy OFz MB RP HP;

robot: R1 R2 R3;

person: MrX MrY MrZ;

object: A B C D E F G H;


initial:
at(A HP) at(B MB) at(C OFx) at(D RP) at(R1 HP) has-fuel(R1)
at(E OFz) at(F OFy) at(G HP) at(H MB) at(R2 RP) has-fuel(R2)
at(R3 RP) has-fuel(R3)
con(C1 C2) con(C2 C1) con(C2 C3) con(C3 C2) con(C3 C4) con(C4 C3)
con(C4 C5) con(C5 C4) con(C5 C6) con(C6 C5)
con(C6 C1) con(C1 C6) con(C2 C5) con(C5 C2)
in-front(C1 OFx) in-front(C2 OFy) in-front(C3 OFz) in-front(C4 MB)
in-front(C5 RP) in-front(C6 HP)
open(RP) open(MB) open(OFz) open(HP)
has-key(MB OFx) has-key(MB OFy) has-key(MB OFz) has-key(MB RP)
has-key(MB HP) at(MrX OFz) at(MrY OFy) at(MrZ OFz);
goal:
has(A MrZ)
at(B OFx)
has(C MrX)
at(D MB)
has(E MrY)
at(F HP)
at(G OFz)
protocolled(G)
at(H HP);
```

La soluzione del problema è composta da un piano di 16 livelli contenuta nel file “post3.sol” e riportata di seguito:

```
0: put-in_A_R1_HP
   exit-office_R3_RP_C5
   put-in_D_R2_RP
   put-in_G_R1_HP
1: exit-office_R1_HP_C6
   exit-office_R2_RP_C5
   move_R3_C5_C4
2: enter-office_R3_C4_MB
   move_R2_C5_C4
   move_R1_C6_C5
3: move_R1_C5_C4
```

```
   enter-office_R2_C4_MB
   get-key_R3_OFx
   put-in_B_R3_MB
4: exit-office_R3_MB_C4
   take-out_D_R2_MB
   enter-office_R1_C4_MB
   put-in_H_R2_MB
   get-key_R2_OFy
5: take-out_G_R1_MB
   exit-office_R2_MB_C4
   move_R3_C4_C3
6: move_R3_C3_C2
   move_R2_C4_C3
   protocol-obj_G
7: put-in_G_R1_MB
   move_R2_C3_C2
   move_R3_C2_C1
8: open-office_R3_C1_OFx
   exit-office_R1_MB_C4
   open-office_R2_C2_OFy
9: move_R1_C4_C3
   enter-office_R2_C2_OFy
   enter-office_R3_C1_OFx
10: take-out_B_R3_OFx
    put-in_C_R3_OFx
    put-in_F_R2_OFy
    enter-office_R1_C3_OFz
11: exit-office_R2_OFy_C2
    take-out_G_R1_OFz
    put-in_E_R1_OFz
    exit-office_R3_OFx_C1
    give-obj_A_R1_MrZ_OFz
12: move_R3_C1_C2
    exit-office_R1_OFz_C3
    move_R2_C2_C5
13: move_R2_C5_C6
    move_R1_C3_C2
    move_R3_C2_C3
14: enter-office_R3_C3_OFz
    enter-office_R1_C2_OFy
    enter-office_R2_C6_HP
15: take-out_H_R2_HP
    take-out_F_R2_HP
    give-obj_E_R1_MrY_OFy
    give-obj_C_R3_MrX_OFz
```

Per i test sono stati utilizzati dei problemi ottenuti da questo problema base mediante aggiunte o variazioni di goal o modifiche dello stato iniziale che rappresentano i possibili imprevisti che il robot può trovarsi ad affrontare; tali problemi sono elencati e descritti di seguito (a1 indica un problema ottenuto per aggiunta di un goal, a3 un problema ottenuto per modifica di un goal e b2 un problema ottenuto per modifica dello stato iniziale):

- a1v1.fct: ottenuto da post3.fct aggiungendo un oggetto I in RP che deve essere portato in Ofz; è possibile ottenere un piano soluzione da post3.sol aggiungendo 2 mosse (piano risultante di 16 livelli);

- a1v2.fct: ottenuto da post3.fct chiedendo che l'oggetto A venga protocollato; per ottenere una soluzione a partire da post3.sol occorre aggiungere tre mosse (soluzione risultante di 16 livelli);

- a1v3.fct: ottenuto da post3.fct chiedendo che alla fine il robot R1 si trovi al punto di ricarica RP; è possibile ottenere un piano soluzione da post3.sol aggiungendo 3 mosse (piano risultante di 19 livelli);

- a3v1.fct: ottenuto da post3.fct chiedendo che A sia portato a Ofy anziché a MrZ; è possibile ottenere un piano soluzione da post3.sol aggiungendo una mossa e togliendone una (piano risultante di 16 livelli);

- a3v2.fct: ottenuto da post3.fct chiedendo che G sia portato a MrZ anziché nell'ufficio Ofz; è possibile ottenere un piano soluzione da post3.sol modificando una (piano risultante di 16 livelli);

- a3v3.fct: ottenuto da post3.fct chiedendo che C sia portato in MB anziché nell'ufficio Ofz; è possibile ottenere un piano soluzione da post3.sol togliendo una mossa e aggiungendone quattro (piano risultante di 20 livelli), ma cambiando molto il piano è possibile trovare una soluzione migliore (di 16 livelli);

- a3v4.fct: ottenuto da post3.fct chiedendo che F sia portato in RP anziché in HP; per ottenere una soluzione a partire da post3.sol occorre apportare molte modifiche al piano;

- b2v1.fct: ottenuto da post3.fct chiudendo la porta di Ofz nello stato iniziale; per ottenere una soluzione a partire da post3.sol occorre apportare molte modifiche (esiste una soluzione in 17 livelli);

- b2v2.fct: ottenuto da post3.fct togliendo il fatto has-fuel(R2) nello stato iniziale; è possibile trovare un piano soluzione da post3.sol aggiungendo una mossa (soluzione risultante di 16 livelli);

- b2v3.fct: ottenuto da post3.fct spostando nello stato iniziale l'oggetto A da HP in MB; è possibile trovare un piano soluzione da post3.sol aggiungendo una mossa e togliendone una (soluzione risultante di 16 livelli).

Tutti i file relativi a questo dominio sono riportati nel CD-ROM allegato.

4.1. Risultati sperimentali

Presentiamo di seguito i risultati dei test con il metodo di ricerca sistematica e con il metodo di ricerca locale e sistematica per i tre domini sopra presentati (tabelle 14, 15 e 16); tutti i tempi sono indicati in secondi.

In ogni tabella vengono riportati, per ogni problema di ogni dominio:

1) i dati della ricerca sistematica (ADJUST-PLAN), costituiti dal tempo impiegato per trovare la prima soluzione (Time1) e il relativo numero di livelli (L1), e dal tempo impiegato per trovare la seconda soluzione (T2) e il relativo numero di livelli (L2).
   Un segno --- per tutti i dati di ADJUST-PLAN indica che il problema non è stato risolto, mentre un segno --- per T2 e L2 indica che un secondo piano non è stato trovato entro 180 secondi dal momento in cui il programma è stato lanciato;
2) i dati della ricerca locale e sistematica (T-Walkplan+ADJUST), costituiti dalla media del tempo totale impiegato per trovare la soluzione (Time; dato da T-L + T-A*(l)/10), dal tempo medio su 10 prove impiegato dalla ricerca locale per trovare una soluzione definitiva o con un numero di inconsistenze inferiore a quello limite (T-L), dal tempo medio sul numero di prove indicato tra parentesi (l) impiegato dalla ricerca sistematica per trovare una soluzione definitiva partendo dal piano dato dalla ricerca locale (T-A; un segno --- indica che il problema era già stato risolto dalla ricerca locale) e dal numero medio di livelli della soluzione finale (Lev). Un segno --- per tutti i dati di T-Walkplan+ADJUST indica che il problema non è stato risolto;
3) il tempo e il numero di livelli in cui IPP risolve il problema effettuando una ripianificazione completa (T e L).

Post_1	**ADJUST-PLAN**				**T-Walkplan+ADJUST**				**IPP**	
	Time1	L1	T2	L2	Time	T-L	T-A (l)	Lev	T	L
a1v1	1.44	20	3.04	19	2,994	1,619	1,375 (10)	20	465,90	19
a1v2	1.48	21	3.18	20	3,053	1,647	1,406 (10)	21	364,85	19
a1v3	4.28	24	11.2	23	5,847	1,788	4,059 (10)	24	830,69	18
a3v1	1.38	18	---	---	1,259	1,259	0	18	474,05	18
a3v2	1.48	21	3.18	20	2,995	1,585	1,41 (10)	21	190,97	18
a3v3	1.48	21	3.18	20	1.822	1,402	1,4 (3)	18,9	578,35	18
a3v4	2.14	23	20.09	22	3,651	1,609	2,042 (10)	23	---	---
b2v1	4.9	27	---	---	5,887	1,192	4,695 (10)	27,5	---	---
b2v2	3.17	23	5.99	20	2.659	1,469	2,975 (4)	20	20,99	16
b2v3	1.96	20	3.77	18	3,262	1,602	1,66 (10)	22	476,98	18

Tabella 14: risultati dei test del metodo di ricerca sistematica e locale + sistematica per il dominio Post_1.

Post_2	**ADJUST-PLAN**				**T-Walkplan+ADJUST**				**IPP**	
	Time1	L1	T2	L2	Time	T-L	T-A (l)	Lev	T	L
a1v1	1.74	17	4.03	16	3,424	1,745	1,679 (10)	17	20,19	15
a1v2	---	---	---	---	---	---	---	---	968,59	15
a1v3	---	---	---	---	1,399	1,399	0	15	580,40	15
a3v1	1.73	15	---	---	1,382	1,382	0	15	550,59	15
a3v2	2.68	19	5.56	17	1,405	1,405	0	15	516,28	15
a3v3	2.19	18	4.75	17	3,965	1,954	2,011 (10)	19	107,81	16
a3v4	3.63	19	---	---	2,182	1,487	3,475 (2)	15,8	556,04	15
b2v1	3	19	8.05	17	7,724	1,605	6,119 (10)	21	20,29	15
b2v2	3.96	23	56.07	21	5,365	1,237	6,88 (6)	20,5	---	---
b2v3	4.6	20	---	---	1,657	1,414	2,43 (1)	15,4	346,85	15

Tabella 15: risultati dei test del metodo di ricerca sistematica e locale + sistematica per il dominio Post_2.

Post_3	ADJUST-PLAN				T-Walkplan+ADJUST				IPP	
	Time1	L1	T2	L2	Time	T-L	T-A (l)	Lev	T	L
a1v1	---	---	---	---	---	---	---	---	---	---
a1v2	68.51	21	120.51	20	1,566	1,566	0	16	599,73	16
a1v3	2.19	19	---	---	4,016	1,897	2,119 (10)	19	---	---
a3v1	4.26	16	---	---	3,384	1,746	4,095 (4)	16	524,89	16
a3v2	1.97	17	4.52	16	2,497	1,739	1,895 (4)	16,4	645,22	16
a3v3	2.92	20	7.28	20	1,569	1,569	0	16	908,72	16
a3v4	2.69	19	---	---	4,507	1,913	2,594 (10)	19	---	---
b2v1	---	---	---	---	7,068	1,862	6,5075 (8)	20,8	105,22	17
b2v2	12.07	19	98.26	16	23,694	2,04	24,06 (9)	21,9	89,15	16
b2v3	3.39	19	12.77	17	1,577	1,577	0	16	629,39	16

Tabella 16: risultati dei test del metodo di ricerca sistematica e locale + sistematica per il dominio Post_3.

Da questi risultati si può notare come il metodo di adattamento mediante ricerca sistematica sia molto più vantaggioso in termini di tempo rispetto ad una ripianificazione completa tramite IPP; il numero di livelli della soluzione non è però sempre vicino all'ottimo. Il metodo di adattamento mediante combinazione di ricerca locale e sistematica risulta invece essere vantaggioso sia in termini di tempo che il termini di qualità della soluzione rispetto alla pianificazione completa tramite IPP.

APPENDICE A

Listati delle procedure d'adattamento automatico dei parametri negli algoritmi di ricerca locale

/* the following two functions, set_param_minimo and set_param_varianza, are used to update walk.numerator and walk.tabu_lenght; the function actually used is the one called from the line 1888 of this module */

```
void  set_param_minimo(int  num_unsat)/*  this  function  updates
                                        walk.numerator and walk.tabu_lenght
                                        (it  doubles  them)  testing  the
                                        minimum    of    unsatisfied
                                        preconditions of added or removed
                                        actions */
   {
   static int position=0, primo = 1, min_prec=MAXINT;
    static  int  nvolte  =  2;    /* "nvolte" contains the maximum number
                                  of  times  the  function  is  allowed  to
                                  double the values of walk.numerator and
                                  walk.tabu_lenght  */
   int i, min;
   static int unsat_vector[UNS_VECT];
                  /* every time the function is called, the number of
               unsatisfied preconditions ("num_unsat") is put into the
               position "position" of this vector. When "position"
               becomes equal to UNS_VECT, the minimum of the stored
               values is calculated and compared with the minimum
               calculated in the previous iteration. If the present
               minimum is greater or equal then the previous one the
               values of "walk.numerator" and "walk.tabu_lenght" are
               doubled, (but only if this isn't already happened
               "nvolte" times) otherwise they are restored to the
               values initially passed to ipp through options -N and -L
               respectively */

   if(position< (UNS_VECT -1))
          unsat_vector[position++]=num_unsat;
                /* the vector is not full yet */
   else
   {
      unsat_vector[position]=num_unsat;
                  /* the vector is full and it's time to see whether
               update "walk.numerator" and "walk.tabu_lenght" or not */
      position=0;                /* reset for a new iteration */
      min=unsat_vector[0];
      for(i=1; i< UNS_VECT; i++)
      {
         if(unsat_vector[i]<min)
            min=unsat_vector[i];
      }

      if(walk.stampa>0)
```

```
    {
            printf("\n     min     %d     N     %d,     L     %d
        \n\n",min,walk.numerator,walk.tabu_lenght);
    }    /* used for debugging; walk.stampa is set through option -A */

     if(!primo)     /* the first time the minimum is calculated it
                    doesn't make sense to update the parameters,
                    because we haven't at our disposal a previous value
                    */
    {
       if(min<min_prec)
       {
          walk.numerator=walk.numerator_ORIG
          walk.tabu_lenght=walk.tabu_length_ORIG; ;
                      /* restored to initial values */

       }
       else
              if(walk.numerator_ORIG * 2.0 * nvolte > walk.numerator
          || walk.tabu_length_ORIG * 2.0 * nvolte > walk.tabu_lenght)
          {
             if (walk.stampa==1) printf("CHANGE \n");
                      /* used for debugging; walk.stampa is set
                    through option -A */
             walk.numerator*=2.0;
             walk.tabu_lenght*=2.0;
                            primo=1;    /* when the parameters are updated,
                              "primo" is set to 1; this is because we
                              want to compare two minimum values only
                              if they have been calculated on the same
                              conditions as regards to the values of
                              "walk.numerator" and "walk.tabu_lenght"
                              */
          }
    }
    else
           primo = 0;    /* to allow comparison of minimum values the
                         next time */
     min_prec = min;     /* the present value is stored for the next
                         time */
  }
  }



void  set_param_varianza(int  num_unsat) /*  this  function  updates
                                         "walk.numerator"    and
                                         "walk.tabu_lenght" (it triples
                                         them) testing the variance of
                                         unsatisfied preconditions of
                                         added or removed actions */
  {
  static int position=0;
  float mean, var, diff;
  int i, min, max;
```

```
 static int unsat_vector[UNS_VECT];     /* every time the function is
                                           called, the number of
                                           unsatisfied preconditions
                                           ("num_unsat") is put into the
                                           position "position" of this
                                           vector. "position" becomes
                                           equal to UNS_VECT, the
                                           variance of the stored values
                                           is calculated and compared
                                           with a predefined value (3.0).
                                           If the variance is less or
                                           equal then 3.0 the values of
                                           "walk.numerator" and
                                           "walk.tabu_lenght" are
                                           triplied, otherwise they are
                                           restored to the values
                                           initially passed to ipp
                                           through options -N and -L
                                           respectively */
if(position< (UNS_VECT -1))
    unsat_vector[position++]=num_unsat;      /* the vector is not
                                                full yet */
else
{
    unsat_vector[position]=num_unsat;        /* the vector is full
                                                and it's time to update
                                                "walk.numerator" and
                                                "walk.tabu_lenght" */
   position=0;                          /* reset for a new iteration */
   mean=min=max=unsat_vector[0];
   for( i=1; i< UNS_VECT; i++ )
      mean+=unsat_vector[i];
      mean  /=  (UNS_VECT);             /* this is the average of values
                                           stored in the vector */
   diff=  unsat_vector[0] - mean;
   var=pow(diff, 2.0);
   for( i=1; i< UNS_VECT; i++)
   {
      if(unsat_vector[i]<min)
         min=unsat_vector[i];
      if(unsat_vector[i]>max)
         max=unsat_vector[i];
      diff= unsat_vector[i] - mean;
      var+=pow(diff, 2.0);
   }
       var  /=  UNS_VECT;               /* this is the variance of the
                                           values stored in the vector */
   if(walk.stampa>0){
           printf("\nMean %3.3f, var %4.3f, min %d , max %d ", mean,
        var, min, max);
   printf(" N %d, L %d \n\n",walk.numerator,walk.tabu_lenght);}
           /* used for debugging; walk.stampa is set through option -A
        */
   if(var>3.0)
   {
      walk.numerator=walk.numerator_ORIG;
      walk.tabu_lenght=walk.tabu_length_ORIG;
```

```
                                    /* restored to initial values */
    }
    else
    {
       if(walk.stampa==1) printf("CHANGE \n");
            /* used for debugging; walk.stampa is set through option -A
              */
       /* getchar(); */
       walk.numerator*=3.0;
       walk.tabu_lenght*=3.0;
    }
}
}
```

APPENDICE B

Descrizione dei domini rocket, logistic e bwlarge

La specifica dei domini viene effettuata descrivendo i tipi di oggetti coinvolti e le operazioni che possono essere eseguite su di essi in notazione strips; la specifica dei problemi viene effettuata descrivendo i fatti che rappresentano la situazione iniziale e i fatti che rappresentano la situazione finale che si vuole raggiungere.

Le operazioni possibili sono elencate in un file di estensione '.ops', mentre la situazione iniziale e finale sono elencate in un file di estensione '.fct' .

Le operazioni sono descritte in notazione strips: per ognuna vengono specificate le variabili utilizzate (:v), le precondizioni che devono essere verificate perché l'azione possa essere eseguita (:p) e gli effetti additivi (ADD) e cancellanti (DEL) prodotti dall'azione (:e).

Presentiamo nel seguito la descrizione dei domini utilizzati per i test.

RocketWorld

Il mondo di rocket è composto da luoghi (identificati da variabili di tipo `place`) tra i quali possono volare alcuni razzi (identificati da variabili di tipo `rocket`) per trasportare alcuni oggetti (identificati da variabili di tipo `cargo`). La posizione iniziale o finale degli oggetti è specificata da proposizioni del tipo `at(obj place)`; analogamente la posizione di un rocket è specificata da una proposizione del tipo `at(rocket place)`. Per potersi muovere da un luogo ad un altro il razzo deve essere rifornito di carburante; nella base di conoscenza deve sempre essere presente una delle due proposizioni `no-fuel(rocket)` o `has-fuel(rocket)`, che indicano rispettivamente che il razzo non ha carburante (nel qual caso si dovrà applicare l'operatore `refuel` prima di farlo muovere) o che il razzo ha carburante (ed è quindi possibile farlo muovere tramite l'operatore `move`).

Gli operatori utilizzati da questo dominio sono specificati nel file rocket.ops e riportati di seguito:

```
# operatore per caricare un oggetto in un rocket (nello stesso posto)
Load
:v ?object cargo ?rocket rocket ?place place
:p at(?rocket ?place)  at(?object ?place)
:e ADD in(?object ?rocket)
   DEL at(?object ?place).

# operatore per scaricare un oggetto da un rocket
Unload
:v ?object cargo ?rocket rocket ?place place
:p at(?rocket ?place)  in(?object ?rocket)
:e ADD at(?object ?place)
   DEL in(?object ?rocket).
```

```
# operatore per muovere un rocket da un posto ad un altro
Move
:v ?rocket rocket ?from place ?to place
:p has-fuel(?rocket) at(?rocket ?from)
:e ADD at(?rocket ?to) no-fuel(?rocket)
   DEL has-fuel(?rocket) at(?rocket ?from).

# operatore per il rifornimento di un rocket
Refuel
:v ?rocket rocket
:p no-fuel(?rocket)
:e ADD has-fuel(?rocket)
   DEL no-fuel(?rocket).
```

In questo dominio si sono considerati due problemi di prova, rocket_a e rocket_b.

Rocket_a è specificato nel file 'rocket_a.fct':

```
place: london paris jfk bos;
rocket: r1 r2;
cargo: mxf avrim alex jason pencil paper april michelle betty lisa;

initial: at(r1 jfk)
         at(r2 bos)
         has-fuel(r1)
         has-fuel(r2)
         at(mxf paris) at(avrim paris)  at(alex paris)  at(jason jfk)
         at(pencil london) at(paper london)  at(michelle london)
         at(april london) at(betty london) at(lisa london);

goal:  at(mxf bos) at(avrim jfk) at(pencil bos) at(alex jfk)
       at(april bos) at(lisa paris) at(michelle jfk) at(jason bos)
       at(paper paris) at(betty jfk);
```

Rocket_b è specificato nel file 'rocket_b.fct':

```
place: london paris jfk bos;
rocket: r1 r2;
cargo: mxf avrim alex jason pencil paper april michelle betty lisa;

initial: at(r1 jfk)
         at(r2 paris)
         has-fuel(r1)
         has-fuel(r2)
         at(mxf jfk) at(avrim paris)  at(alex bos)  at(jason jfk)
         at(pencil paris) at(paper london)  at(michelle bos)
         at(april paris) at(betty london) at(lisa london);

goal:    at(mxf bos) at(avrim jfk) at(pencil bos) at(alex jfk)
         at(april bos) at(lisa paris) at(michelle jfk) at(jason bos)
         at(paper paris) at(betty jfk);
```

In appendice D è possibile trovare uno schema dettagliato di tutte le modifiche fatte ai problemi base, con i livelli della soluzione determinata da GPG e la difficoltà prevista per l'adattamento del piano.

Logistic

Il mondo di logistic è composto da città (identificate da variabili di tipo `city`) nelle quali è presente un aeroporto (identificato da una variabile di tipo `airport` e `location`) e da altri luoghi (identificati da variabili di tipo `location`) tra i quali un camion (identificato da una variabile di tipo `truck`) può muoversi e trasportare oggetti. Tra i luoghi, che sono anche aeroporti, possono volare alcuni aeroplani (identificati da variabili di tipo `airplane`) per trasportare alcuni oggetti (identificati da variabili di tipo `object`). La posizione iniziale o finale degli oggetti è specificata da proposizioni del tipo `at(obj location)`; analogamente la posizione di un aeroplano è specificata da una proposizione del tipo `at(airplane airport)`. Per ogni locazione occorre specificare in quale città si trova tramite una proposizione del tipo `loc-at(location city)`.

Gli operatori utilizzati sono specificati nel file logistic.ops e riportati di seguito:

```
# operatore per caricare un oggetto su un camion (nello stesso posto)
Load-truck
:v ?obj object ?truck truck ?loc location
:p at( ?truck ?loc ) at( ?obj ?loc )
:e ADD in( ?obj ?truck )
   DEL at( ?obj ?loc ).

# operatore per caricare un oggetto su un aeroplano (nello stesso
posto)
Load-airplane
:v ?obj object ?plane airplane ?loc location
:p at( ?plane ?loc ) at( ?obj ?loc )
:e ADD in( ?obj ?plane )
   DEL at( ?obj ?loc ).

# operatore per scaricare un oggetto da un camion
Unload-truck
:v ?obj object ?truck truck ?loc location
:p at( ?truck ?loc ) in( ?obj ?truck )
:e ADD at( ?obj ?loc )
   DEL in( ?obj ?truck ).

# operatore per scaricare un oggetto da un aeroplano
Unload-airplane
:v ?obj object ?plane airplane ?loc location
:p at( ?plane ?loc ) in( ?obj ?plane )
:e ADD at( ?obj ?loc )
   DEL in( ?obj ?plane ).

# operatore per muovere un camion da un posto ad un altro nella
# stessa città
Drive-truck
:v ?truck truck ?loc-f ?loc-t location ?city city
:p at( ?truck ?loc-f ) loc-at( ?loc-f ?city ) loc-at( ?loc-t ?city )
```

```
:e ADD at( ?truck ?loc-t )
   DEL at( ?truck ?loc-f ).

# operatore per far volare un aeroplano da una città ad un'altra
Fly-airplane
:v ?plane airplane ?loc-f ?loc-t airport
:p at( ?plane ?loc-f )
:e ADD at( ?plane ?loc-t )
   DEL at( ?plane ?loc-f ).
```

In questo dominio si sono considerati tre problemi di prova, logistic_a, logistic_b e logistic_c.

Logistic_a è specificato nel file 'loga.fct':

```
object  : package1 package2 package3 package4 package5 package6
          package7 package8;
truck   : pgh-truck bos-truck la-truck;
airplane: airplane1 airplane2;
location: bos-po la-po pgh-po bos-airport pgh-airport la-airport;
airport : bos-airport pgh-airport la-airport;
city    : pgh bos la;

initial: at(package1 pgh-po) at(package2 pgh-po) at(package3 pgh-po)
         at(package4 pgh-po) at(package5 bos-po) at(package6 bos-po)
         at(package7 bos-po) at(package8 la-po)
         at(airplane1 pgh-airport) at(airplane2 pgh-airport)
         at(bos-truck bos-po) at(pgh-truck pgh-po) at(la-truck la-po)
         loc-at(pgh-po pgh) loc-at(pgh-airport pgh)
         loc-at(bos-po bos) loc-at(bos-airport bos)
         loc-at(la-po la) loc-at(la-airport la);

goal: at(package1 bos-po) at(package2 bos-airport) at(package3 la-po)
      at(package4 la-airport) at(package5 pgh-po)
      at(package6 pgh-airport) at(package7 pgh-po)
      at(package8 pgh-po);
```

Logistic_b è specificato nel file 'logb.fct':

```
object  : package1 package2 package3 package5 package7;
truck   : pgh-truck bos-truck la-truck ny-truck;
airplane: airplane1 airplane2;
location: bos-po la-po pgh-po ny-po bos-airport pgh-airport
          la-airport ny-airport;
airport : bos-airport pgh-airport la-airport ny-airport;
city    : pgh bos la ny;

initial: at(package1 pgh-po) at(package2 pgh-po) at(package3 pgh-po)
         at(package5 bos-po)
         at(package7 ny-po)
         at(airplane1 pgh-airport) at(airplane2 pgh-airport)
         at(bos-truck bos-po) at(pgh-truck pgh-po) at(la-truck la-po)
         at(ny-truck ny-po)
         loc-at(pgh-po pgh) loc-at(pgh-airport pgh)
         loc-at(bos-po bos) loc-at(bos-airport bos)
```

```
          loc-at(la-po la) loc-at(la-airport la)
          loc-at(ny-po ny) loc-at(ny-airport ny);

goal: at(package1 bos-po) at(package2 ny-po) at(package3 la-po)
      at(package5 pgh-po)
      at(package7 pgh-po);
```

Logistic_c è specificato nel file 'logc.fct':

```
object  : package1 package2 package3 package4 package5 package6
          package7;
truck   : pgh-truck bos-truck la-truck ny-truck;
airplane: airplane1 airplane2;
location: bos-po la-po pgh-po ny-po bos-airport pgh-airport
          la-airport ny-airport;
airport : bos-airport pgh-airport la-airport ny-airport;
city    : pgh bos la ny;

initial: at(package1 pgh-po) at(package2 pgh-po) at(package3 pgh-po)
         at(package4 pgh-po) at(package5 bos-po)
         at(package6 bos-po) at(package7 ny-po)
         at(airplane1 pgh-airport) at(airplane2 pgh-airport)
         at(bos-truck bos-po) at(pgh-truck pgh-po) at(la-truck la-po)
         at(ny-truck ny-po)
         loc-at(pgh-po pgh) loc-at(pgh-airport pgh)
         loc-at(bos-po bos) loc-at(bos-airport bos)
         loc-at(la-po la) loc-at(la-airport la)
         loc-at(ny-po ny) loc-at(ny-airport ny);

goal: at(package1 bos-po) at(package2 ny-po) at(package3 la-po)
      at(package4 la-airport) at(package5 pgh-po)
      at(package6 ny-airport) at(package7 pgh-po);
```

In appendice D è possibile trovare uno schema dettagliato di tutte le modifiche fatte ai problemi base, con i livelli della soluzione ottima determinata da GPG e la difficoltà prevista per l'adattamento del piano.

BlocksWorld

Il mondo di BwLarge è costituito da un braccio meccanico che può spostare, uno per volta, alcuni blocchi (identificati da variabili di tipo `object`) che possono trovarsi sul tavolo oppure su altri blocchi. Lo scopo è partire da una certa configurazione iniziale dei blocchi (specificata da un insieme di proposizioni del tipo `on-table(object)`, `on(object1 object2)`, `clear(object)`) per arrivare ad un'altra configurazione finale (specificata analogamente a quella iniziale). Per prendere un blocco il braccio deve essere libero (fatto specificato dalla proposizione `arm-empty()`), e per appoggiare un blocco su di un altro quest'ultimo deve essere libero (fatto specificato da una proposizione del tipo `clear(object)`)

Gli operatori utilizzati sono descritti nel file blocks.ops e sono riportati di seguito:

```
# operatore per prendere un blocco dal tavolo
```

```
Pickup
:v ?ob object
:p clear(?ob) on-table(?ob) arm-empty()
:e ADD holding(?ob)
   DEL clear(?ob) on-table(?ob) arm-empty().

# operatore per appoggiare un blocco sul tavolo
Putdown
:v ?ob object
:p holding(?ob)
:e ADD clear(?ob) arm-empty() on-table(?ob)
   DEL holding(?ob).

# operatore per appoggiare un blocco su un altro blocco
Stack
:v ?ob object ?underob object
:p  clear(?underob) holding(?ob)
:e  ADD arm-empty() clear(?ob) on(?ob ?underob)
    DEL  clear(?underob) holding(?ob).

# operatore per prendere un blocco che si trova su un altro blocco
Unstack
:v ?object object ?underob object
:p on(?object ?underob) clear(?object) arm-empty()
:e ADD holding(?object) clear(?underob)
   DEL on(?object ?underob) clear(?object) arm-empty().
```

In questo dominio si sono considerati due problemi di prova, BwLarge_a e BwLarge_b.

BwLarge_a è specificato nel file 'bwla.fct':

```
object: blocka blockb blockc blockd blocke blockf blockg blockh
blocki;

initial: on-table(blocka)
         on(blockb blocka)
         on(blockc blockb)
         clear(blockc)
         on-table(blockd)
         on(blocke blockd)
         clear(blocke)
         on-table(blockf)
         on(blockg blockf)
         on(blockh blockg)
         on(blocki blockh)
         clear(blocki)
         arm-empty();

goal: on(blocka blocke)
      on(blocki blockd)
      on(blockh blocki)
      on(blockc blockg)
      on(blockb blockc);
```

BwLarge_b è specificato nel file 'bwlb.fct':

```
object: blocka blockb blockc blockd blocke blockf blockg blockh
blocki
        blockj blockk;

initial: on-table(blocka)
         on(blockb blocka)
         on(blockc blockb)
         clear(blockc)
         on-table(blockd)
         on(blocke blockd)
         on(blockj blocke)
         on(blockk blockj)
         clear(blockk)
         on-table(blockf)
         on(blockg blockf)
         on(blockh blockg)
         on(blocki blockh)
         clear(blocki)
         arm-empty() ;

goal: on(blocke blockj)
      on(blocka blocke)
      on(blocki blockd)
      on(blockh blocki)
      on(blockc blockk)
      on(blockk blockg)
      on(blockb blockc);
```

In appendice D è possibile trovare uno schema dettagliato di tutte le modifiche fatte ai problemi base, con i livelli della soluzione ottima determinata da GPG e la difficoltà prevista per l'adattamento del piano.

APPENDICE C

Grafici per l'analisi dei dati nei test delle tecniche di ricerca locale

Riportiamo di seguito alcuni grafici dei dati ottenuti dai test delle tecniche di ricerca locale, per due varianti di Rocket (Rocket_A e Rocket_B) e tre di Logistic (Logistic_A, Logistic_B e Logistic_C).

Per ogni tecnica di ricerca e per ogni dominio viene presentata una coppia di grafici, che riporta per un problema scelto (vedi elenco sottostante) i risultati ottenuti senza adattamento dei parametri e con adattamento dei parametri (utilizzando il metodo della varianza); si può vedere il miglioramento nelle prestanzioni e nell'indipendenza dai parametri N e/o L ottenuto con l'adattamento automatico dei parametri.

Walkplan: adattamento automatico del parametro N
RocketA: a3v3
RocketB: b2v3
LogisticA: b2v2
LogisticB: b2v2
LogisticC: a3v5

Tabuplan: adattamento automatico del parametro L
RocketA: a3v3
RocketB: a3v3
LogisticA: a3v7
LogisticB: a3v2
LogisticC: a3v5

T-walkplan: adattamento automatico dei parametri N e L
RocketA: a3v3
RocketB: b2v5
LogisticA: b2v2
LogisticB: a3v2
LogisticC: b2v5

Nota: nei grafici viene utilizzata per comodità una notazione diversa per i parametri α^i, β^i, γ^i, α^r, β^r, γ^r della funzione euristica, con le seguenti corrispondenze:

α^i -> x, β^i -> y, γ^i -> z;
α^r -> X, β^r -> Y, γ^r -> Z

APPENDICE D

Descrizione delle varianti per i domini di test

APPENDICE E

Specifiche degli ambienti del dominio reale

Riportiamo di seguito i seguenti file degli operatori:

- guard.ops che contiene gli operatori del dominio dei corridoi;
- room.ops che contiene gli operatori del dominio delle stanze.

```
# operatori con risorse per il dominio dei corridoi

# 1. operatori di movimento

# 1.1 operatori per muoversi nel corridoio

move
:v ?r robot ?s location ?a location
:p at(?r ?s) con(?s ?a) >($fuel() cost(?s ?a))
:e ADD at(?r ?a)
   DEL at(?r ?s)
   $fuel() -= cost(?s ?a).

refuel
:v ?r robot ?p refuel-point
:p at(?r ?p)
:e $fuel() :=600.

# 1.2 operatori per l'utilizzo dell'ascensore

call-lift
:v ?r robot ?p location ?l lift
:p at(?r ?p) in-front-of(?p ?l) closed(?l ?p) >($fuel() 4)
:e ADD open(?l ?p)
   DEL closed(?l ?p)
   $fuel() -=1.

take-lift
:v ?r robot ?s location ?a location ?l lift
:p at(?r ?s) in-front-of(?s ?l) in-front-of(?a ?l) open(?l ?s) con-
lift(?s ?a) >($fuel() 3)
:e ADD at(?r ?a) closed(?l ?s)
   DEL at(?r ?s) open(?l ?s)
   $fuel() -=2.

# 1.3 operatori per accedere alle stanze

open-room
:v ?r robot ?l location ?p room
:p at(?r ?l) in-front-of(?l ?p) closed(?p) sensor-off(?p) not-
locked(?p) >($fuel() 11)
:e ADD open(?p)
   DEL closed(?p)
   $fuel() -=1.
```

```
open-door
:v ?r robot ?p room ?l location
:p in(?r ?p) closed(?p ?l) >($fuel() 7)
:e ADD open(?p ?l);
   ALL ?s location in-front-of(?s ?p) !eq(?s ?l) => ADD closed(?s ?p)
   $fuel() -=3.

close-room
:v ?r robot ?l location ?p room
:p at(?r ?l) in-front-of(?l ?p) open(?p) >($fuel() 3)
:e ADD closed(?p)
   DEL open(?p)
   $fuel() -=1.

enter-room
:v ?r robot ?l location ?p room
:p at(?r ?l) in-front-of(?l ?p) open(?p) >($fuel() 10)
:e ADD in(?r ?p)
   DEL at(?r ?l)
   $fuel() -=2.

exit-room
:v ?r robot ?l location ?p room
:p in(?r ?p) in-front-of(?l ?p) open(?p) >($fuel() 5)
:e ADD at(?r ?l)
   DEL in(?r ?p)
   $fuel() -=2.

control-room
:v ?r robot ?p room
:p in(?r ?p) >($fuel() 8)
:e ADD controlled(?p)
   $fuel() -=3.


# 1.4 operatore per sbloccare le stanze

unlock-room
:v ?r robot ?l location ?p room
:p at(?r ?l) in-front-of(?l ?p) locked(?p) >($fuel() 12)
:e ADD not-locked(?p)
   DEL locked(?p)
   $fuel() -=1.

# 2. operatori per l'interazione con il centro di controllo

# 2.1 operatori per la disattivazione dei sensori

sensor-off
:v ?r robot ?l location ?p room
:p at(?r ?l) in-front-of(?l ?p) closed(?p) >($fuel() 13)
:e ADD sensor-off(?p)
   DEL sensor-on(?p)
   $fuel() -=1;
    ALL ?o room in-front-of(?l ?o) !eq(?o ?p) sensor-on(?o) => ADD
sensor-off(?o) DEL sensor-on(?o).
```

```
sensor-on
:v ?r robot ?l location ?p room
:p at(?r ?l) in-front-of(?l ?p) closed(?p) >($fuel() 2)
:e ADD sensor-on(?p)
   DEL sensor-off(?p)
   $fuel() -=1.

# 2.2 operatori per il controllo delle telecamere e dei sensori
antincendio

control-fire
:v ?r robot ?l location ?f fire
:p at(?r ?l) at(?f ?l) >($fuel() 0)
:e ADD controlled(?f)
   $fuel() -=1.

control-camera
:v ?r robot ?l location ?c camera
:p at(?r ?l) at(?c ?l) >($fuel() 0)
:e ADD controlled(?c)
   $fuel() -=1.


# 3. operatori per l'interazione con le persone

person-ok
:v ?r robot ?l location ?p person
:p at(?r ?l) at(?p ?l) pass(?p) >($fuel() 1)
:e ADD identified(?p)
   $fuel() -=2.

alarm-person
:v ?r robot ?l location ?p person ?a alarm
:p at(?r ?l) at(?p ?l) not-pass(?p) >($fuel() 1)
:e ADD identified(?p) alarm(?a)
   $fuel() -=2.
```

 # operatori con risorse per il dominio delle stanze

```
# 1. operatori di movimento

move
:v ?r robot ?s location ?a location
:p at(?r ?s) con(?s ?a) >($fuel() cost(?s ?a))
:e ADD at(?r ?a)
   DEL at(?r ?s)
   $fuel() -= cost(?s ?a).

refuel
:v ?r robot ?p refuel-point
:p at(?r ?p)
:e $fuel() :=600.

# 2. operatori per accendere e spegnere luci e telecamera
```

```
light-image-on
:v ?l light ?r robot
:p off(?l) >($fuel() 4)
:e ADD on(?l) image-on(?r)
   DEL off(?l) image-off(?r)
   $fuel() -=1.

send-image
:v ?l light ?r robot
:p on(?l) >($fuel() 2)
:e ADD image-on(?r)
   DEL image-off(?r)
   $fuel() -=3.

light-image-off
:v ?l light ?r robot
:p on(?l) image-on(?r) >($fuel() 0)
:e ADD off(?l) image-off(?r) controlled()
   DEL on(?l) image-on(?r)
   $fuel() -=1.


# 3. operatori per il controllo della presenza di oggetti

control-object
:v ?o object ?r robot ?l location
:p present(?o ?l) at(?r ?l) image-on(?r) identified(?o) >($fuel() 3)
:e ADD controlled(?o)
   $fuel() -=1.

alarm-object
:v ?o object ?r robot ?l location ?a alarm
:p not-present(?o ?l) at(?r ?l) image-on(?r) >($fuel() 1)
:e ADD controlled(?o) alarm-on(?a)
   $fuel() -=2.

identify-object
:v ?o object ?r robot ?l location
:p at(?o ?l) at(?r ?l) >($fuel() 4)
:e ADD identified(?o)
   $fuel() -=1.

# 4. operatori per il controllo dell'integrita' degli oggetti
preziosi

control-precious-object
:v ?p precious-object ?r robot ?l location
:p at(?r ?l) present-complete(?p ?l) image-on(?r) identified(?p)
>($fuel() 3)
:e ADD controlled(?p)
   $fuel() -=1.

alarm-precious-object
:v ?p precious-object ?r robot ?l location ?a alarm
:p at(?r ?l) not-present-complete(?p ?l) image-on(?r) >($fuel() 1)
:e ADD controlled(?p) alarm-on(?a)
   $fuel() -=2.
```

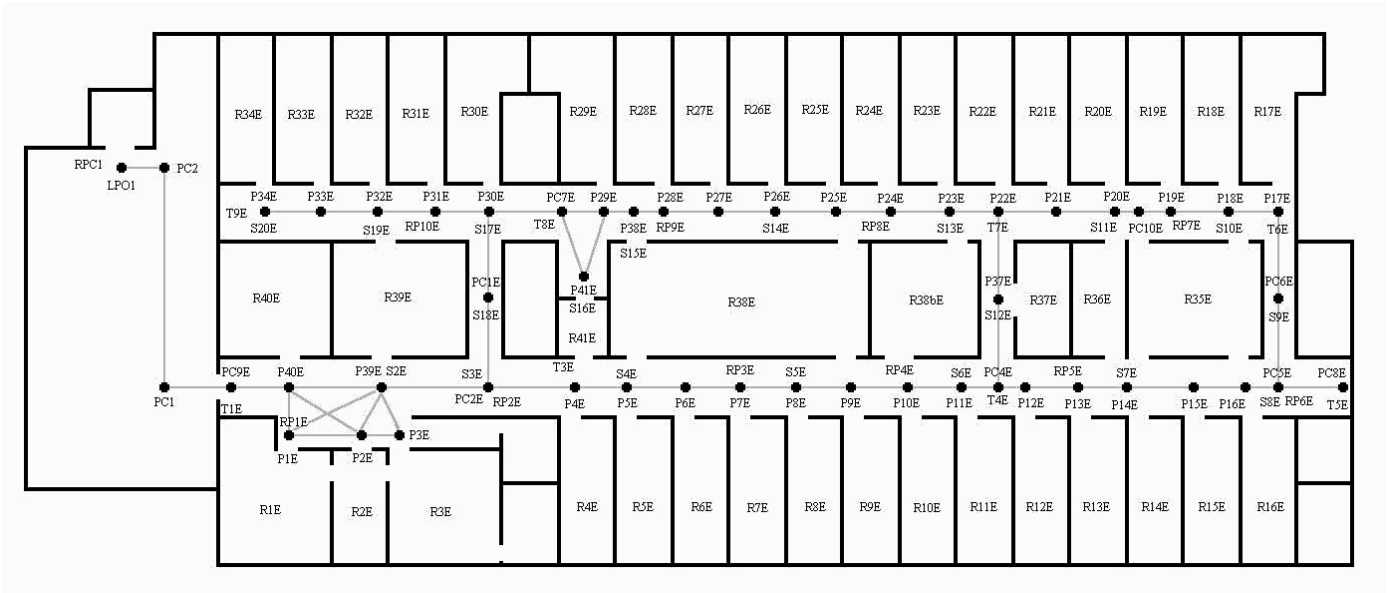
```
identify-precious-object
:v ?p precious-object ?r robot ?l location
:p at(?p ?l) at(?r ?l) >($fuel() 4)
:e ADD identified(?p)
   $fuel() -=1.

# 5. operatori per il controllo delle finestre

control-window
:v ?w window ?r robot ?l location
:p at(?w ?l) at(?r ?l) closed(?w) image-on(?r) identified(?w)
>($fuel() 3)
:e ADD controlled(?w)
   $fuel() -=1.

alarm-window
:v ?w window ?r robot ?l location ?a alarm
:p at(?w ?l) at(?r ?l) open(?w) image-on(?r) identified(?w) >($fuel()
1)
:e ADD controlled(?w) alarm-on(?a)
   $fuel() -=2.

identify-window
:v ?w window ?r robot ?l location
:p at(?w ?l) at(?r ?l) >($fuel() 4)
:e ADD identified(?w)
   $fuel() -=1.

# 6. operatori per il controllo della cassaforte

control-safe-box
:v ?s safe-box ?r robot ?l location
:p at(?s ?l) at(?r ?l) locked(?s) image-on(?r) identified(?s)
>($fuel() 3)
:e ADD controlled(?s)
   $fuel() -=1.

alarm-safe-box
:v ?s safe-box ?r robot ?l location ?a alarm
:p at(?s ?l) at(?r ?l) not-locked(?s) image-on(?r) identified(?s)
>($fuel() 1)
:e ADD controlled(?s) alarm-on(?a)
   $fuel() -=2.

identify-safe-box
:v ?s safe-box ?r robot ?l location
:p at(?s ?l) at(?r ?l) >($fuel() 4)
:e ADD identified(?s)
   $fuel() -=1.

# 7. operatori per il controllo dei sensori antincendio

control-fire
:v ?r robot ?l location ?f fire
:p at(?r ?l) at(?f ?l) >($fuel() 0)
:e ADD controlled(?f)
```

```
   $fuel() -=1.

# 8. operatori per l'interazione con le persone

person-ok
:v ?r robot ?l location ?p person
:p at(?r ?l) at(?p ?l) pass(?p) >($fuel() 1)
:e ADD identified(?p)
   $fuel() -=2.

alarm-person
:v ?r robot ?l location ?p person ?a alarm
:p at(?r ?l) at(?p ?l) not-pass(?p) >($fuel() 1)
:e ADD identified(?p) alarm(?a)
   $fuel() -=2.
```

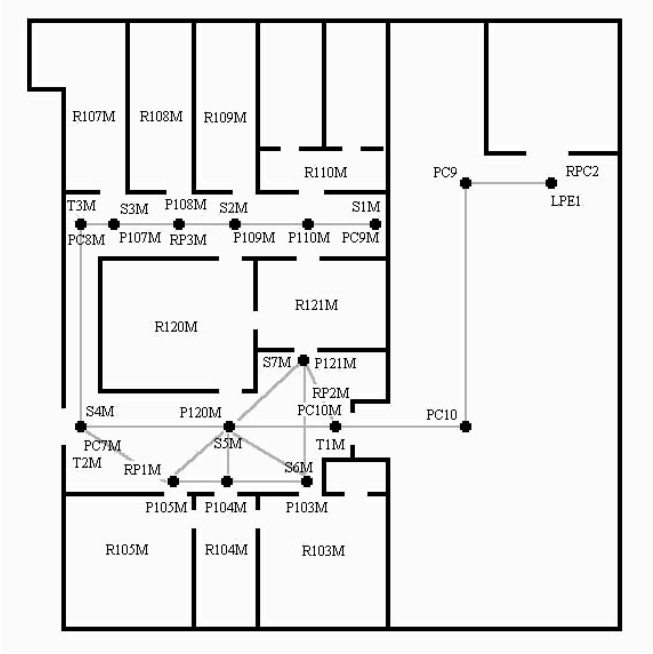
Riportiamo di seguito i file che descrivono gli ambienti dei corridoi e delle stanze:

- g-ing.fct, che rappresenta il dominio dei corridoi dei tre dipartimenti;
- r-el.fct, che rappresenta il laboratorio d'informatica del dipartimento di ingegneria elettronica;
- r-mec.fct, che rappresenta l'aula CAD del dipartimento di ingegneria meccanica;
- r-civ.fct, che rappresenta l'aula tesisti del dipartimento di ingegneria civile.

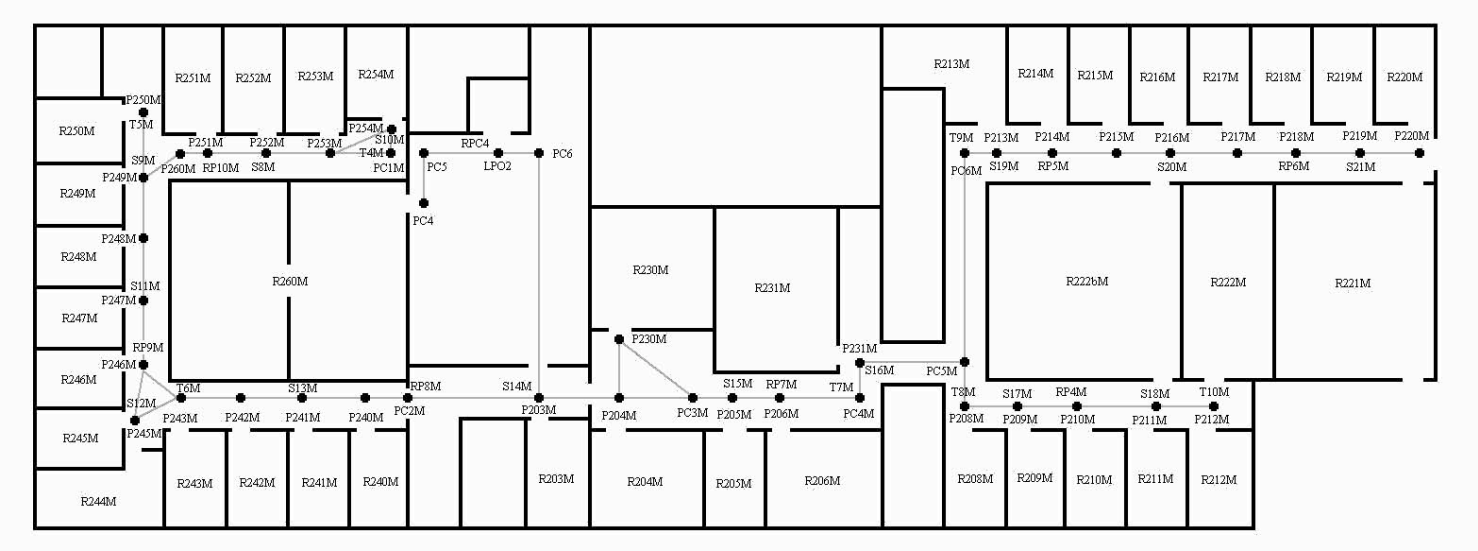
```
# università degli studi di Brescia - facolta' di ingegneria
# dipartimento di elettronica
```



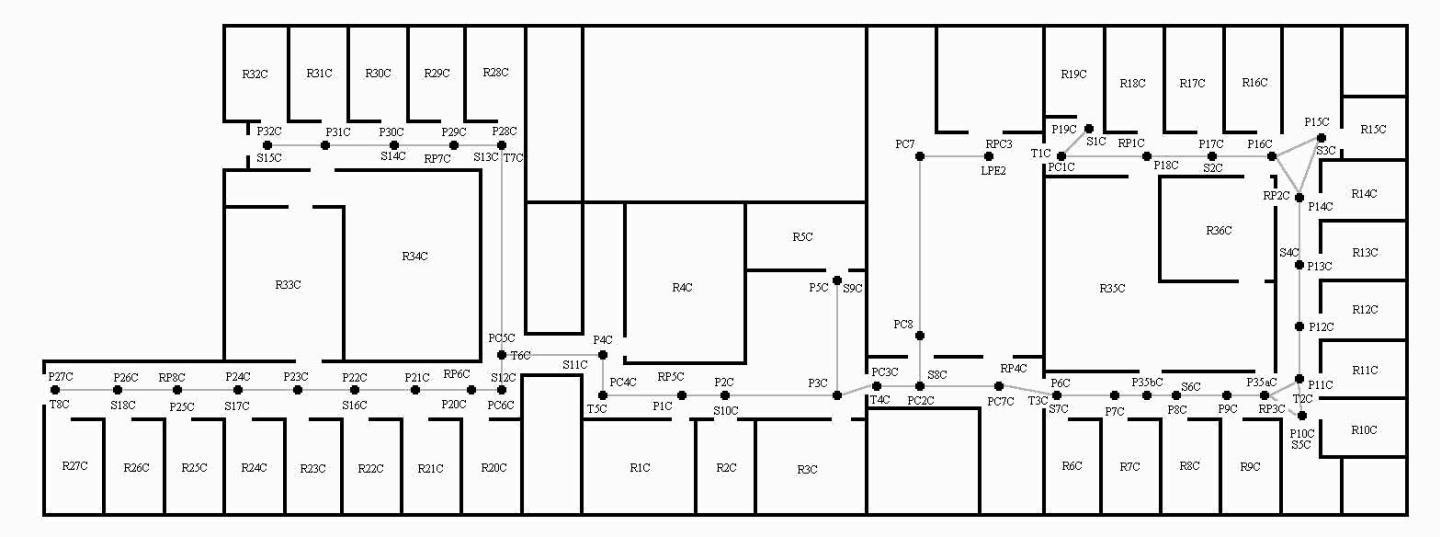
 # dipartimento di meccanica – 1° piano



 # dipartimento di meccanica – 2° piano



 # dipartimento di civile



 # g-ing.fct

```
location: P1E P2E P3E P4E P5E P6E P7E P8E P9E P10E P11E P12E P13E
          P14E P15E P16E P17E P18E P19E P20E P21E P22E P23E P24E P25E
          P26E P27E P28E P29E P30E P31E P32E P33E P34E P37E P38E P39E
          P40E P41E
          PC1E PC2E PC4E PC5E PC6E PC7E PC8E PC9E PC10E

          P103M P104M P105M P107M P108M P109M P110M P120M P121M
          PC7M PC8M PC9M PC10M
          P203M P204M P205M P206M P208M P209M P210M P211M P212M P213M
          P214M P215M P216M P217M P218M P219M P220M P230M P231M
          P240M P241M P242M P243M P245M P246M P247M P248M P249M P250M
          P260M P251M P252M P253M P254M
          PC1M PC2M PC3M PC4M PC5M PC6M

          P1C P2C P3C P4C P5C P6C P7C P8C P9C P10C P11C P12C P13C
          P14C P15C P16C P17C P18C P19C P20C
          P21C P22C P23C P24C P25C P26C P27C P28C P29C P30C P31C P32C
          P35aC P35bC
          PC1C PC2C PC3C PC4C PC5C PC6C PC7C

          PC1 PC2 PC4 PC5 PC6 PC7 PC8
          LPE1 LPE2 LPO1 LPO2;

room: R1E R2E R3E R4E R5E R6E R7E R8E R9E R10E R11E R12E R13E R14E
      R15E R16E R17E R18E R19E R20E R21E R22E R23E R24E R25E R26E
       R27E R28E R29E R30E R31E R32E R33E R34E R35E R36E R37E R38E
       R38bE R39E R40E R41E

      R103M R104M R105M R107M R108M R109M R110M R120M R121M
      R203M R204M R205M R206M R208M R209M R210M R211M R212M R213M
      R214M R215M R216M R217M R218M R219M R220M R221M R222M R222bM
      R230M R231M R240M R241M R242M R243M R244M R245M R246M R247M
      R248M R249M R250M R251M R252M R253M R254M R260M
```

```
      R1C R2C R3C R4C R5C R6C R7C R8C R9C R10C R11C R12C R13C R14C
      R15C R16C R17C R18C R19C R20C
      R21C R22C R23C R24C R25C R26C R27C R28C R29C R30C R31C R32C
      R33C R34C R35C R36C;

robot: RG;

lift: LE LO;

fire: S1E S2E S3E S4E S5E S6E S7E S8E S9E S10E S11E S12E S13E S14E
      S15E S16E S17E S18E S19E S20E
      S1M S2M S3M S4M S5M S6M S7M S8M S9M S10M S11M S12M S13M S14M
      S15M S16M S17M S18M S19M S20M S21M
      S1C S2C S3C S4C S5C S6C S7C S8C S9C S10C S11C S12C S13C S14C
      S15C S16C S17C S18C;

camera: T1E T2E T3E T4E T5E T6E T7E T8E T9E
        T1M T2M T3M T4M T5M T6M T7M T8M T9M T10M
        T1C T2C T3C T4C T5C T6C T7C T8C;

refuel-point: RP1E RP2E RP3E RP4E RP5E RP6E RP7E RP8E RP9E RP10E
              RP1M RP2M RP3M RP4M RP5M RP6M RP7M RP8M RP9M RP10M
              RP1C RP2C RP3C RP4C RP5C RP6C RP7C RP8C
              RPC1 RPC2 RPC3 RPC4;

$fuel(): 0 600;

database:

# meccanica piano primo - ascensore est piano primo
cost(PC10M PC10) 4 cost(PC10 PC10M) 4
cost(PC10 PC9) 6 cost(PC9 PC10) 6
cost(PC9 LPE1) 3 cost(LPE1 PC9) 3

# elettronica - ascensore ovest piano primo
cost(PC9E PC1) 3 cost(PC1 PC9E) 3 cost(PC1 PC2) 6 cost(PC2 PC1) 6
cost(PC2 LPO1) 2 cost(LPO1 PC2) 2

# civile - ascensore est piano secondo
cost(PC7 PC8) 6 cost(PC8 PC7) 6 cost(PC8 PC2C) 3 cost(PC2C PC8) 3
cost(PC7 LPE2) 3 cost(LPE2 PC7) 3

# meccanica piano secondo - ascensore ovest piano secondo
cost(LPO2 PC5) 4 cost(PC5 LPO2) 4 cost(PC5 PC4) 3 cost(PC4 PC5) 3
cost(LPO2 PC6) 2 cost(PC6 LPO2) 2 cost(PC6 P203M) 7 cost(P203M PC6) 7

#################### DIPARTIMENTO DI ELETTRONICA ####################

# connessioni dell'atrio
cost(PC9E P40E) 3 cost(P40E PC9E) 3
cost(P40E P39E) 3 cost(P39E P40E) 3
cost(P40E P1E) 2 cost(P1E P40E) 2 cost(P1E P2E) 3 cost(P2E P1E) 3
cost(P2E P3E) 2 cost(P3E P2E) 2 cost(P39E PC2E) 3 cost(PC2E P39E) 3
cost(P1E P39E) 4 cost(P39E P1E) 4 cost(P40E P2E) 4 cost(P2E P40E) 4
cost(P39E P2E) 3 cost(P2E P39E) 3 cost(P39E P3E) 4 cost(P3E P39E) 4

# connessioni del corridoio sud
```

```
cost(PC2E P4E) 4 cost(P4E PC2E) 4 cost(P4E P5E) 4 cost(P5E P4E) 4
cost(P5E P6E) 4 cost(P6E P5E) 4 cost(P6E P7E) 4 cost(P7E P6E) 4
cost(P7E P8E) 4 cost(P8E P7E) 4 cost(P8E P9E) 4 cost(P9E P8E) 4
cost(P9E P10E) 4 cost(P10E P9E) 4 cost(P10E P11E) 4 cost(P11E P10E) 4
cost(P11E PC4E) 3 cost(PC4E P11E) 3
cost(PC4E P12E) 3 cost(P12E PC4E) 3
cost(P12E P13E) 4 cost(P13E P12E) 4
cost(P13E P14E) 4 cost(P14E P13E) 4
cost(P14E P15E) 4 cost(P15E P14E) 4
cost(P15E P16E) 4 cost(P16E P15E) 4
cost(P16E PC5E) 3 cost(PC5E P16E) 3
cost(PC5E PC8E) 5 cost(PC8E PC5E) 5

# connessioni della traversa ovest
cost(PC2E PC1E) 5 cost(PC1E PC2E) 5 cost(PC1E P30E) 5 cost(P30E PC1E)
5

# connessioni della traversa centrale
cost(PC4E P37E) 5 cost(P37E PC4E) 5 cost(P37E P22E) 5 cost(P22E P37E)
5

# connessioni della traversa est
cost(PC5E PC6E) 5 cost(PC6E PC5E) 5 cost(PC6E P17E) 5 cost(P17E PC6E)
5

# connessioni del corridoio nord
cost(P34E P33E) 4 cost(P33E P34E) 4
cost(P33E P32E) 4 cost(P32E P33E) 4
cost(P32E P31E) 4 cost(P31E P32E) 4
cost(P31E P30E) 4 cost(P30E P31E) 4
cost(P30E PC7E) 4 cost(PC7E P30E) 4
cost(PC7E P41E) 4 cost(P41E PC7E) 4
cost(PC7E P29E) 4 cost(P29E PC7E) 4
cost(P41E P29E) 4 cost(P29E P41E) 4
cost(P29E P38E) 3 cost(P38E P29E) 3
cost(P29E P39E) 3 cost(P39E P29E) 3
cost(P28E P27E) 4 cost(P27E P28E) 4
cost(P27E P26E) 4 cost(P26E P27E) 4
cost(P26E P25E) 4 cost(P25E P26E) 4
cost(P25E P24E) 4 cost(P24E P25E) 4
cost(P24E P23E) 4 cost(P23E P24E) 4
cost(P23E P22E) 4 cost(P22E P23E) 4
cost(P22E P21E) 4 cost(P21E P22E) 4
cost(P21E P20E) 4 cost(P20E P21E) 4
cost(P20E PC10E) 2 cost(PC10E P20E) 2
cost(PC10E P19E) 3 cost(P19E PC10E) 3
cost(P19E P18E) 4 cost(P18E P19E) 4
cost(P18E P17E) 4 cost(P17E P18E) 4

############## DIPARTIMENTO DI MECCANICA - PIANO PRIMO ##############

# connessioni
cost(PC7M P105M) 4 cost(P105M PC7M) 4
cost(PC7M P120M) 5 cost(P120M PC7M) 5
cost(P120M P121M) 4 cost(P121M P120M) 4
cost(P105M P104M) 2 cost(P104M P105M) 2
cost(P105M P120M) 3 cost(P120M P105M) 3
```

```
cost(P120M P104M) 3 cost(P104M P120M) 3
cost(P104M P103M) 3 cost(P103M P104M) 3
cost(P120M P103M) 4 cost(P103M P120M) 4
cost(P103M P121M) 5 cost(P121M P103M) 5
cost(PC10M P120M) 4 cost(P120M PC10M) 4
cost(PC10M P121M) 3 cost(P121M PC10M) 3
cost(PC7M PC8M) 6 cost(PC8M PC7M) 6
cost(P107M PC8M) 2 cost(PC8M P107M) 2
cost(P107M P108M) 3 cost(P108M P107M) 3
cost(P108M P109M) 3 cost(P109M P108M) 3
cost(P109M P110M) 3 cost(P110M P109M) 3
cost(P110M PC9M) 3 cost(PC9M P110M) 3

###################### DIPARTIMENTO DI CIVILE ######################

# connessioni del lato est
cost(P19C PC1C) 3 cost(PC1C P19C) 3
cost(PC1C P18C) 4 cost(P18C PC1C) 4
cost(P18C P17C) 3 cost(P17C P18C) 3
cost(P17C P16C) 3 cost(P16C P17C) 3
cost(P16C P15C) 3 cost(P15C P16C) 3
cost(P16C P14C) 3 cost(P14C P16C) 3
cost(P15C P14C) 4 cost(P14C P15C) 4
cost(P14C P13C) 3 cost(P13C P14C) 3
cost(P13C P12C) 4 cost(P12C P13C) 4
cost(P12C P11C) 3 cost(P11C P12C) 3
cost(P11C P10C) 3 cost(P10C P11C) 3
cost(P11C P35aC) 3 cost(P35aC P11C) 3
cost(P35aC P10C) 3 cost(P10C P35aC) 3
cost(P35aC P9C) 3 cost(P9C P35aC) 3
cost(P9C P8C) 3 cost(P8C P9C) 3 cost(P8C P35bC) 2 cost(P35bC P8C) 2
cost(P35bC P7C) 2 cost(P7C P35bC) 2 cost(P7C P6C) 3 cost(P6C P7C) 3

# connessioni dell'atrio
cost(P6C PC7C) 3 cost(PC7C P6C) 3 cost(PC7C PC2C) 4 cost(PC2C PC7C) 4
cost(PC2C PC3C) 3 cost(PC3C PC2C) 3 cost(PC3C P3C) 3 cost(P3C PC3C) 3
cost(P3C P5C) 6 cost(P5C P3C) 6 cost(P3C P2C) 5 cost(P2C P3C) 5
cost(P2C P1C) 3 cost(P1C P2C) 3 cost(P1C PC4C) 4 cost(PC4C P1C) 4
cost(PC4C P4C) 3 cost(P4C PC4C) 3 cost(P4C PC5C) 5 cost(PC5C P4C) 5

# connessioni del lato ovest
cost(PC5C PC6C) 3 cost(PC6C PC5C) 3
cost(PC6C P20C) 3 cost(P20C PC6C) 3
cost(P20C P21C) 3 cost(P21C P20C) 3
cost(P21C P22C) 3 cost(P22C P21C) 3
cost(P22C P23C) 3 cost(P23C P22C) 3
cost(P23C P24C) 3 cost(P24C P23C) 3
cost(P24C P25C) 3 cost(P25C P24C) 3
cost(P25C P26C) 3 cost(P26C P25C) 3
cost(P26C P27C) 3 cost(P27C P26C) 3
cost(PC5C P28C) 10 cost(P28C PC5C) 10
cost(P28C P29C) 3 cost(P29C P28C) 3
cost(P29C P30C) 3 cost(P30C P29C) 3
cost(P30C P31C) 3 cost(P31C P30C) 3
cost(P31C P32C) 3 cost(P32C P31C) 3

############# DIPARTIMENTO DI MECCANICA - PIANO SECONDO #############
```

```
# connessioni dell'atrio
cost(P240M PC2M) 3 cost(PC2M P240M) 3
cost(PC2M P203M) 6 cost(P203M PC2M) 6
cost(P203M P204M) 5 cost(P204M) 5
cost(P204M P230M) 5 cost(P230M P204M) 5
cost(P204M PC3M) 4 cost(PC3M P204M) 4
cost(P230M PC3M) 6 cost(PC3M P230M) 6
cost(PC3M P205M) 3 cost(P205M PC3M) 3
cost(P205M P206M) 3 cost(P206M P205M) 3
cost(P206M PC4M) 4 cost(PC4M P206M) 4
cost(PC4M P231M) 3 cost(P231M PC4M) 3
cost(P231M PC5M) 5 cost(PC5M P231M) 5

# connessioni della parte est
cost(PC5M P208M) 3 cost(P208M PC5M) 3
cost(P208M P209M) 3 cost(P209M P208M) 3
cost(P209M P210M) 3 cost(P210M P209M) 3
cost(P210M P211M) 4 cost(P211M P210M) 4
cost(P211M P212M) 3 cost(P212M P211M) 3
cost(PC5M PC6M) 10 cost(PC6M PC5M) 10
cost(PC6M P213M) 2 cost(P213M PC6M) 2
cost(P213M P214M) 3 cost(P214M P213M) 3
cost(P214M P215M) 3 cost(P215M P214M) 3
cost(P215M P216M) 3 cost(P216M P215M) 3
cost(P216M P217M) 3 cost(P217M P216M) 3
cost(P217M P218M) 3 cost(P218M P217M) 3
cost(P218M P219M) 3 cost(P219M P218M) 3
cost(P219M P220M) 3 cost(P220M P219M) 3

# connessioni della parte ovest
cost(P254M PC1M) 2 cost(PC1M P254M) 2
cost(P254M P253M) 4 cost(P253M P254M) 4
cost(PC1M P253M) 4 cost(P253M PC1M) 4
cost(P253M P252M) 4 cost(P252M P253M) 4
cost(P252M P251M) 3 cost(P251M P252M) 3
cost(P251M P260M) 2 cost(P260M P251M) 2
cost(P260M P249M) 4 cost(P249M P260M) 4
cost(P249M P250M) 5 cost(P250M P249M) 5
cost(P249M P248M) 4 cost(P248M P249M) 4
cost(P248M P247M) 4 cost(P247M P248M) 4
cost(P247M P246M) 4 cost(P246M P247M) 4
cost(P246M P245M) 4 cost(P245M P246M) 4
cost(P246M P243M) 4 cost(P243M P246M) 4
cost(P245M P243M) 4 cost(P243M P245M) 4
cost(P243M P242M) 3 cost(P242M P243M) 3
cost(P242M P241M) 3 cost(P241M P242M) 3
cost(P241M P240M) 3 cost(P240M P241M) 3;

initial:

=($fuel() 600)

# posizione iniziale del robot guardiano
at(RG PE1) has-fuel(RG)

# connessioni esterne
```

```
# meccanica piano primo - ascensore est piano primo
con(PC10M PC10) con(PC10 PC10M) con(PC10 PC9) con(PC9 PC10)
con(PC9 LPE1) con(LPE1 PC9)

# elettronica - ascensore ovest piano primo
con(PC9E PC1) con(PC1 PC9E) con(PC1 PC2) con(PC2 PC1) con(PC2 LPO1)
con(LPO1 PC2)

# civile - ascensore est piano secondo
con(PC7 PC8) con(PC8 PC7) con(PC8 PC2C) con(PC2C PC8) con(PC7 LPE2)
con(LPE2 PC7)

# meccanica piano secondo - ascensore ovest piano secondo
con(LPO2 PC5) con(PC5 LPO2) con(PC5 PC4) con(PC4 PC5) con(LPO2 PC6)
con(PC6 LPO2) con(PC6 P203M) con(P203M PC6)

# connessioni degli ascensori
con-lift(LPO1 LPO2) con-lift(LPO2 LPO1)
con-lift(LPE1 LPE2) con-lift(LPE2 LPE1)
in-front-of(LPO1 LPO) closed(LPO LPO1)
in-front-of(LPO2 LPO) closed(LPO LPO2)
in-front-of(LPE1 LPE) closed(LPE LPE1)
in-front-of(LPE2 LPE) closed(LPE LPE2)


# punti di ricarica
at(RPC1 LPO1) at(RPC2 LPE1) at(RPC4 LPO2) at(RPC3 LPE2)

#################### DIPARTIMENTO DI ELETTRONICA ####################

# connessioni dell'atrio
con(PC9E P40E) con(P40E PC9E) con(P40E P39E) con(P39E P40E)
con(P40E P1E) con(P1E P40E) con(P1E P2E) con(P2E P1E)
con(P2E P3E) con(P3E P2E) con(P39E PC2E) con(PC2E P39E)
con(P1E P39E) con(P39E P1E) con(P40E P2E) con(P2E P40E)
con(P39E P2E) con(P2E P39E) con(P39E P3E) con(P3E P39E)

# connessioni del corridoio sud
con(PC2E P4E) con(P4E PC2E) con(P4E P5E) con(P5E P4E)
con(P5E P6E) con(P6E P5E)  con(P6E P7E) con(P7E P6E)
con(P7E P8E) con(P8E P7E) con(P8E P9E) con(P9E P8E)
con(P9E P10E) con(P10E P9E) con(P10E P11E) con(P11E P10E)
con(P11E PC4E) con(PC4E P11E) con(PC4E P12E) con(P12E PC4E)
con(P12E P13E) con(P13E P12E) con(P13E P14E) con(P14E P13E)
con(P14E P15E) con(P15E P14E) con(P15E P16E) con(P16E P15E)
con(P16E PC5E) con(PC5E P16E) con(PC5E PC8E) con(PC8E PC5E)

# connessioni della traversa ovest
con(PC2E PC1E) con(PC1E PC2E) con(PC1E P30E) con(P30E PC1E)

# connessioni della traversa centrale
con(PC4E P37E) con(P37E PC4E) con(P37E P22E) con(P22E P37E)

# connessioni della traversa est
con(PC5E PC6E) con(PC6E PC5E) con(PC6E P17E) con(P17E PC6E)
```

```
# connessioni del corridoio nord
con(P34E P33E) con(P33E P34E) con(P33E P32E) con(P32E P33E)
con(P32E P31E) con(P31E P32E) con(P31E P30E) con(P30E P31E)
con(P30E PC7E) con(PC7E P30E) con(PC7E P41E) con(P41E PC7E)
con(PC7E P29E) con(P29E PC7E) con(P41E P29E) con(P29E P41E)
con(P29E P38E) con(P38E P29E) con(P29E P28E) con(P28E P29E)
con(P28E P27E) con(P27E P28E) con(P27E P26E) con(P26E P27E)
con(P26E P25E) con(P25E P26E) con(P25E P24E) con(P24E P25E)
con(P24E P23E) con(P23E P24E) con(P23E P22E) con(P22E P23E)
con(P22E P21E) con(P21E P22E) con(P21E P20E) con(P20E P21E)
con(P20E PC10E) con(PC10E P20E) con(PC10E P19E) con(P19E PC10E)
con(P19E P18E) con(P18E P19E) con(P18E P17E) con(P17E P18E)

# posizione delle stanze e stato dei relativi sensori e porte
in-front-of(P1E R1E) sensor-on(R1E) closed(R1E P1E) not-locked(R1E)
in-front-of(P2E R2E) sensor-on(R2E) closed(R2E P2E) not-locked(R2E)
in-front-of(P3E R3E) sensor-on(R3E) closed(R3E P3E) not-locked(R3E)
in-front-of(P4E R4E) sensor-on(R4E) closed(R4E P4E) not-locked(R4E)
in-front-of(P5E R5E) sensor-on(R5E) closed(R5E P5E) not-locked(R5E)
in-front-of(P6E R6E) sensor-on(R6E) closed(R6E P6E) not-locked(R6E)
in-front-of(P7E R7E) sensor-on(R7E) closed(R7E P7E) not-locked(R7E)
in-front-of(P8E R8E) sensor-on(R8E) closed(R8E P8E) not-locked(R8E)
in-front-of(P9E R9E) sensor-on(R9E) closed(R9E P9E) not-locked(R9E)
in-front-of(P10E R10E) sensor-on(R10E)
closed(R10E P10E) not-locked(R10E)
in-front-of(P11E R11E) sensor-on(R11E)
closed(R11E P11E) not-locked(R11E)
in-front-of(P12E R12E) sensor-on(R12E)
closed(R12E P12E) not-locked(R12E)
in-front-of(P13E R13E) sensor-on(R13E)
closed(R13E P13E) not-locked(R13E)
in-front-of(P14E R14E) sensor-on(R14E)
closed(R14E P14E) not-locked(R14E)
in-front-of(P15E R15E) sensor-on(R15E)
closed(R15E P15E) not-locked(R15E)
in-front-of(P16E R16E) sensor-on(R16E)
closed(R16E P16E) not-locked(R16E)
in-front-of(P17E R17E) sensor-on(R17E)
closed(R17E P17E) not-locked(R17E)
in-front-of(P18E R18E) sensor-on(R18E)
closed(R18E P18E) not-locked(R18E)
in-front-of(P19E R19E) sensor-on(R19E)
closed(R19E P19E) not-locked(R19E)
in-front-of(P20E R20E) sensor-on(R20E)
closed(R20E P20E) not-locked(R20E)
in-front-of(P21E R21E) sensor-on(R21E)
closed(R21E P21E) not-locked(R21E)
in-front-of(P22E R22E) sensor-on(R22E)
closed(R22E P22E) not-locked(R22E)
in-front-of(P23E R23E) sensor-on(R23E)
closed(R23E P23E) not-locked(R23E)
in-front-of(P24E R24E) sensor-on(R24E)
closed(R24E P24E) not-locked(R24E)
in-front-of(P25E R25E) sensor-on(R25E)
closed(R25E P25E) not-locked(R25E)
in-front-of(P26E R26E) sensor-on(R26E)
closed(R26E P26E) not-locked(R26E)
```

```
in-front-of(P27E R27E) sensor-on(R27E)
closed(R27E P27E) not-locked(R27E)
in-front-of(P28E R28E) sensor-on(R28E)
closed(R28E P28E) not-locked(R28E)
in-front-of(P29E R29E) sensor-on(R29E)
closed(R29E P29E) not-locked(R29E)
in-front-of(P30E R30E) sensor-on(R30E)
closed(R30E P30E) not-locked(R30E)
in-front-of(P31E R31E) sensor-on(R31E)
closed(R31E P31E) not-locked(R31E)
in-front-of(P32E R32E) sensor-on(R32E)
closed(R32E P32E) not-locked(R32E)
in-front-of(P33E R33E) sensor-on(R33E)
closed(R33E P33E) not-locked(R33E)
in-front-of(P34E R34E) sensor-on(R34E)
closed(R34E P34E) not-locked(R34E)
in-front-of(P16E R35E) in-front-of(PC10E R35E)
in-front-of(P18E R35E) in-front-of(P14E R35E) sensor-on(R35E)
closed(R35E P16E) closed(R35E PC10E) closed(R35E P18E)
closed(R35E P14E) not-locked(R35E)
in-front-of(P14E R36E) in-front-of(P20E R36E) sensor-on(R36E)
closed(R36E P14E) closed(R36E P20E) not-locked(R36E)
in-front-of(P37E R37E) sensor-on(R37E) closed(R37E P37E)
not-locked(R37E)
in-front-of(P9E R38E) in-front-of(P5E R38E) in-front-of(P38E R38E)
in-front-of(P25E R38E) sensor-on(R38E)
closed(R38E P9E) closed(R38E P5E) closed(R38E P38E) closed(R38E P25E)
not-locked(R38E)
in-front-of(P10E R38bE) in-front-of(P23E R38bE) sensor-on(R38bE)
closed(R38bE P10E) closed(R38bE P23E) not-locked(R38bE)
in-front-of(P39E R39E) in-front-of(P32E R39E) sensor-on(R39E)
closed(R39E P39E) closed(R39E P32E) not-locked(R39E)
in-front-of(P40E R40E) sensor-on(R40E) closed(R40E P40E)
not-locked(R40E)
in-front-of(P41E R41E) in-front-of(P4E R41E) sensor-on(R41E)
closed(R41E P41E) closed(R41E P4E) not-locked(R41E)

# telecamere
at(T1E PC9E) at(T2E P30E) at(T3E P4E) at(T4E PC4E) at(T5E PC8E)
at(T6E P17E) at(T7E P22E) at(T8E PC7E) at(T9E P34E)

# sensori antincendio
at(S1E PC9E) at(S2E P39E) at(S3E PC2E) at(S4E P5E) at(S5E P8E)
at(S6E P11E) at(S7E P36E) at(S8E PC5E) at(S9E PC6E) at(S10E P17E)
at(S11E P20E) at(S12E P37E) at(S13E P23E) at(S14E P26E) at(S15E P38E)
at(S16E P41E) at(S17E P30E) at(S18E PC1E) at(S19E P32E) at(S20E P34E)

# punti di ricarica
at(RP1E P1E) at(RP2E PC2E) at(RP3E P7E) at(RP4E P10E) at(RP5E P13E)
at(RP6E PC5E) at(RP7E P19E) at(RP8E P24E) at(RP9E P28E) at(RP10E
P31E);


############## DIPARTIMENTO DI MECCANICA - PIANO PRIMO ##############

# connessioni
con(PC7M P105M) con(P105M PC7M) con(PC7M P120M) con(P120M PC7M)
```

```
con(P120M P121M) con(P121M P120M) con(P105M P104M) con(P104M P105M)
con(P105M P120M) con(P120M P105M) con(P120M P104M) con(P104M P120M)
con(P104M P103M) con(P103M P104M) con(P120M P103M) con(P103M P120M)
con(P103M P121M) con(P121M P103M) con(PC10M P120M) con(P120M PC10M)
con(PC10M P121M) con(P121M PC10M) con(PC7M PC8M) con(PC8M PC7M)
con(P107M PC8M) con(PC8M P107M) con(P107M P108M) con(P108M P107M)
con(P108M P109M) con(P109M P108M) con(P109M P110M) con(P110M P109M)
con(P110M PC9M) con(PC9M P110M)

# posizione delle stanze e stato dei relativi sensori e porte
in-front-of(P103M R103M) sensor-on(R103M) closed(R103M P103M)
not-locked(R103M)
in-front-of(P104M R104M) sensor-on(R104M) closed(R104M P104M)
not-locked(R104M)
in-front-of(P105M R105M) sensor-on(R105M) closed(R105M P105M)
not-locked(R105M)
in-front-of(P107M R107M) sensor-on(R107M) closed(R107M P107M)
not-locked(R107M)
in-front-of(P108M R108M) sensor-on(R108M) closed(R108M P108M)
not-locked(R108M)
in-front-of(P109M R109M) sensor-on(R109M) closed(R109M P109M)
not-locked(R109M)
in-front-of(P110M R110M) sensor-on(R110M) closed(R110M P110M)
not-locked(R110M)
in-front-of(P120M R120M) in-front-of(P109M R120M) sensor-on(R120M)
closed(R120M P120M) closed(R120M P109M) not-locked(R120M)
in-front-of(P121M R121M) in-front-of(P110M R121M) sensor-on(R121M)
closed(R121M P121M) closed(R121M P110M) not-locked(R121M)

# telecamere
at(T1M PC10M) at(T2M PC7M) at(T3M PC8M)

# sensori antincendio
at(S1M PC9M) at(S2M P109M) at(S3M P107M) at(S4M PC7M) at(S5M P120M)
at(S6M P103M) at(S7M P121M)

# punti di ricarica
at(RP1M P105M) at(RP2M PC10M) at(RP3M P108M)

######################## DIPARTIMENTO DI CIVILE ######################

# connessioni del lato est
con(P19C PC1C) con(PC1C P19C) con(PC1C P18C) con(P18C PC1C)
con(P18C P17C) con(P17C P18C) con(P17C P16C) con(P16C P17C)
con(P16C P15C) con(P15C P16C) con(P16C P14C) con(P14C P16C)
con(P15C P14C) con(P14C P15C) con(P14C P13C) con(P13C P14C)
con(P13C P12C) con(P12C P13C) con(P12C P11C) con(P11C P12C)
con(P11C P10C) con(P10C P11C) con(P11C P35aC) con(P35aC P11C)
con(P35aC P10C) con(P10C P35aC) con(P35aC P9C) con(P9C P35aC)
con(P9C P8C) con(P8C P9C) con(P8C P35bC) con(P35bC P8C)
con(P35bC P7C) con(P7C P35bC) con(P7C P6C) con(P6C P7C)

# connessioni dell'atrio
con(P6C PC7C) con(PC7C P6C) con(PC7C PC2C) con(PC2C PC7C)
con(PC2C PC3C) con(PC3C PC2C) con(PC3C P3C) con(P3C PC3C)
con(P3C P5C) con(P5C P3C) con(P3C P2C) con(P2C P3C)
con(P2C P1C) con(P1C P2C) con(P1C PC4C) con(PC4C P1C)
```

```
con(PC4C P4C) con(P4C PC4C) con(P4C PC5C) con(PC5C P4C)

# connessioni del lato ovest
con(PC5C PC6C) con(PC6C PC5C) con(PC6C P20C) con(P20C PC6C)
con(P20C P21C) con(P21C P20C) con(P21C P22C) con(P22C P21C)
con(P22C P23C) con(P23C P22C) con(P23C P24C) con(P24C P23C)
con(P24C P25C) con(P25C P24C) con(P25C P26C) con(P26C P25C)
con(P26C P27C) con(P27C P26C) con(PC5C P28C) con(P28C PC5C)
con(P28C P29C) con(P29C P28C) con(P29C P30C) con(P30C P29C)
con(P30C P31C) con(P31C P30C) con(P31C P32C) con(P32C P31C)

# posizione delle stanze e stato dei relativi sensori e porte
in-front-of(P1C R1C) sensor-on(R1C) closed(R1C P1C) not-locked(R1C)
in-front-of(P2C R2C) sensor-on(R2C) closed(R2C P2C) not-locked(R2C)
in-front-of(P3C R3C) sensor-on(R3C) closed(R3C P3C) not-locked(R3C)
in-front-of(P4C R4C) sensor-on(R4C) closed(R4C P4C) not-locked(R4C)
in-front-of(P5C R5C) sensor-on(R5C) closed(R5C P5C) not-locked(R5C)
in-front-of(P6C R6C) sensor-on(R6C) closed(R6C P6C) not-locked(R6C)
in-front-of(P7C R7C) sensor-on(R7C) closed(R7C P7C) not-locked(R7C)
in-front-of(P8C R8C) sensor-on(R8C) closed(R8C P8C) not-locked(R8C)
in-front-of(P9C R9C) sensor-on(R9C) closed(R9C P9C) not-locked(R9C)
in-front-of(P10C R10C) sensor-on(R10C) closed(R10C P10C)
not-locked(R10C)
in-front-of(P11C R11C) sensor-on(R11C) closed(R11C P11C)
not-locked(R11C)
in-front-of(P12C R12C) sensor-on(R12C) closed(R12C P12C)
not-locked(R12C)
in-front-of(P13C R13C) sensor-on(R13C) closed(R13C P13C)
not-locked(R13C)
in-front-of(P14C R14C) sensor-on(R14C) closed(R14C P14C)
not-locked(R14C)
in-front-of(P15C R15C) sensor-on(R15C) closed(R15C P15C)
not-locked(R15C)
in-front-of(P16C R16C) sensor-on(R16C) closed(R16C P16C)
not-locked(R16C)
in-front-of(P17C R17C) sensor-on(R17C) closed(R17C P17C)
not-locked(R17C)
in-front-of(P18C R18C) sensor-on(R18C) closed(R18C P18C)
not-locked(R18C)
in-front-of(P19C R19C) sensor-on(R19C) closed(R19C P19C)
not-locked(R19C)
in-front-of(P20C R20C) sensor-on(R20C) closed(R20C P20C)
not-locked(R20C)
in-front-of(P21C R21C) sensor-on(R21C) closed(R21C P21C)
not-locked(R21C)
in-front-of(P22C R22C) sensor-on(R22C) closed(R22C P22C)
not-locked(R22C)
in-front-of(P23C R23C) sensor-on(R23C) closed(R23C P23C)
not-locked(R23C)
in-front-of(P24C R24C) sensor-on(R24C) closed(R24C P24C)
not-locked(R24C)
in-front-of(P25C R25C) sensor-on(R25C) closed(R25C P25C)
not-locked(R25C)
in-front-of(P26C R26C) sensor-on(R26C) closed(R26C P26C)
not-locked(R26C)
in-front-of(P27C R27C) sensor-on(R27C) closed(R27C P27C)
not-locked(R27C)
```

```
in-front-of(P28C R28C) sensor-on(R28C) closed(R28C P28C)
not-locked(R28C)
in-front-of(P29C R29C) sensor-on(R29C) closed(R29C P29C)
not-locked(R29C)
in-front-of(P30C R30C) sensor-on(R30C) closed(R30C P30C)
not-locked(R30C)
in-front-of(P31C R31C) sensor-on(R31C) closed(R31C P31C)
not-locked(R31C)
in-front-of(P32C R32C) sensor-on(R32C) closed(R32C P32C)
not-locked(R32C)
in-front-of(P23C R33C) sensor-on(R33C) closed(R33C P23C)
not-locked(R33C)
in-front-of(P31C R34C) sensor-on(R34C) closed(R34C P31C)
not-locked(R34C)
in-front-of(P35aC R35C) in-front-of(P35bC R35C)
in-front-of(P18C R35C) sensor-on(R35C)
closed(R35C P35aC) closed(R35C P35bC) closed(R35C P18C)
not-locked(R35C)
in-front-of(P16C R36C) sensor-on(R36C) closed(R36C P16C)
not-locked(R36C)

# telecamere
at(T1C PC1C) at(T2C P10C) at(T3C P6C) at(T4C PC3C) at(T5C PC4C)
at(T6C PC5C) at(T7C P28C) at(T8C P27C)

# sensori antincendio
at(S1C P19C) at(S2C P17C) at(S3C P15C) at(S4C P13C) at(S5C P10C)
at(S6C P8C) at(S7C P6C) at(S8C PC2C) at(S9C P5C) at(S10C P10C)
at(S11C P4C) at(S12C PC6C) at(S13C P28C) at(S14C P30C) at(S15C P32C)
at(S16C P22C) at(S17C P24C) at(S18C P26C)

# punti di ricarica
at(RP1C P18C) at(RP2C P14C) at(RP3C P35aC) at(RP4C PC7C) at(RP5C P1C)
at(RP6C P20C) at(RP7C P29C) at(RP8C P25C)


############# DIPARTIMENTO DI MECCANICA - PIANO SECONDO #############

# connessioni dell'atrio
con(P240M PC2M) con(PC2M P240M) con(PC2M P203M) con(P203M PC2M)
con(P203M P204M) con(P204M) con(P204M P230M) con(P230M P204M)
con(P204M PC3M) con(PC3M P204M) con(P230M PC3M) con(PC3M P230M)
con(PC3M P205M) con(P205M PC3M) con(P205M P206M) con(P206M P205M)
con(P206M PC4M) con(PC4M P206M) con(PC4M P231M) con(P231M PC4M)
con(P231M PC5M) con(PC5M P231M)

# connessioni della parte est
con(PC5M P208M) con(P208M PC5M) con(P208M P209M) con(P209M P208M)
con(P209M P210M) con(P210M P209M) con(P210M P211M) con(P211M P210M)
con(P211M P212M) con(P212M P211M) con(PC5M PC6M) con(PC6M PC5M)
con(PC6M P213M) con(P213M PC6M) con(P213M P214M) con(P214M P213M)
con(P214M P215M) con(P215M P214M) con(P215M P216M) con(P216M P215M)
con(P216M P217M) con(P217M P216M) con(P217M P218M) con(P218M P217M)
con(P218M P219M) con(P219M P218M) con(P219M P220M) con(P220M P219M)

# connessioni della parte ovest
con(P254M PC1M) con(PC1M P254M) con(P254M P253M) con(P253M P254M)
```

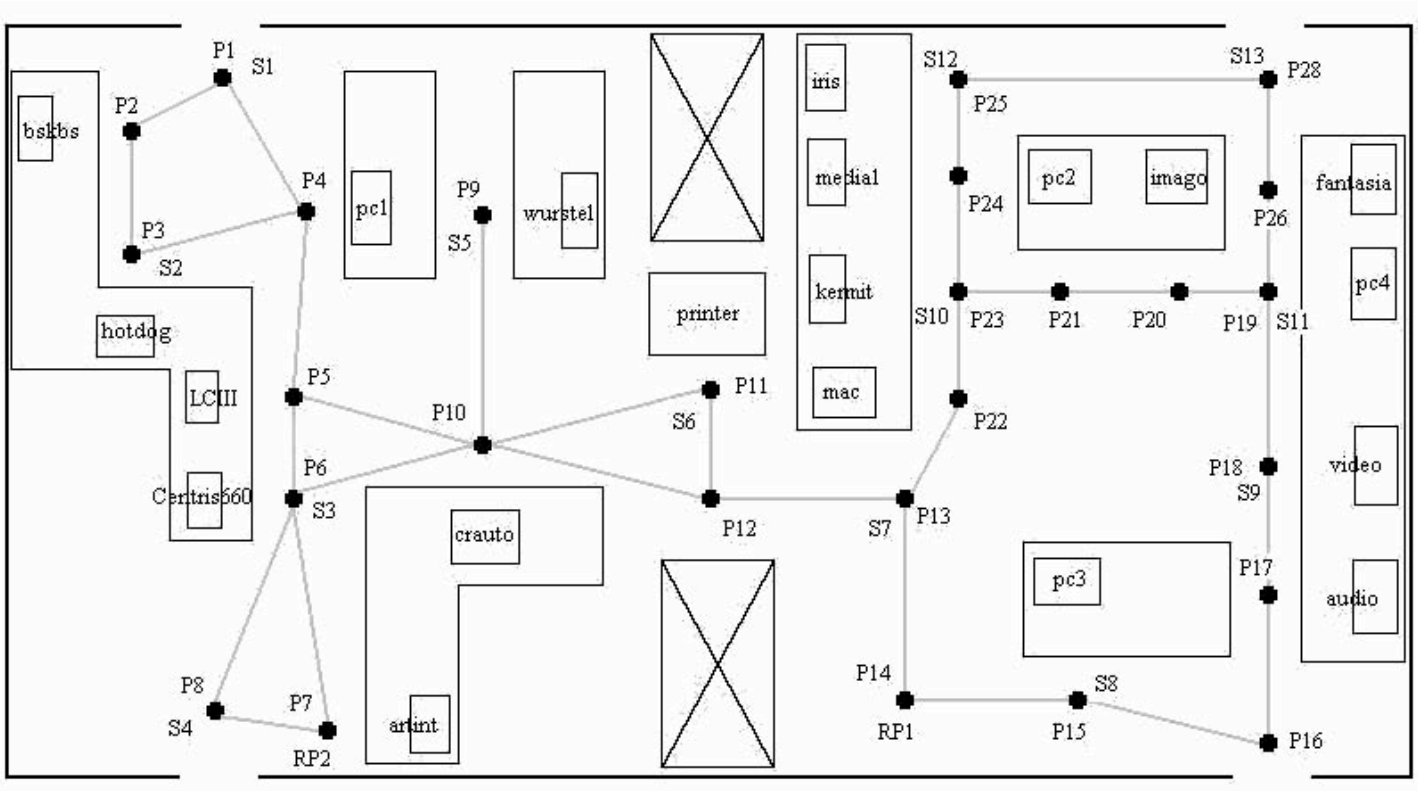
```
con(PC1M P253M) con(P253M PC1M) con(P253M P252M) con(P252M P253M)
con(P252M P251M) con(P251M P252M) con(P251M P260M) con(P260M P251M)
con(P260M P249M) con(P249M P260M) con(P249M P250M) con(P250M P249M)
con(P249M P248M) con(P248M P249M) con(P248M P247M) con(P247M P248M)
con(P247M P246M) con(P246M P247M) con(P246M P245M) con(P245M P246M)
con(P246M P243M) con(P243M P246M) con(P245M P243M) con(P243M P245M)
con(P243M P242M) con(P242M P243M) con(P242M P241M) con(P241M P242M)
con(P241M P240M) con(P240M P241M)

# posizione delle stanze e stato dei relativi sensori e porte
in-front-of(P203M R203M) sensor-on(R203M) closed(R203M P203M)
not-locked(R203M)
in-front-of(P204M R204M) sensor-on(R204M) closed(R204M P204M)
not-locked(R204M)
in-front-of(P205M R205M) sensor-on(R205M) closed(R205M P205M)
not-locked(R205M)
in-front-of(P206M R206M) sensor-on(R206M) closed(R206M P206M)
not-locked(R206M)
in-front-of(P208M R208M) sensor-on(R208M) closed(R208M P208M)
not-locked(R208M)
in-front-of(P209M R209M) sensor-on(R209M) closed(R209M P209M)
not-locked(R209M)
in-front-of(P210M R210M) sensor-on(R210M) closed(R210M P210M)
not-locked(R210M)
in-front-of(P211M R211M) sensor-on(R211M) closed(R211M P211M)
not-locked(R211M)
in-front-of(P212M R212M) sensor-on(R212M) closed(R212M P212M)
not-locked(R212M)
in-front-of(P213M R213M) sensor-on(R213M) closed(R213M P213M)
not-locked(R213M)
in-front-of(P214M R214M) sensor-on(R214M) closed(R214M P214M)
not-locked(R214M)
in-front-of(P215M R215M) sensor-on(R215M) closed(R215M P215M)
not-locked(R215M)
in-front-of(P216M R216M) sensor-on(R216M) closed(R216M P216M)
not-locked(R216M)
in-front-of(P217M R217M) sensor-on(R217M) closed(R217M P217M)
not-locked(R217M)
in-front-of(P218M R218M) sensor-on(R218M) closed(R218M P218M)
not-locked(R218M)
in-front-of(P219M R219M) sensor-on(R219M) closed(R219M P219M)
not-locked(R219M)
in-front-of(P220M R220M) sensor-on(R220M) closed(R220M P220M)
not-locked(R220M)
in-front-of(P220M R221M) sensor-on(R221M) closed(R221M P220M)
not-locked(R221M)
in-front-of(P212M R222M) sensor-on(R222M) closed(R222M P212M)
not-locked(R222M)
in-front-of(P211M R222bM) in-front-of(P216M R222bM) sensor-on(R222bM)
closed(R222bM P211M) closed(R222bM P216M) not-locked(R222bM)
in-front-of(P230M R230M) sensor-on(R230M) closed(R230M P230M)
not-locked(R230M)
in-front-of(P231M R231M) sensor-on(R231M) closed(R231M P231M)
not-locked(R231M)
in-front-of(P240M R240M) sensor-on(R240M) closed(R240M P240M)
not-locked(R240M)
in-front-of(P241M R241M) sensor-on(R241M) closed(R241M P241M)
```

```
not-locked(R241M)
in-front-of(P242M R242M) sensor-on(R242M) closed(R242M P242M)
not-locked(R242M)
in-front-of(P243M R243M) sensor-on(R243M) closed(R243M P243M)
not-locked(R243M)
in-front-of(P245M R244M) sensor-on(R244M) closed(R244M P245M)
not-locked(R244M)
in-front-of(P245M R245M) sensor-on(R245M) closed(R245M P245M)
not-locked(R245M)
in-front-of(P246M R246M) sensor-on(R246M) closed(R246M P246M)
not-locked(R246M)
in-front-of(P247M R247M) sensor-on(R247M) closed(R247M P247M)
not-locked(R247M)
in-front-of(P248M R248M) sensor-on(R248M) closed(R248M P248M)
not-locked(R248M)
in-front-of(P249M R249M) sensor-on(R249M) closed(R249M P249M)
not-locked(R249M)
in-front-of(P250M R250M) sensor-on(R250M) closed(R250M P250M)
not-locked(R250M)
in-front-of(P251M R251M) sensor-on(R251M) closed(R251M P251M)
not-locked(R251M)
in-front-of(P252M R252M) sensor-on(R252M) closed(R252M P252M)
not-locked(R252M)
in-front-of(P253M R253M) sensor-on(R253M) closed(R253M P253M)
not-locked(R253M)
in-front-of(P254M R254M) sensor-on(R254M) closed(R254M P254M)
not-locked(R254M)
in-front-of(PC4 R260M) sensor-on(R260M) closed(R260M PC4)
not-locked(R260M)

# telecamere
at(T4M PC1M) at(T5M P250M) at(T6M P243M) at(T7M PC4M) at(T8M P208M)
at(T9M PC6M) at(T10M P212M)

# sensori antincendio
at(S8M P252M) at(S9M P249M) at(S10M P254M) at(S11M P247M)
at(S12M P245M) at(S13M P241M) at(S14M P203M) at(S15M P205M)
at(S16M P231M) at(S17M P209M) at(S18M P211M) at(S19M P213M)
at(S20M P216M) at(S21M P219M)

# punti di ricarica
at(RP4M P210M) at(RP5M P214M) at(RP6M P218M) at(RP7M P206M)
at(RP8M PC2M) at(RP9M P246M) at(RP10M P251M);

goal:

at(RG PE1)

#################### DIPARTIMENTO DI ELETTRONICA ####################

controlled(R1E) controlled(R2E) controlled(R3E) controlled(R4E)
controlled(R5E) controlled(R6E) controlled(R7E) controlled(R8E)
controlled(R9E) controlled(R10E) controlled(R11E) controlled(R12E)
controlled(R13E) controlled(R14E) controlled(R15E) controlled(R16E)
controlled(R17E) controlled(R18E) controlled(R19E) controlled(R20E)
controlled(R21E) controlled(R22E) controlled(R23E) controlled(R24E)
controlled(R25E) controlled(R26E) controlled(R27E) controlled(R28E)
```

```
controlled(R29E) controlled(R30E) controlled(R31E) controlled(R32E)
controlled(R33E) controlled(R34E) controlled(R35E) controlled(R36E)
controlled(R37E) controlled(R38E) controlled(R38bE) controlled(R39E)
controlled(R40E) controlled(R41E)

sensor-on(R1E) sensor-on(R2E) sensor-on(R3E) sensor-on(R4E)
sensor-on(R5E) sensor-on(R6E) sensor-on(R7E) sensor-on(R8E)
sensor-on(R9E) sensor-on(R10E) sensor-on(R11E) sensor-on(R12E)
sensor-on(R13E) sensor-on(R14E) sensor-on(R15E) sensor-on(R16E)
sensor-on(R17E) sensor-on(R18E) sensor-on(R19E) sensor-on(R20E)
sensor-on(R21E) sensor-on(R22E) sensor-on(R23E) sensor-on(R24E)
sensor-on(R25E) sensor-on(R26E) sensor-on(R27E) sensor-on(R28E)
sensor-on(R29E) sensor-on(R30E) sensor-on(R31E) sensor-on(R32E)
sensor-on(R33E) sensor-on(R34E) sensor-on(R35E) sensor-on(R36E)
sensor-on(R37E) sensor-on(R38E) sensor-on(R38bE) sensor-on(R39E)
sensor-on(R40E) sensor-on(R41E)


############## DIPARTIMENTO DI MECCANICA - PIANO PRIMO ##############


controlled(R103M) controlled(R104M) controlled(R105M)
controlled(R107M) controlled(R108M) controlled(R109M)
controlled(R110M) controlled(R120M) controlled(R121M)

sensor-on(R103M)  sensor-on(R104M)  sensor-on(R105M)  sensor-on(R107M)
sensor-on(R108M)  sensor-on(R109M)  sensor-on(R110M)  sensor-on(R120M)
sensor-on(R121M)


###################### DIPARTIMENTO DI CIVILE ######################


controlled(R1C) controlled(R2C) controlled(R3C) controlled(R4C)
controlled(R5C) controlled(R6C) controlled(R7C) controlled(R8C)
controlled(R9C) controlled(R10C) controlled(R11C) controlled(R12C)
controlled(R13C) controlled(R14C) controlled(R15C) controlled(R16C)
controlled(R17C) controlled(R18C) controlled(R19C) controlled(R20C)
controlled(R21C) controlled(R22C) controlled(R23C) controlled(R24C)
controlled(R25C) controlled(R26C) controlled(R27C) controlled(R28C)
controlled(R29C) controlled(R30C) controlled(R31C) controlled(R32C)
controlled(R33C) controlled(R34C) controlled(R35C) controlled(R36C)

sensor-on(R1C) sensor-on(R2C) sensor-on(R3C) sensor-on(R4C)
sensor-on(R5C) sensor-on(R6C) sensor-on(R7C) sensor-on(R8C)
sensor-on(R9C) sensor-on(R10C) sensor-on(R11C) sensor-on(R12C)
sensor-on(R13C) sensor-on(R14C) sensor-on(R15C) sensor-on(R16C)
sensor-on(R17C) sensor-on(R18C) sensor-on(R19C) sensor-on(R20C)
sensor-on(R21C) sensor-on(R22C) sensor-on(R23C) sensor-on(R24C)
sensor-on(R25C) sensor-on(R26C) sensor-on(R27C) sensor-on(R28C)
sensor-on(R29C) sensor-on(R30C) sensor-on(R31C) sensor-on(R32C)
sensor-on(R33C) sensor-on(R34C) sensor-on(R35C) sensor-on(R36C)


############# DIPARTIMENTO DI MECCANICA - PIANO SECONDO #############
```

```
controlled(R203M) controlled(R204M) controlled(R205M)
controlled(R206M) controlled(R208M) controlled(R209M)
controlled(R210M) controlled(R211M) controlled(R212M)
controlled(R213M) controlled(R214M) controlled(R215M)
controlled(R216M) controlled(R217M) controlled(R218M)
controlled(R219M) controlled(R220M) controlled(R221M)
controlled(R222M) controlled(R222bM) controlled(R230M)
controlled(R231M) controlled(R240M) controlled(R241M)
controlled(R242M) controlled(R243M) controlled(R244M)
controlled(R245M) controlled(R246M) controlled(R247M)
controlled(R248M) controlled(R249M) controlled(R250M)
controlled(R251M) controlled(R252M) controlled(R253M)
controlled(R254M) controlled(R260M) controlled(R261M)

sensor-on(R203M) sensor-on(R204M) sensor-on(R205M) sensor-on(R206M)
sensor-on(R208M) sensor-on(R209M) sensor-on(R210M) sensor-on(R211M)
sensor-on(R212M) sensor-on(R213M) sensor-on(R214M) sensor-on(R215M)
sensor-on(R216M) sensor-on(R217M) sensor-on(R218M) sensor-on(R219M)
sensor-on(R220M) sensor-on(R221M) sensor-on(R222M) sensor-on(R222bM)
sensor-on(R230M) sensor-on(R231M) sensor-on(R240M) sensor-on(R241M)
sensor-on(R242M) sensor-on(R243M) sensor-on(R244M) sensor-on(R245M)
sensor-on(R246M) sensor-on(R247M) sensor-on(R248M) sensor-on(R249M)
sensor-on(R250M) sensor-on(R251M) sensor-on(R252M) sensor-on(R253M)
sensor-on(R254M) sensor-on(R260M) sensor-on(R261M);
```

 # laboratorio d'informatica del dipartimento di elettronica

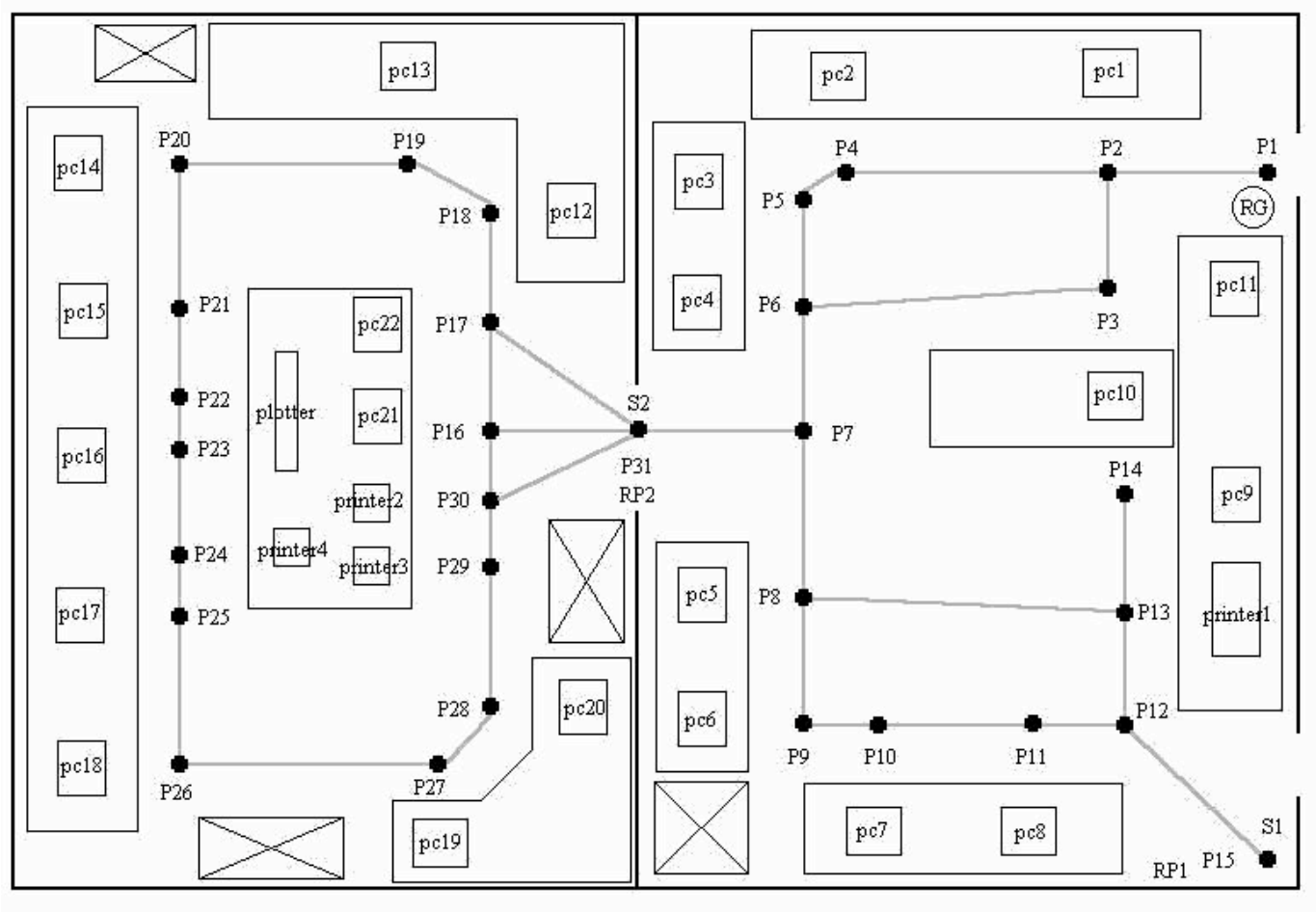


 # r-el.fct

```
alarm: A;
robot: RG;
#object: ;
precious-object: bskbs hotdog LCIII Centris660 artint crauto pc1
                 wurstel printer mac kermit media1 iris
                 pc2 imago pc3 fantasia pc4 video audio;
#window: ;
#safe-box: ;
light: L;
location: P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12 P13 P14 P15 P16 P17
          P18 P19 P20 P21 P22 P23 P24 P25 P26 P27 P28 ;
fire: S1 S2 S3 S4 S5 S6 S7 S8 S9 S10 S11 S12;
refuel-point: RP1 RP2;

$fuel(): 0 600;

database:

cost(P1 P2) 2 cost(P2 P1) 2 cost(P2 P3) 3 cost(P3 P2) 3
cost(P1 P4) 3 cost(P4 P1) 3 cost(P3 P4) 3 cost(P4 P3) 3
cost(P1 P3) 4 cost(P3 P1) 4 cost(P2 P4) 3 cost(P4 P2) 3
cost(P4 P5) 4 cost(P5 P4) 4 cost(P5 P6) 2 cost(P6 P5) 2
cost(P6 P7) 4 cost(P7 P6) 4 cost(P7 P8) 2 cost(P8 P7) 2
cost(P8 P6) 4 cost(P6 P8) 4 cost(P5 P10) 3 cost(P10 P5) 3
cost(P6 P10) 3 cost(P10 P6) 3 cost(P10 P9) 4 cost(P9 P10) 4
cost(P10 P11) 4 cost(P11 P10) 4 cost(P10 P12) 4 cost(P12 P10) 4
cost(P11 P12) 2 cost(P12 P11) 2 cost(P12 P13) 3 cost(P13 P12) 3
cost(P13 P14) 3 cost(P14 P13) 3 cost(P14 P15) 3 cost(P15 P14) 3
cost(P15 P16) 3 cost(P16 P15) 3 cost(P16 P17) 3 cost(P17 P16) 3
cost(P17 P18) 2 cost(P18 P17) 2 cost(P18 P13) 5 cost(P13 P18) 5
cost(P13 P22) 2 cost(P22 P13) 3 cost(P18 P19) 3 cost(P19 P18) 3
cost(P22 P19) 5 cost(P19 P22) 5 cost(P22 P23) 3 cost(P23 P22) 3
cost(P23 P21) 2 cost(P21 P23) 2 cost(P21 P20) 2 cost(P20 P21) 2
cost(P20 P19) 2 cost(P19 P20) 2 cost(P23 P24) 2 cost(P24 P23) 2
cost(P24 P25) 2 cost(P25 P24) 2 cost(P25 P28) 4 cost(P28 P25) 4
cost(P28 P26) 3 cost(P26 P28) 3 cost(P26 P19) 2 cost(P19 P26) 2
cost(P21 P18) 4 cost(P18 P21) 4;

initial:

=($fuel() 600)

image-off(RG)
off(L)
at(RG P8) has-fuel(RG)

at(bskbs P2) at(hotdog P3) at(LCIII P5)  at(Centris660 P6)
at(artint P7) at(crauto P10) at(pc1 P9)  at(wurstel P9)
at(printer P11) at(mac P22) at(kermit P23) at(media1 P24)
at(iris P25) at(pc2 P21) at(imago P20) at(fantasia P26)
at(pc4 P19) at(video P18) at(audio P17) at(pc3 P15)


present-complete(bskbs P2) present-complete(hotdog P3)
```

```
present-complete(LCIII P5)  present-complete(Centris660 P6)
present-complete(artint P7) present-complete(crauto P10)
present-complete(pc1 P9)  present-complete(wurstel P9)
present-complete(printer P11) present-complete(mac P22)
present-complete(kermit P23) present-complete(media1 P24)
present-complete(iris P25) present-complete(pc2 P21)
present-complete(imago P20) present-complete(fantasia P26)
present-complete(pc4 P19) present-complete(video P18)
present-complete(audio P17) present-complete(pc3 P15)

con(P1 P2) con(P2 P1) con(P2 P3) con(P3 P2) con(P1 P4) con(P4 P1)
con(P3 P4) con(P4 P3) con(P1 P3) con(P3 P1) con(P2 P4) con(P4 P2)
con(P4 P5) con(P5 P4) con(P5 P6) con(P6 P5) con(P6 P7) con(P7 P6)
con(P7 P8) con(P8 P7) con(P8 P6) con(P6 P8) con(P5 P10) con(P10 P5)
con(P6 P10) con(P10 P6) con(P10 P9) con(P9 P10)
con(P10 P11) con(P11 P10) con(P10 P12) con(P12 P10) con(P11 P12)
con(P12 P11) con(P12 P13) con(P13 P12) con(P13 P14) con(P14 P13)
con(P14 P15) con(P15 P14) con(P15 P16) con(P16 P15) con(P16 P17)
con(P17 P16) con(P17 P18) con(P18 P17) con(P18 P13) con(P13 P18)
con(P13 P22) con(P22 P13) con(P18 P19) con(P19 P18) con(P22 P19)
con(P19 P22) con(P22 P23) con(P23 P22) con(P23 P21) con(P21 P23)
con(P21 P20) con(P20 P21) con(P20 P19) con(P19 P20) con(P23 P24)
con(P24 P23) con(P24 P25) con(P25 P24) con(P25 P28) con(P28 P25)
con(P28 P26) con(P26 P28) con(P26 P19) con(P19 P26) con(P21 P18)
con(P18 P21)

at(S1 P1) at(S2 P3) at(S3 P6) at(S4 P8) at(S5 P9) at(S6 P11)
at(S7 P13) at(S8 P15) at(S9 P18) at(S10 P23) at(S11 P19) at(S12 P25)

at(RP1 P14) at(RP2 P7);


goal:

at(RG P8)

controlled(bskbs) controlled(hotdog) controlled(LCIII)
controlled(Centris660)  controlled(artint) controlled(crauto)
controlled(pc1) controlled(wurstel) controlled(printer)
controlled(mac)  controlled(kermit) controlled(media1)
controlled(iris) controlled(pc2) controlled(imago) controlled(pc3)
controlled(fantasia) controlled(pc4) controlled(video)
controlled(audio) controlled();
```

 # aula CAD del dipartimento di ingegneria meccanica



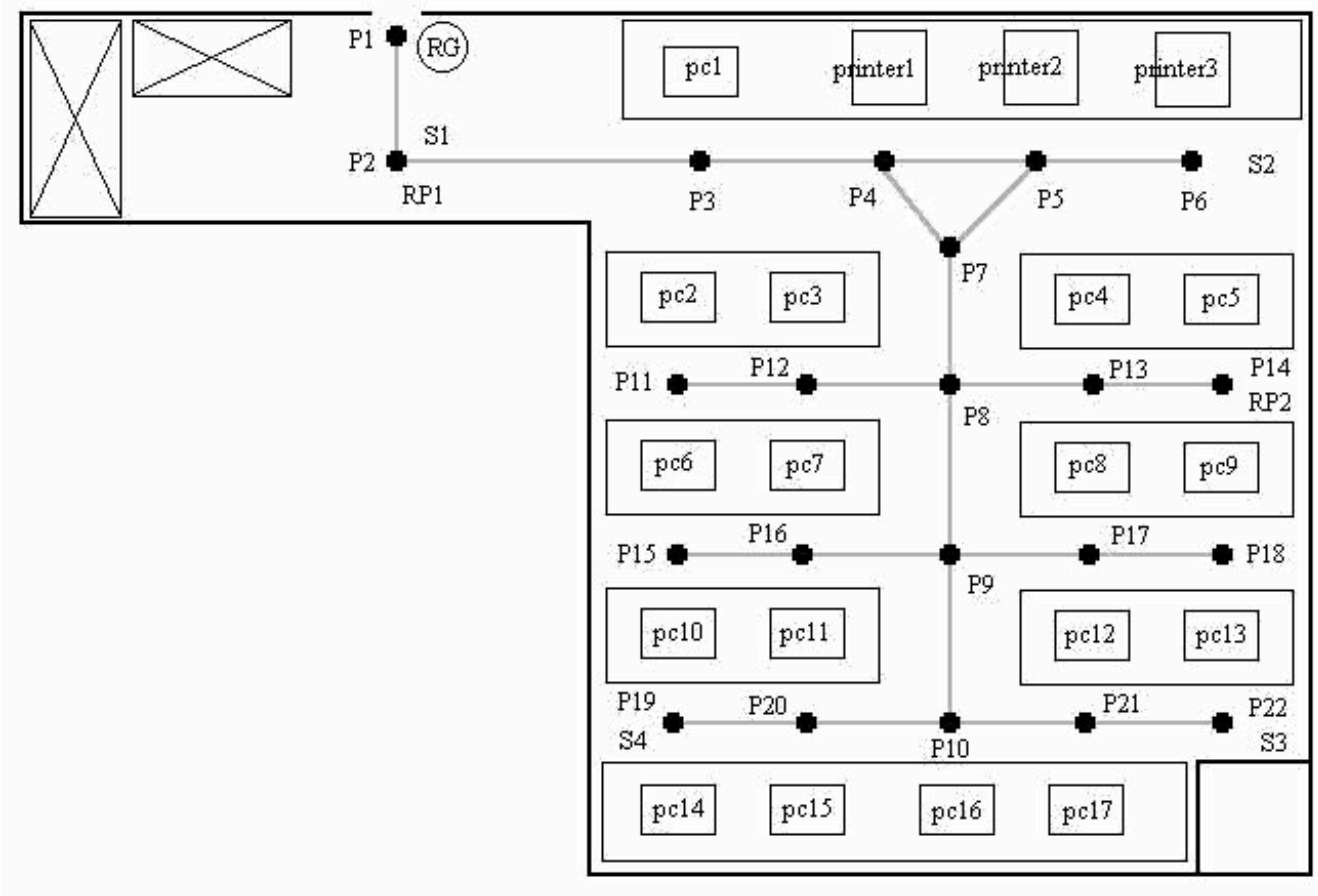
 # r-mec.fct

```
alarm: A;
robot: RG;
object: pc1 pc2 pc3 pc4 pc5 pc6 pc7 pc8 pc9 pc10 pc11 pc12 pc13 pc14
        pc15 pc16 pc17 pc18 pc19 pc20 pc21 pc22;
precious-object: printer1 printer2 printer3 printer4 plotter;
# window: ;
# safe-box: ;
light: L;
location: P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12 P13 P14 P15 P16 P17
          P18 P19 P20 P21 P22 P23 P24 P25 P26 P27 P28 P29 P30 P31;
fire: S1;
refuel-point: RP1 RP2;

$fuel(): 0 600;

database:

cost(P1 P2) 3 cost(P2 P1) 2
cost(P2 P3) 3 cost(P3 P2) 3
cost(P6 P3) 5 cost(P3 P6) 5
cost(P2 P4) 4 cost(P4 P2) 4
```

```
cost(P4 P5) 2 cost(P5 P4) 2
cost(P5 P6) 3 cost(P6 P5) 3
cost(P6 P7) 3 cost(P7 P6) 3
cost(P7 P8) 4 cost(P8 P7) 4
cost(P8 P9) 2 cost(P9 P8) 2
cost(P8 P13) 5 cost(P13 P8) 5
cost(P9 P10) 2 cost(P10 P9) 2
cost(P11 P10) 3 cost(P10 P11) 3
cost(P11 P12) 2 cost(P12 P11) 2
cost(P15 P12) 4 cost(P12 P15) 4
cost(P13 P12) 3 cost(P12 P13) 3
cost(P13 P14) 3 cost(P14 P13) 3
cost(P7 P31) 4 cost(P31 P7) 4
cost(P31 P30) 4 cost(P30 P31) 4
cost(P30 P29) 2 cost(P29 P30) 2
cost(P29 P28) 3 cost(P28 P29) 3
cost(P28 P27) 2 cost(P27 P28) 2
cost(P27 P26) 4 cost(P26 P27) 4
cost(P26 P25) 3 cost(P25 P26) 3
cost(P25 P24) 2 cost(P24 P25) 2
cost(P24 P23) 3 cost(P23 P24) 3
cost(P23 P22) 2 cost(P22 P23) 2
cost(P22 P21) 3 cost(P21 P22) 3
cost(P21 P20) 4 cost(P20 P21) 4
cost(P20 P19) 5 cost(P19 P20) 5
cost(P19 P18) 3 cost(P18 P19) 3
cost(P18 P17) 3 cost(P17 P18) 3
cost(P17 P31) 4 cost(P31 P17) 4
cost(P31 P16) 3 cost(P16 P31) 3
cost(P16 P30) 2 cost(P30 P16) 2
cost(P16 P17) 3 cost(P17 P16) 3;

initial:

=($fuel() 600)

image-off(RG)
off(L)
at(RG P1) has-fuel(RG)

at(pc1 P2) present(pc1 P2)
at(pc2 P4) present(pc2 P4)
at(pc3 P5) present(pc3 P5)
at(pc4 P6) present(pc4 P6)
at(pc5 P8) present(pc5 P8)
at(pc6 P9) present(pc6 P9)
at(pc7 P10) present(pc7 P10)
at(pc8 P11) present(pc8 P11)
at(pc9 P14) present(pc9 P14)
at(pc10 P14) present(pc10 P14)
at(pc11 P3) present(pc11 P3)
at(pc12 P18) present(pc12 P18)
at(pc13 P19) present(pc13 P19)
at(pc14 P20) present(pc14 P20)
at(pc15 P21) present(pc15 P21)
at(pc16 P23) present(pc16 P23)
at(pc17 P25) present(pc17 P25)
```

```
at(pc18 P26) present(pc18 P26)
at(pc19 P27) present(pc19 P27)
at(pc20 P28) present(pc20 P28)
at(pc21 P16) present(pc21 P16)
at(pc22 P17) present(pc22 P17)

at(printer1 P13) present-complete(printer1 P13)
at(printer2 P30) present-complete(printer2 P30)
at(printer3 P29) present-complete(printer3 P29)
at(printer3 P24) present-complete(printer3 P24)
at(plotter P22) present-complete(plotter P22)

con(P1 P2) con(P2 P1) con(P2 P3) con(P3 P2) con(P6 P3) con(P3 P6)
con(P2 P4) con(P4 P2) con(P4 P5) con(P5 P4) con(P5 P6) con(P6 P5)
con(P6 P7) con(P7 P6) con(P7 P8) con(P8 P7)  con(P8 P9) con(P9 P8)
con(P8 P13) con(P13 P8) con(P9 P10) con(P10 P9)
con(P11 P10) con(P10 P11) con(P11 P12) con(P12 P11)
con(P15 P12) con(P12 P15) con(P13 P12) con(P12 P13)
con(P13 P14) con(P14 P13) con(P7 P31) con(P31 P7)
con(P31 P30) con(P30 P31) con(P30 P29) con(P29 P30)
con(P29 P28) con(P28 P29) con(P28 P27) con(P27 P28)
con(P27 P26) con(P26 P27) con(P26 P25) con(P25 P26)
con(P25 P24) con(P24 P25) con(P24 P23) con(P23 P24)
con(P23 P22) con(P22 P23) con(P22 P21) con(P21 P22)
con(P21 P20) con(P20 P21) con(P20 P19) con(P19 P20)
con(P19 P18) con(P18 P19) con(P18 P17) con(P17 P18)
con(P17 P31) con(P31 P17) con(P31 P16) con(P16 P31)
con(P16 P30) con(P30 P16) con(P16 P17) con(P17 P16)

at(S1 P15)

at(RP1 P15) at(RP2 P31);


goal:

at(RG P1)
controlled(pc1) controlled(pc2)
controlled(pc3) controlled(pc4)
controlled(pc5) controlled(pc6)
controlled(pc7) controlled(pc8)
controlled(pc9) controlled(pc10)
controlled(pc11) controlled(pc12)
controlled(pc13) controlled(pc14)
controlled(pc15) controlled(pc16)
controlled(pc17) controlled(pc18)
controlled(pc19) controlled(pc20)
controlled(pc21) controlled(pc22)
controlled(printer1)
controlled(printer2)
controlled(printer3)
controlled(printer4)
controlled(plotter)
controlled();
```

 # aula tesisti del dipartimento di ingegneria civile



 # r-civ.fct

```
alarm: A;
robot: RG;
object: pc1 pc2 pc3 pc4 pc5 pc6 pc7 pc8 pc9 pc10 pc11 pc12 pc13 pc14
        pc15 pc16 pc17;
precious-object: printer1 printer2 printer3;
# window: ;
# safe-box: ;
light: L;
location: P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12 P13 P14 P15 P16 P17
          P18 P19 P20 P21 P22;
fire: S1 S2 S3;
refuel-point: RP1 RP2;

$fuel(): 0 600;

database:

cost(P1 P2) 3 cost(P2 P1) 3
cost(P2 P3) 4 cost(P3 P2) 4
cost(P3 P4) 3 cost(P4 P3) 3
cost(P4 P5) 3 cost(P5 P4) 3
cost(P4 P7) 3 cost(P7 P4) 3
cost(P5 P6) 3 cost(P6 P5) 3
cost(P5 P7) 3 cost(P7 P5) 3
cost(P7 P8) 4 cost(P8 P7) 4
```

```
cost(P8 P9) 4 cost(P9 P8) 4
cost(P9 P10) 4 cost(P10 P9) 4
cost(P11 P12) 3 cost(P12 P11) 3
cost(P12 P8) 3 cost(P8 P12) 3
cost(P13 P8) 3 cost(P8 P13) 3
cost(P14 P13) 3 cost(P13 P14) 3
cost(P15 P16) 3 cost(P16 P15) 3
cost(P16 P9) 3 cost(P9 P16) 3
cost(P9 P17) 3 cost(P17 P9) 3
cost(P17 P18) 3 cost(P18 P17) 3
cost(P19 P20) 3 cost(P20 P19) 3
cost(P10 P20) 3 cost(P20 P10) 3
cost(P10 P21) 3 cost(P21 P10) 3
cost(P21 P22) 3 cost(P22 P21) 3;

initial:

=($fuel() 600)

image-off(RG)
off(L)
at(RG P1) has-fuel(RG)

at(pc1 P3) present(pc1 P3)
at(pc2 P11) present(pc2 P11)
at(pc3 P12) present(pc3 P12)
at(pc4 P13) present(pc4 P13)
at(pc5 P14) present(pc5 P14)
at(pc6 P15) present(pc6 P15)
at(pc7 P16) present(pc7 P16)
at(pc8 P17) present(pc8 P17)
at(pc9 P18) present(pc9 P18)
at(pc10 P19) present(pc10 P19)
at(pc11 P20) present(pc11 P20)
at(pc12 P21) present(pc12 P21)
at(pc13 P22) present(pc13 P22)
at(pc14 P19) present(pc14 P19)
at(pc15 P20) present(pc15 P20)
at(pc16 P10) present(pc16 P10)
at(pc17 P21) present(pc17 P21)

at(printer1 P4) present-complete(printer1 P4)
at(printer2 P5) present-complete(printer2 P5)
at(printer3 P6) present-complete(printer3 P6)

con(P1 P2) con(P2 P1) con(P2 P3) con(P3 P2)
con(P3 P4) con(P4 P3) con(P4 P5) con(P5 P4)
con(P4 P7) con(P7 P4) con(P5 P6) con(P6 P5)
con(P5 P7) con(P7 P5) con(P7 P8) con(P8 P7)
con(P8 P9) con(P9 P8) con(P9 P10) con(P10 P9)
con(P11 P12) con(P12 P11) con(P12 P8) con(P8 P12)
con(P13 P8) con(P8 P13) con(P14 P13) con(P13 P14)
con(P15 P16) con(P16 P15) con(P16 P9) con(P9 P16)
con(P9 P17) con(P17 P9) con(P17 P18) con(P18 P17)
con(P19 P20) con(P20 P19) con(P10 P20) con(P20 P10)
con(P10 P21) con(P21 P10) con(P21 P22) con(P22 P21)
```

```
at(S1 P2) at(S2 P6) at(S3 P22)

at(RP1 P2) at(RP2 P14);


goal:

at(RG P1)
controlled(pc1) controlled(pc2)
controlled(pc3) controlled(pc4)
controlled(pc5) controlled(pc6)
controlled(pc7) controlled(pc8)
controlled(pc9) controlled(pc10)
controlled(pc11) controlled(pc12)
controlled(pc13) controlled(pc14)
controlled(pc15) controlled(pc16) controlled(pc17)
controlled(printer1)
controlled(printer2)
controlled(printer3)
controlled();
```