UNIVERSITÀ DI BRESCIA
FACOLTÀ DI INGEGNERIA
Dipartimento di Elettronica per l'Automazione

Laboratorio di Robotica Avanzata
Advanced Robotics Laboratory

Corso di Robotica Mobile
(Prof. Riccardo Cassinis)

Navigazione di Tobor con mappe derivate da URG

Elaborato di esame di:

Alessandro Acerbis, Marco Ambrosini, Paolo Cacciaguida

Consegnato il: **10 luglio 2012**

Sommario

Il lavoro svolto consiste nello sviluppo di un software, per il robot Tobor, che permetta la navigazione dello stesso con mappe derivate dal sensore laser URG-04LX della Hokuyo. L'obiettivo è di creare una mappa da fornire al robot, farlo localizzare all'interno dell'ambiente mappato, pianificare tramite l'algoritmo A il percorso ottimo per raggiungere un punto prestabilito e percorrere la traiettoria calcolata.*

1. Introduzione

In questo capitolo saranno presentate le condizioni ambientali necessarie al funzionamento del software, le caratteristiche del robot utilizzato e l'approccio al problema.

1.1. Precondizioni sull'ambiente

Il corretto funzionamento del programma realizzato può essere garantito soltanto se si rispettano le precondizioni di seguito esposte.

La più importante è la necessità di mantenere un ambiente invariato rispetto al momento della creazione della mappa. Questo è indispensabile perché sia la localizzazione sia la pianificazione del percorso ottimo si basano sulla mappa fornita, e nel caso in cui ci fossero differenze significative tra la mappa e l'ambiente reale, chiaramente queste due attività non potrebbero più essere svolte correttamente.

Ulteriori vincoli sono imposti dai sensori utilizzati dal robot per la localizzazione e la navigazione. Infatti la localizzazione risulta tanto più precisa quanto più gli ostacoli sono regolari. In particolare nel laboratorio di robotica, dove sono stati svolti i test, elementi come sedie e tavoli hanno creato qualche problema specialmente in fase di localizzazione.

1.2. Robot utilizzato

Il software prodotto è stato testato sul robot Tobor, appartenente alla serie Pioneer 1, prodotta da MobileRobots.

Questo robot anolonomo è dotato di tre ruote, due motrici fisse poste nella parte anteriore e una folle e pivottante nella parte posteriore. Per il movimento utilizza il meccanismo del differential drive. Tobor è equipaggiato anche con una pinza nella parte anteriore. Le due ganasce della pinza incorporano dei sensori di pressione, che permettono di identificare il contatto con un ostacolo. Nella parte anteriore sono presenti sette sonar, e il sensore laser URG-04LX.

Il laser può rilevare da una distanza minima di 20mm a una massima di 5600mm con un angolo di scansione di 240°. L'errore è di 10mm per le rilevazioni effettuate tra i 20mm e i 1000mm, e dell'1% per le rilevazioni a distanza superiore.

Tobor è anche equipaggiato con un calcolatore portatile Acer Aspire One, che permette l'accesso da remoto, oltre alla compilazione e all'esecuzione dei programmi basati sulle librerie Aria e Arnl, entrambe indispensabili in questo progetto.

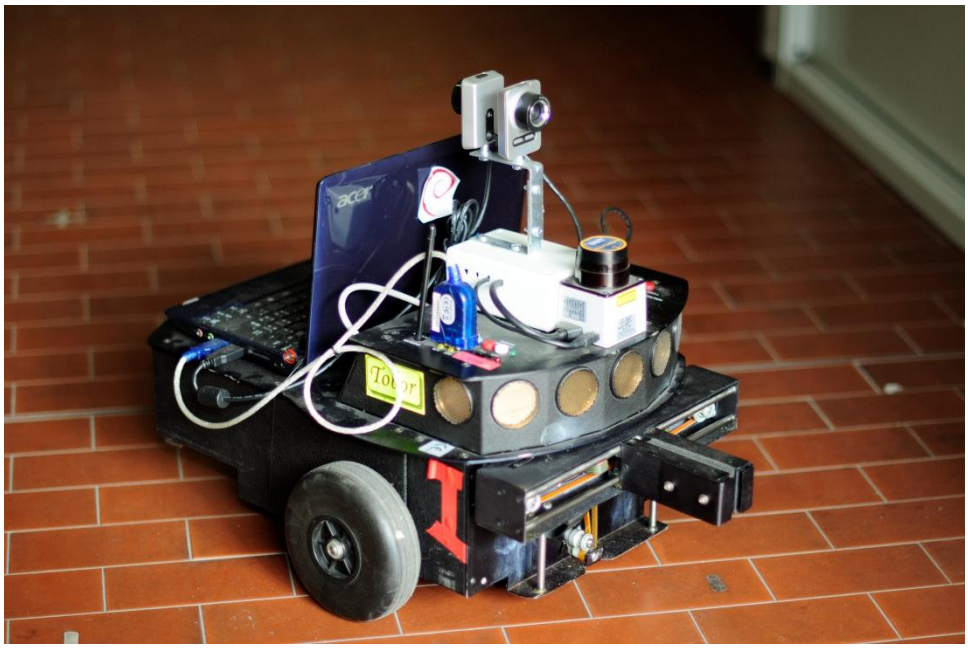


Figura 1. 1 - Il robot Tobor

1.3. Approccio al problema

L'idea di base è stata la seguente: una volta creata la mappa dell'ambiente in cui il robot dovrà navigare (in un momento precedente all'esecuzione, usando il software MobileEyes e Mapper3), dovrà essere svolta una fase di "wander", cioè di navigazione casuale sicura, durante la quale sarà avviata anche la procedura di localizzazione. Ottenuta una certa confidenza nella stima della posizione, il robot dovrà interrompere la fase di "wander", pianificare il percorso ottimo per raggiungere dalla posizione corrente un obiettivo prefissato (goal) percorrendo la traiettoria calcolata.

La complessità globale del progetto ha reso necessario suddividere il lavoro nei seguenti sottoproblemi:

- Creazione della mappa del laboratorio ARL per punti;
- Autolocalizzazione del robot all'interno della mappa;
- Discretizzazione della mappa in una matrice di celle libere e occupate, di granularità desiderata;
- Growing degli ostacoli nella mappa discretizzata, effettuato in base alle dimensioni del robot;
- Pianificazione del percorso ottimo per raggiungere il goal utilizzando l'algoritmo A*;
- Navigazione lungo la traiettoria calcolata fino al raggiungimento del goal.

2. Il problema affrontato

In questo capitolo saranno esposti più in dettaglio i diversi sottoproblemi precedentemente evidenziati.

2.1. Creazione della mappa del laboratorio ARL

Questa prima fase è stata quella meno collegata alle successive, in quanto volta a creare una mappa da fornire poi data in input al programma vero e proprio, il quale risolve tutte le problematiche successive.

Per fare ciò è stato eseguito sul calcolatore montato su Tobor il software ArnlServer; il programma client MobileEyes è stato quindi eseguito su un altro calcolatore e connesso ad ArnlServer. Sono state sfruttate le funzionalità di "wander" e di scansione dell'ambiente per creare un file con estensione ".2d" contenente tutte le informazioni necessarie per generare una mappa a punti. Queste informazioni sono state ottenute tramite il laser URG montato su Tobor.

Successivamente è stato elaborato il file ".2d" con il software Mapper3, al fine di produrre un file ".map" rappresentante la vera e propria mappa dell'ambiente.

Questa attività è stata il punto di partenza per le successive elaborazioni, che risultano di fatto indipendenti da essa. È infatti possibile creare la mappa in un momento precedente rispetto all'esecuzione del software di localizzazione e pianificazione, a patto naturalmente che l'ambiente non subisca variazioni significative, come già introdotto nel paragrafo 1.1.

2.2. Autolocalizzazione del robot all'interno della mappa

Nota la mappa, il robot dovrà esplorare l'ambiente confrontando le rilevazioni dei suoi sensori con le informazioni presenti nella mappa. L'obiettivo è stato quindi trovare una corrispondenza tra mappa e ambiente, in modo da tradurre le coordinate di Tobor in un punto della mappa, che sarà poi utilizzato come posizione di partenza per la successiva fase di pianificazione.

Quello appena descritto è stato il primo passo implementato all'interno del software prodotto in questo progetto.

2.3. Discretizzazione della mappa

Per calcolare il percorso ottimo da un punto di partenza al goal è stato necessario trasformare il file ".map" in una matrice di celle che possono assumere i valori mutuamente esclusivi di "libero" o "occupato" a seconda della presenza o meno di ostacoli nell'area da esse rappresentata.

La matrice opera una discretizzazione dell'ambiente, in cui ogni cella rappresenta una porzione dell'ambiente reale. La granularità è definita tramite un parametro. A un valore di granularità più elevato corrisponde una dimensione maggiore di ogni cella, e quindi una mappa meno dettagliata, composta da meno celle. Al contrario, un valore di granularità più basso determina una mappa composta da più celle, e quindi più dettagliata.

2.4. Growing

Poiché in generale la dimensione delle celle della matrice sarà inferiore rispetto alla dimensione del robot (per non perdere eccessivamente dettaglio), è stato necessario trovare un modo per stabilire se il robot si trovi o no in una certa cella. Ciò è stato possibile considerando il robot come puntiforme. Naturalmente non è stato sufficiente assumere che il robot fosse un punto lasciando la mappa invariata: così facendo infatti si sarebbe rischiato di pianificare una traiettoria percorribile dal robot modellizzato ma non da quello reale, avente una dimensione maggiore.

È stato quindi necessario operare un growing sulla matrice, in modo da ingrandire le posizioni "occupate" di una quantità sufficiente ad evitare collisioni con gli ostacoli reali. Per la successiva fase di pianificazione è stata usata la matrice risultante dal growing, ottenendo così la garanzia che le celle appartenenti alla traiettoria ottima calcolata fossero realmente percorribili dal robot.

2.5. Pianificazione del percorso ottimo

Dopo aver effettuato il growing della mappa, il passo successivo è stato la pianificazione di un percorso ottimo tra il punto in cui si trova il robot al momento della localizzazione ed il punto d'arrivo dato in input.

Il raggiungimento di questo obiettivo si è basato sulla certezza, data dal growing, che ogni cella della mappa classificata come "libera" fosse sicuramente priva di ostacoli e percorribile dal robot.

2.6. Navigazione lungo il percorso ottimo

Il problema finale affrontato è stato, a valle di tutti i calcoli eseguiti in precedenza, permettere al robot di seguire la traiettoria ottima.

3. La soluzione adottata

Questo capitolo presenta passo per passo la strategia adottata e tutte le scelte computazionali che il gruppo ha messo in atto per la risoluzione del problema.

Come visto precedentemente, il lavoro è stato scomposto in diversi sotto obiettivi interdipendenti, che hanno richiesto l'applicazione di diverse conoscenze acquisite in questo ed altri corsi.

Avendo già approfondito la fase preliminare di creazione della mappa, verranno di seguito descritte le fasi successive ad essa.

Per lo sviluppo del software è stato modificato il file "arnlServer.cpp", il quale permette il collegamento di programmi client come MobileEyes, utile ad esempio in fase di localizzazione.

3.1. Autolocalizzazione del robot all'interno della mappa

Per questa attività ci si è avvalsi della localizzazione inclusa in Arnl, che è di tipo Markoviano.

Associando questo algoritmo ad un comportamento di "wander", cioè di movimento libero e casuale per l'ambiente evitando gli ostacoli, il robot scansiona attraverso laser e sonar ciò che lo circonda ed identifica la sua posizione nella mappa con una certa percentuale di confidenza.

Trattandosi di una localizzazione di tipo probabilistico è stata scelta, attraverso opportune prove, una soglia di confidenza sopra la quale si considera esatta la posizione stimata dal robot.

Nello specifico, il gruppo ha ritenuto idonea allo scopo una percentuale dell'85%. Tuttavia si è visto sperimentalmente che non sempre il robot riesce a raggiungere questa soglia in tempi ragionevoli. Si è perciò deciso di interrompere l'esecuzione del programma nel caso la localizzazione non porti risultati accettabili nell'arco di 90 secondi.

Quando il robot si localizza, questo si ferma nella posizione individuata. I calcoli successivi iniziano solo dopo un tempo di attesa di qualche secondo. Ciò è stato fatto per distinguere il momento della localizzazione dal successivo instradamento lungo il tragitto ottimo.

3.2. Discretizzazione della mappa

Uno dei parametri di input al software è la mappa in formato ".map" elaborata precedentemente.

Per poter calcolare la traiettoria ottima si è reso necessario discretizzarla. Il file ".map" contiene già un insieme di punti e può quindi essere considerato esso stesso una discretizzazione ma, per esigenze di calcolo, è stato indispensabile tradurlo in una matrice in cui ogni cella rappresenti una porzione dell'ambiente.

Dato che a dimensioni di cella diverse corrisponde granularità, quindi precisione, diversa, si è scelto di richiedere questo valore come parametro di input in mm (ad esempio `–cella 40` significa che ogni cella mappa una porzione dell'ambiente di 40mm x 40mm).

Si è quindi proceduto alla lettura del file ".map", contenente, oltre che l'insieme dei punti occupati dagli ostacoli, anche una serie di informazioni aggiuntive utili ai calcoli successivi. In particolare sono state utilizzate le coordinate minima e massima in entrambe le direzioni e il numero di punti rappresentati.

Inizialmente, per stabilire le dimensioni della matrice, si è utilizzata la formula

$$\frac{Coordinata\_massima - Coordinata\_minima}{dimensione\_cella}$$

scegliendo le coordinate y e x rispettivamente per righe e colonne.

Per assegnare poi il valore di "libera" (0) oppure "occupata" (1) ad ogni cella della matrice è stata pensata una formula di corrispondenza univoca tra coordinate del file ".map" e celle.

Considerando un punto mappato nelle coordinate (x,y), esso ricadrà nella cella $M_{i,j}$ della mappa dove:

$$i = \left\lfloor \left| \frac{coordinata\_\max\_y - y}{dimensione\_cella} \right| \right\rfloor - 1 \qquad (1)$$

$$j = \left\lfloor \frac{x + |coordinata\_\min\_x|}{dimensione\_cella} \right\rfloor - 1 \qquad (2)$$

Le formule acquistano maggior chiarezza considerando il problema dei diversi sistemi di coordinate adottati per la creazione della mappa e per la matrice. Il primo infatti pone l'origine nel punto di partenza della scansione, il secondo nell'angolo in alto a sinistra della matrice. Inoltre il semiasse positivo della y è diretto verso l'alto nella mappa, mentre nella matrice l'indice di riga cresce andando verso il basso. Tutto ciò è rappresentato nella figura 3.1.

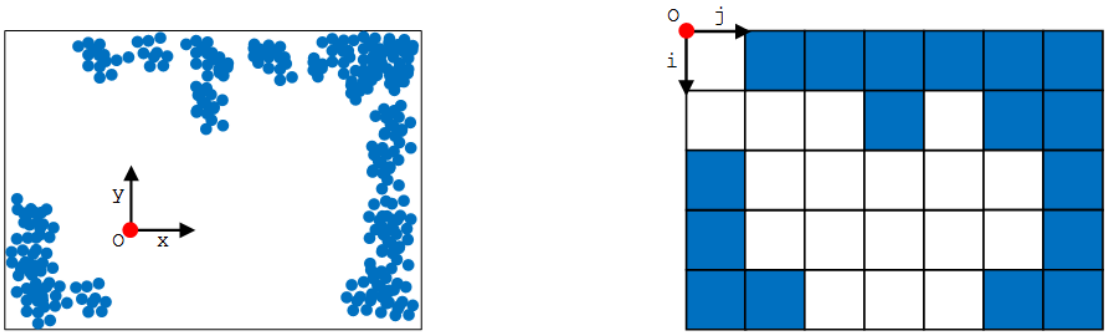


Figura 3. 1 - Esempio di mapping delle posizioni occupate. Si evidenziano i diversi sistemi di riferimento.

Si è scelto di assegnare il valore 1 a tutte quelle celle contenenti almeno un punto rappresentante un ostacolo. Ogni punto del file ".map" è stato quindi tradotto nella corrispondente cella, e questa marcata come occupata. La figura 3.2 mette a confronto la rappresentazione di un dettaglio del file ".map" (a) con la matrice risultante dalla traduzione (b).

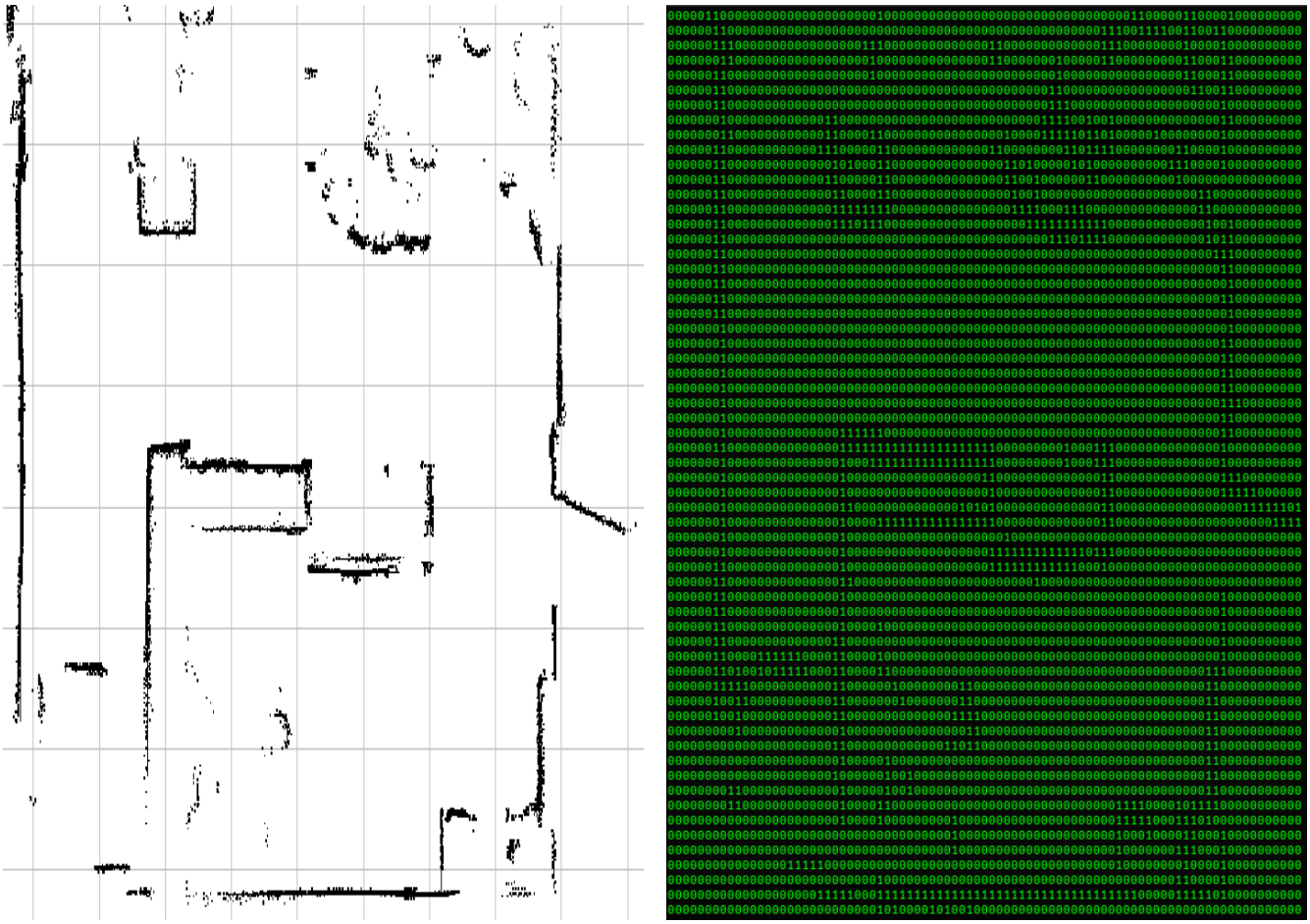


Figura 3. 2 - (a) File ".map" visualizzato con Mapper3; (b) Traduzione in forma matriciale della mappa. In questa rappresentazione una cella della matrice corrisponde ad una superficie di 120mm x 120mm.

Oltre alla traduzione da coordinate della mappa a celle della matrice, è stata ricavata anche la trasformazione inversa, il cui utilizzo si è reso necessario nelle fasi del progetto descritte più avanti. Chiaramente, dato che in fase di discretizzazione si perde di dettaglio (più punti sono mappati in una medesima cella), con l'applicazione della trasformazione inversa non è più possibile recuperare tale informazione. Una cella viene tradotta nella coordinata della mappa corrispondente al centro della cella. Le formule utilizzate per compiere questa traduzione sono le seguenti:

$$x = (1 + j)dimensione\_cella - |coordinata\_\min\_x| \quad (3)$$

$$y = coordinata\_max\_y - (i + 1)dimensione\_cella \quad (4)$$

3.3. Growing

Il calcolo del percorso ottimo avrebbe potuto essere eseguito già sulla matrice risultante dalla precedente elaborazione. Tuttavia sarebbe stato necessario considerare le dimensioni reali del robot che, in generale, può occupare più celle.

È stato invece molto più comodo assimilare il robot ad una cella e calcolare quindi il percorso come successione di singole celle libere. A tale scopo è risultata idonea la tecnica del region growing spiegata a lezione, dove ogni ostacolo è ingrandito di una dimensione proporzionale alla dimensione reale del robot e questo rimpicciolito di conseguenza.

Per applicare questa tecnica si è stabilito innanzi tutto un fattore di ingrandimento pari a

$$fattoreIngrandimento = \left\lceil \frac{dimesione\_reale\_robot}{2 * dimensione\_cella} \right\rceil$$

questo indica il raggio entro il quale le celle attorno ad una occupata sono identificate anch'esse come occupate. Come si può notare, questo è correlato alle dimensioni reali del robot, quindi il comportamento corretto del programma dipende da una precedente misurazione del robot che lo eseguirà.

Per effettuare il growing, si è scelto di utilizzare una matrice diversa da quella di partenza per evitare che la ricorsività della tecnica portasse a creare una matrice completamente occupata.

L'attività di growing ha soddisfatto le precondizioni per il calcolo del percorso ottimo, ovvero la garanzia che il robot possa transitare su una cella considerata libera e che occupi una sola cella alla volta.

La figura numero 3.3 permette di confrontare la matrice rappresentante la mappa originale (a), con il risultato del growing (b). Si può notare come ostacoli vicini risultino un unico elemento: questo è corretto perché indica il fatto che il robot non avrebbe abbastanza spazio per passare tra essi.

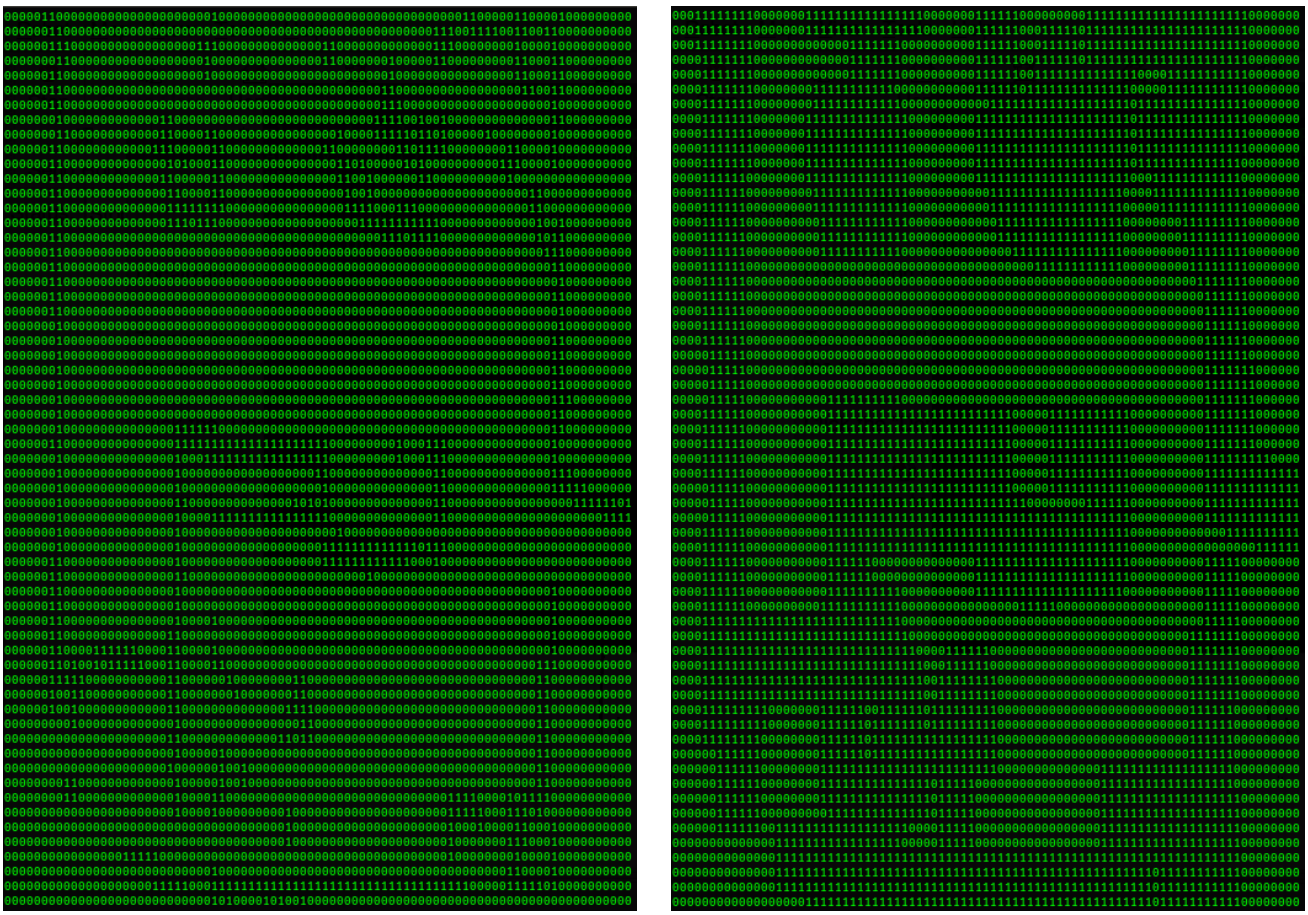


Figura 3. 3 - (a) Matrice rappresentante la mappa; (b) Risultato del growing.

3.4. Pianificazione del percorso ottimo

Per la pianificazione della traiettoria ottima dalla posizione corrente al goal, è stato utilizzato l'algoritmo di ricerca A*, noto dall'intelligenza artificiale.

Tale algoritmo di ricerca best-first è risultato applicabile al problema in quanto le celle della matrice contrassegnate come libere dopo il growing possono essere considerate come i nodi di un grafo i cui archi sono i percorsi che le collegano. A* è stato scelto perché garantisce l'ottimalità della soluzione se applicato con un'euristica ammissibile.

Brevemente si ricorda che un'euristica è ammissibile se

$$h(n) \leq h^*(n) \ \ \forall n$$

dove $h(n)$ è il costo, stimato attraverso l'euristica, del cammino dal nodo n al goal, mentre $h^*(n)$ è il costo effettivo del cammino in esame.

L'euristica adottata, che ha permesso di rispettare questa proprietà e di conseguenza di garantire l'ottimalità del percorso calcolato da A*, è la distanza euclidea. L'euristica applicata in particolare è stata la seguente:

$$h(M_{i,j}) = \left\lfloor \sqrt{(i_{goal} - i)^2 + (j_{goal} - j)^2} \right\rfloor * 10 \ \ \forall M_{i,j}$$

dove con $M_{i,j}$ si intende la cella della matrice risultante dal growing in considerazione, mentre i_{goal} e j_{goal} sono gli indici di riga e colonna della cella contenente il goal.

La funzione di costo $f(n)$ usata dall'algoritmo per selezionare i successivi nodi da espandere è composta da due termini: l'euristica definita precedentemente e il costo $g(n)$ sostenuto per percorrere il cammino dal nodo iniziale a quello considerato.

Questo secondo fattore è la somma dei singoli costi per passare da un nodo al successivo. È stato utilizzato un vicinato di tipo N8 ma, per privilegiare gli spostamenti rettilinei rispetto a quelli obliqui, si è assunto un costo pari a 10 per i primi e 14 per i secondi.

La funzione di costo risulta quindi:

$$f(n) = g(n) + h(n) \ \ \forall n$$

A* per ogni iterazione seleziona il nodo che assume il valore di $f(n)$ minore.

Per comprendere meglio il funzionamento dell'algoritmo, è stato inserito in appendice un esempio numerico.

Nello specifico, l'implementazione di A* all'interno del programma realizzato, prevede i seguenti input:

- La mappa dell'ambiente, risultato del growing;
- Le coordinate del punto di partenza in forma matriciale, tradotte mediante le formule (1) e (2). Il punto di partenza è la posizione in cui si trova il robot nel momento in cui la localizzazione supera la soglia di confidenza prevista;
- Le coordinate del punto di arrivo in forma matriciale, tradotte mediante le formule (1) e (2). Queste coordinate sono fornite in input dall'utente sottoforma di parametri del programma principale.

L'output dell'algoritmo è costituito da:

- Due stack, che contengono le coordinate i e j del percorso ottimo, in forma matriciale;
- Una stringa, in cui ogni carattere rappresenta la cella successiva da raggiungere tra le otto possibili (alto, basso, sinistra, destra e le quattro diagonali).

I valori di output dell'algoritmo sono stati ulteriormente elaborati per ottenere sia una rappresentazione grafica del percorso, sia la traiettoria ottima sotto forma di coordinate per il robot.

La figura 3.4 rappresenta un esempio di percorso ottimo calcolato.
La posizione iniziale è indicata con "S", quella finale con "F", il percorso calcolato con "P", mentre le celle occupate sono contrassegnate con "O".

Figura 3. 4 - Mappa con percorso ottimo calcolato da A*

Come precedentemente affermato, l'algoritmo popola due stack contenenti le coordinate matriciali i e j delle celle del percorso ottimo.

Per permettere l'effettiva navigazione è necessario tradurre questi valori in coordinate cartesiane.
Questo è stato fatto utilizzando le formule (3) e (4). Il procedimento ha portato ad ottenere due nuovi stack contenenti le pose che il robot dovrà visitare, ma in ordine inverso (a causa della struttura dati LIFO utilizzata).
Si è dovuto quindi ricorrere ad un ulteriore ciclo per ripristinare l'ordine corretto delle coordinate.

Si noti che prima di effettuare queste operazioni è stata scartata la prima coppia di coordinate in quanto non rappresentano un obiettivo da raggiungere, ma la posizione già occupata dal robot.

3.5. Navigazione lungo il percorso ottimo

Una volta che la traiettoria ottima è disponibile al robot nei due stack separati, essa deve essere percorsa.

Come già spiegato nel paragrafo precedente, i due stack contengono le coordinate esatte del centro della prossima cella che il robot dovrà raggiungere. In questo modo, passo dopo passo, esso giungerà alla cella finale, rappresentante il goal.

È stato quindi naturale pensare di creare un ActionGroup "Naviga" al quale si è aggiunta l'azione "ArActionGoTo" resa disponibile da ARIA.

Questa azione comanda al robot di navigare verso una posa indicata fermandolo quando è arrivato ad una certa distanza dalla posa ("closeDist"); è possibile indicare un nuovo traguardo intermedio dinamicamente attraverso la funzione setGoal() e, con la funzione haveAchievedGoal(), capire quando l'obiettivo è stato raggiunto.

Questa soluzione è stata implementata fin da subito ma ha presentato principalmente due problemi:

- Una volta che il robot ha finito la prima procedura di localizzazione passa le coordinate all'algoritmo A* che calcola il percorso ottimo. Come precedentemente detto, la posa iniziale viene discretizzata introducendo un errore; questo, unito al fatto che il robot è anolonomo e quindi non riesce a raggiungere immediatamente la posa indicata, ha evidenziato un problema nel controllo con la funzione haveAchievedGoal().
  Infatti si è sperimentato che, settando una tolleranza per il controllo con un valore troppo basso, il robot iniziava a ruotare attorno alla posa goal indicata, urtando eventuali ostacoli e rendendo così impossibile la prosecuzione dell'algoritmo. Quando invece la prima posa veniva raggiunta, il controllo non presentava più alcun problema e il programma proseguiva fino al suo completamento.

- La seconda problematica deriva dal fatto che nel ciclo di navigazione al robot è data la posa del centro di ogni cella della traiettoria ottima ed esso aziona i propri motori per arrivarci, finché non è valida la condizione di raggiungimento dell'obiettivo (haveAchievedGoal()).
  L'effetto di questo continuo controllo è un movimento "a scatti" in cui il robot si ferma ad ogni passo del percorso. Il problema risulta ancora più evidente utilizzando una elevata granularità (dimensione della cella piccola).

Questi due problemi sono stati risolti attraverso due strategie diverse: la prima attuata in modo dinamico, mentre la seconda in un unico momento, dopo la pianificazione del percorso.

Il primo problema è stato risolto suddividendo il percorso in 3 parti alla quali è stata assegnata una diversa tolleranza: per il primo 30% è stata impostata una tolleranza pari a 4 volte la dimensione della cella, per il successivo 30% di 3 volte, per il percorso restante di 2. Il risultato è stato quello di ottenere una sorta di "imbuto" che dirige il robot gradualmente verso il percorso ottimale. L'applicazione di questa soluzione si può vedere in figura 3.5.

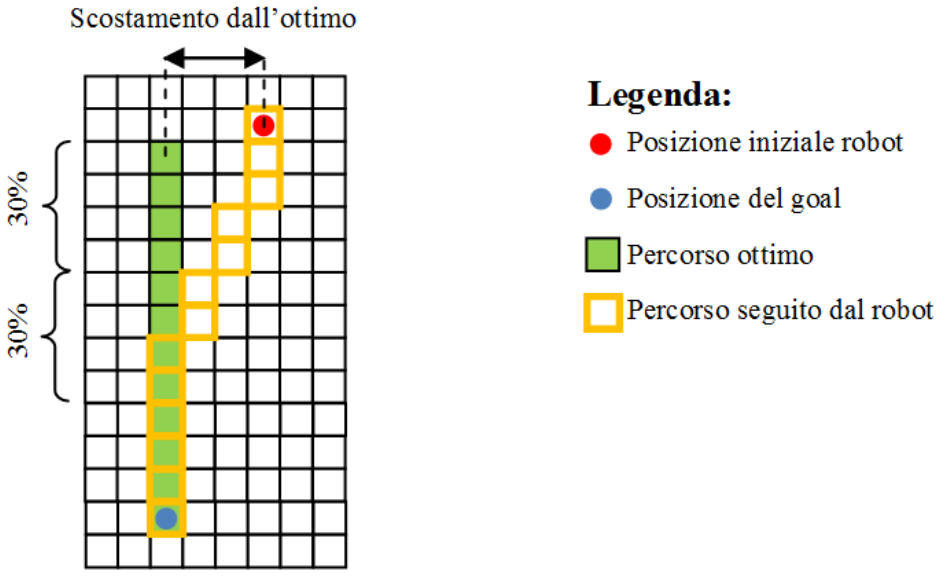


Figura 3. 5 - Avvicinamento graduale al percorso ottimo mediante tolleranza variabile

Il secondo problema invece è stato attenuato attraverso la strategia visibile in figura 3.6: nei tratti rettilinei del percorso ottimo è inutile far muovere il robot di cella in cella. È risultato più vantaggioso indicargli solamente le celle iniziale e finale di ogni segmento ed evitare così tutti quei controlli di posizione intermedi che avrebbe dovuto effettuare altrimenti.

Questa ottimizzazione è facoltativa: nella fase di chiamata del programma è possibile indicare attraverso il parametro booleano `-rett` se eseguirla o meno.

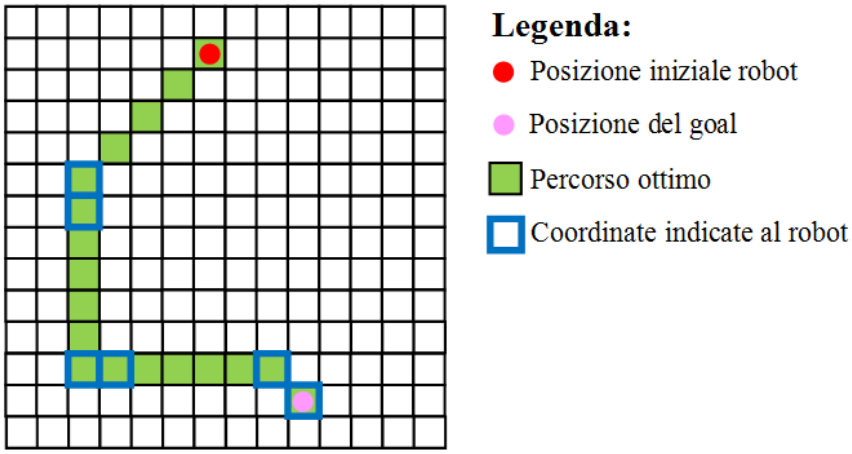


Figura 3. 6 - Ottimizzazione dei tratti rettilinei

4. Modalità operative

In questo capitolo si descrive nel dettaglio tutto ciò che è necessario per il corretto funzionamento del programma ed il modo in cui deve essere utilizzato.

4.1. Componenti necessari

Il progetto è stato realizzato e testato utilizzando Tobor, un robot Pioneer1 della MobileRobots.
Nulla vieta che il programma possa essere eseguito su un qualsiasi robot in grado interpretare ed eseguire i comandi di ARIA e ARNL.
È infatti necessario che queste due librerie siano installate sul calcolatore a bordo del robot. Se non lo fossero, esse sono reperibili sul sito della MobileRobots.

Per interfacciarsi al calcolatore di bordo è richiesto un ulteriore pc da utilizzare come client dotato preferibilmente di sistema operativo Linux.

Per la fase di creazione della mappa dell'ambiente sono necessari i seguenti software: MobileEyes e Mapper3 (non in versione "Basic") da installare sul calcolatore client e arnlServer da eseguire sul computer del robot.
Occorre inoltre che il robot disponga di un laser URG, utile anche per la fase di localizzazione.

Il file della mappa creata dovrà essere posto nella stessa cartella contenente il software sviluppato nell'ambito di questo progetto.

4.2. Modalità di installazione

Si suppone che il calcolatore a bordo del robot abbia già installate le librerie ARIA e ARNL.

Per quanto riguarda il client si dovranno installare i software MobileEyes e Mapper3.
Dopo esserseli procurati dal sito della MobileRobots o dal sito del laboratorio di robotica avanzata, installarli con il seguente comando:

```
$ sudo dpkg –i <nomePacchetto>
```

Nel caso in cui l'architettura del calcolatore client sia a 64bit, può essere necessario aggiungere al comando precedente l'opzione `––forceArchitecture`.

Per l'esecuzione del programma realizzato non è necessaria alcuna installazione: è sufficiente trasferirlo sul calcolatore del robot insieme alla mappa dell'ambiente.

4.3. Modalità di taratura

Il software realizzato necessita di alcuni parametri obbligatori per funzionare. Quattro di questi devono essere indicati per ultimi e servono a tararlo per le esigenze dell'ambiente.

Le coordinate del punto di destinazione vengono indicate con `–xDest` e `–yDest`; l'utente dovrà aver cura che il punto indicato sia all'interno della mappa e non corrisponda ad un ostacolo. In caso contrario il software terminerà la sua esecuzione indicando che non è stato trovato alcun percorso.
Si consiglia di utilizzare il software Mapper3 per rilevare le coordinate del punto desiderato sulla mappa passata come parametro.

Il parametro `–cella` imposta la dimensione della porzione di spazio reale in millimetri che una singola cella dovrà rappresentare.
Tale valore deve essere scelto accuratamente in quanto a celle grandi corrisponde una velocità di navigazione maggiore, ma una peggiore modellizzazione dell'ambiente che potrebbe ridurre sensibilmente gli spazi considerati sgombri da ostacoli. Al contrario a celle piccole corrisponde un ottima aderenza della mappa discretizzata all'ambiente reale, ma si riscontra un aumento del tempo di calcolo del percorso ottimo ed una navigazione più lenta.
Per i test realizzati è stato assegnato un valore di 40 a questo parametro.

Infine il parametro `–rett` indica la volontà di eseguire o meno l'ottimizzazione dei tratti rettilinei del percorso (vedi paragrafo 3.5). Il valore 1 la abilita, lo 0 la disabilita.

4.4. Avvertenze

Di seguito si descriveranno alcuni accorgimenti per testare in modo corretto il software nel laboratorio di robotica ARL sul robot Tobor.

Si suppone che la mappa del laboratorio (nominata "mappaARL.map") sia già stata creata. Per la procedura di creazione si rimanda alla lettura del riferimento bibliografico [2].

Innanzitutto è necessario verificare che il robot, i suoi motori ed il calcolatore di bordo siano accesi ed operativi.

Dal terminale del pc client collegarsi via ssh al calcolatore di Tobor:

```
$ ssh user@192.0.2.21
```

Quando richiesto inserire la password richiedendola a chi di competenza.

Successivamente, una volta acceduti alla cartella contenente programma e mappa, eseguire il software digitando il comando:

```
$ ./NavigazioneOttima –rp /dev/ttyUSB0 –laserType urg –lp /dev/ttyACM0
–map mappaARL.map –xDest 3000 –yDest -1000 –rett 1 –cella 40
```

Esso permette la navigazione verso il goal posizionato in (3000,-1000) sulla mappa "mappaARL.map" discretizzata in celle di 40mm, ottimizzando i percorsi rettilinei.

Utilizzando un altro terminale aprire MobileEyes, inserire i dati richiesti (host: 192.0.2.21, username: user, password: la stessa del ssh) e "aiutare" il robot nella localizzazione attraverso il comando "Localize to Point".

Questo è utile perché la localizzazione autonoma è molto dispendiosa in termini di tempo.

5. Conclusioni e sviluppi futuri

L'elaborato ha raggiunto gli obiettivi prefissati di creazione della mappa, localizzazione del robot all'interno di essa, pianificazione di un percorso ottimale verso un punto desiderato e conseguente raggiungimento dello stesso.

Lo sviluppo del programma ha permesso di applicare le conoscenze teoriche apprese a lezione. Tra i principali concetti applicati si ricordano per la parte concettuale la discretizzazione dell'ambiente in celle occupate e libere, il region growing nonché la tecnica dell'algoritmo A* studiata nel corso di intelligenza artificiale. Aspetti più pratici approfonditi sono stati l'utilizzo delle action applicate agli actionGroup (in particolare il "wander" e la successiva navigazione implementata), l'utilizzo delle librerie per robot mobili, i tool di supporto come MobileEyes ed i limiti dei diversi tipi di sensore.

Aver seguito il corso ha aiutato a discriminare tra le possibili vie implementative, scegliendo le più funzionali per gli obiettivi posti. Ad esempio si è preferito far ricorso al metodo ArActionGoTo per passare da una cella alla successiva lungo il percorso ottimo invece che, nota la dimensione della stessa, dare al robot una successione di movimenti di lunghezza prestabilita. Questo perché la seconda scelta avrebbe portato all'accumulo di errori tipico dei robot mobili.

Le sessioni di laboratorio hanno confermato quanto detto a lezione sullo scostamento tra comportamento simulato e realtà. Spesso infatti ci si è trovati in situazioni dove comportamenti corretti a livello teorico non trovavano riscontro nell'applicazione pratica.

I possibili sviluppi futuri riguardano l'estensione del funzionamento del software in ambienti dinamici. Può essere infatti interessante che il robot possa individuare nuovi ostacoli durante la navigazione lungo il percorso ottimo e ricalcolare questo alla luce delle nuove rilevazioni. Il focus del progetto non è stato posto sul meccanismo di localizzazione (per il quale è stato utilizzato l'algoritmo implementato in ARNL), tuttavia può essere uno spunto l'ottimizzazione di questo per rendere più veloce ed efficace questa fase iniziale.

Bibliografia

[1] Russel, S., Norvig, P.: "Artificial Intelligence: A Modern Approach (3rd Edition)", Prentice Hall, dicembre 2009.

[2] Cerutti, B., Ducoli, F., Lombardi, L., Rizzardi, E.: "Creazione nuove mappe del laboratorio con Tobor", febbraio 2012. http://www.ing.unibs.it/%7Earl/docs/projects/Nav_26.pdf

[3] ARIA Developer's Reference Manual: http://www.ing.unibs.it/~arl/docs/documentation/Aria%20documentation/Current/

[4] ARNL Developer's Reference Manual: http://www.ing.unibs.it/~arl/docs/documentation/Aria%20documentation/Current/BaseArnl-Reference/

Appendice A:

Di seguito è esposto un esempio di esecuzione dell'algoritmo A* per la ricerca del goal G e del percorso ottimo dal punto di partenza S. Si considera la mappa riportata in figura A.1, in cui le celle nere rappresentano ostacoli. I numeri nelle celle libere rappresentano i valori dell'euristica h, che calcola la distanza euclidea tra la cella in esame e il goal, ne considera la parte intera inferiore e moltiplica per 10 (è l'euristica usata nel progetto, già esposta nel paragrafo 3.4). Anche il costo del cammino da una cella alla successiva è lo stesso usato nel progetto, ovvero 10 per gli spostamenti verticali e orizzontali e 14 per gli spostamenti in diagonale. Il valore di f è calcolato come la somma di g (costo sostenuto nel cammino dalla cella S alla cella corrente) e h (l'euristica).

La tabella mostra per ogni passo dell'algoritmo i nodi (celle) già esplorati e quelli nella "frontiera", cioè i prossimi nodi da esplorare. Per ogni nodo si tiene traccia sia del valore di f = g + h di quando è stato inserito nella frontiera, sia del nodo che lo precede nel percorso (indicato con "predec" nella tabella). I nodi esplorati sono visualizzati in verde, mentre quelli della frontiera sono in viola.

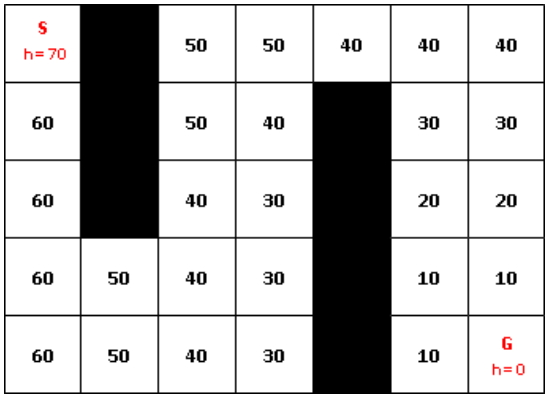
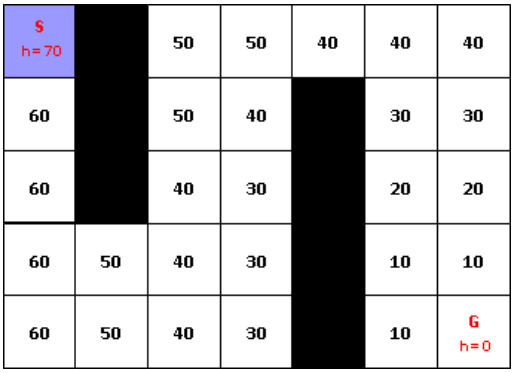
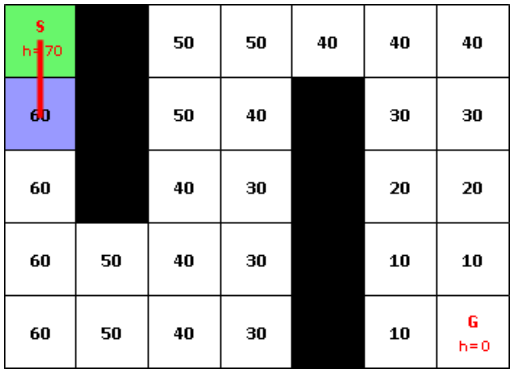
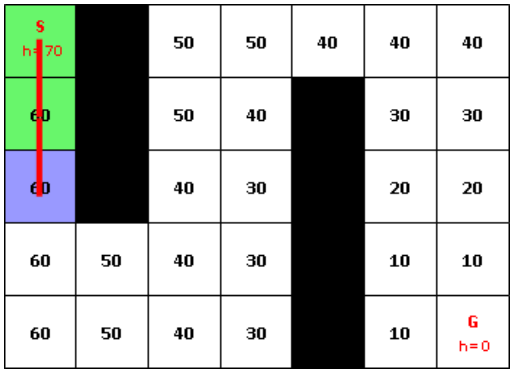
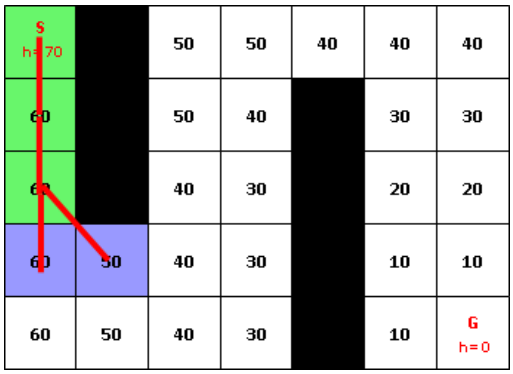
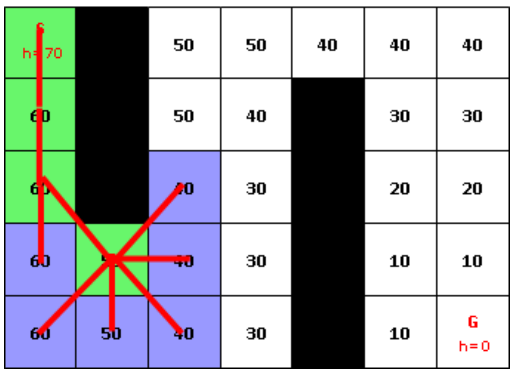
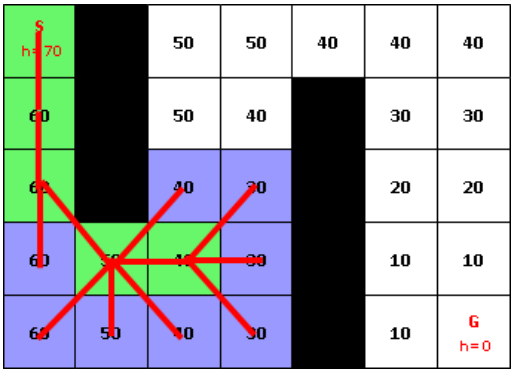
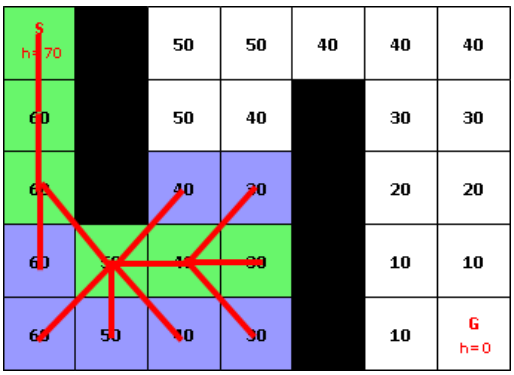
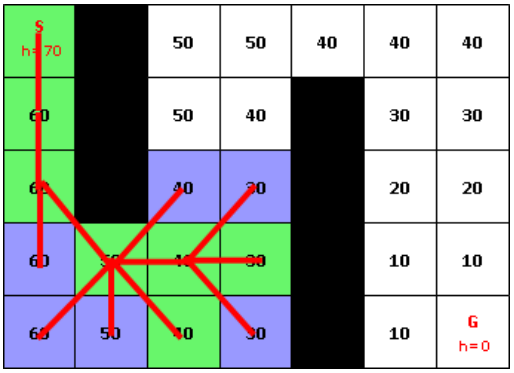
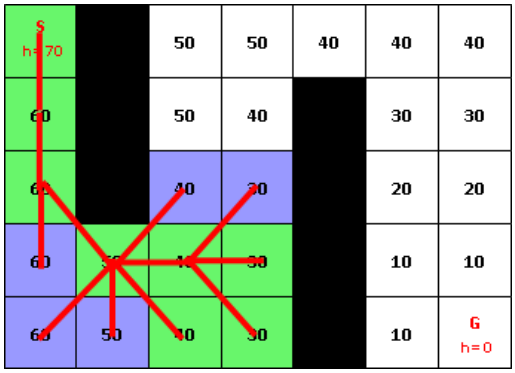
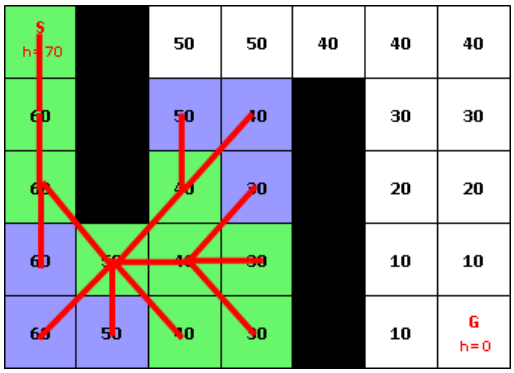
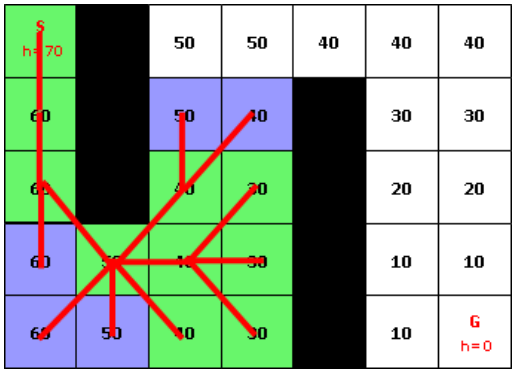


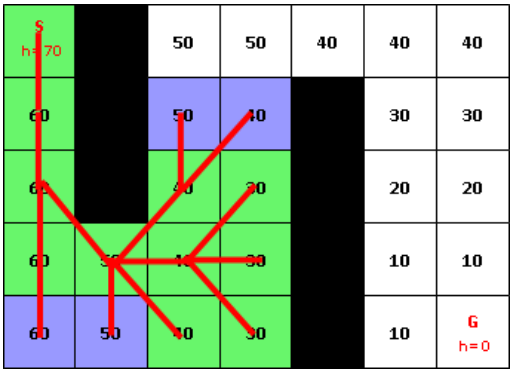
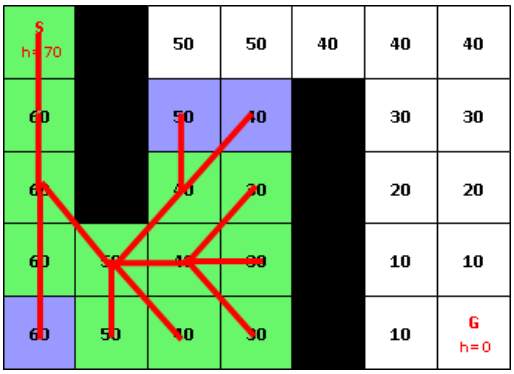
Figura A.1 - Mappa usata nell'esempio. I numeri sono i valori dell'euristica

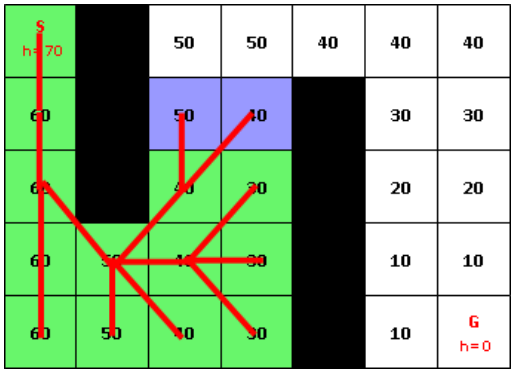
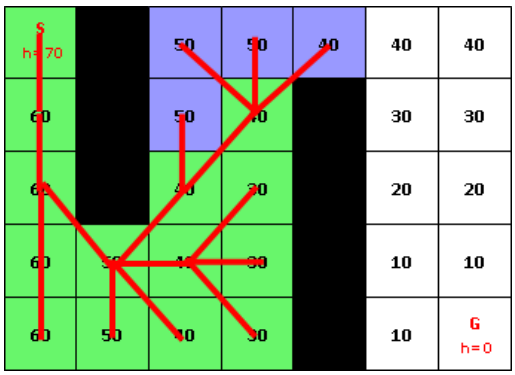
FRONTIERA	NODI ESPLORATI	SPIEGAZIONE	VISUALIZZAZIONE
M0,0 - f = 70; predec = no	vuoto		

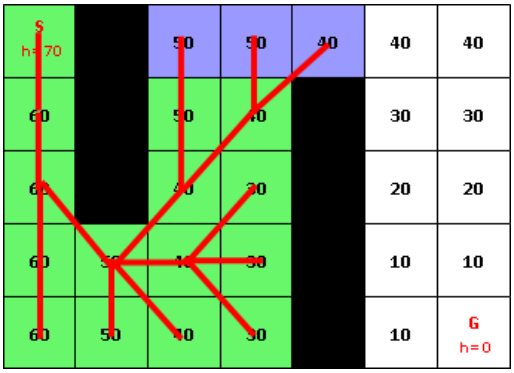
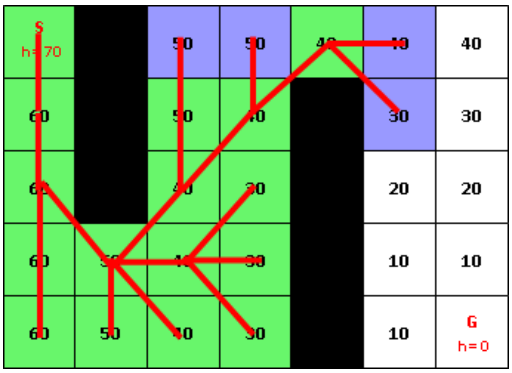
M1,0 - f = 70; predec = M0,0	M0,0 - f = 70; predec = no	Espando il nodo iniziale, metto l'unico successore nella frontiera, con f = g + h = 10 + 60	
M2,0 - f = 80; predec = M1,0	M0,0 - f = 70; predec = no M1,0 - f = 70; predec = M0,0	Espando il nodo M1,0. I successori sarebbero M0,0 e M2,0, ma il primo è già nei nodi espansi => non lo rimetto nella frontiera	
M3,0 - f = 90; predec = M2,0 M3,1 - f = 84; predec = M2,0	M0,0 - f = 70; predec = no M1,0 - f = 70; predec = M0,0 M2,0 - f = 80; predec = M1,0	Espando il nodo M2,0, questa volta aggiungo i due successori alla frontiera	
M3,0 - f = 90; predec = M2,0 M4,0 - f = 108; predec = M3,1 M4,1 - f = 94; predec = M3,1 M4,2 - f = 88; predec = M3,1 M3,2 - f = 84; predec = M3,1 M2,2 - f = 88; predec = M3,1	M0,0 - f = 70; predec = no M1,0 - f = 70; predec = M0,0 M2,0 - f = 80; predec = M1,0 M3,1 - f = 84; predec = M2,0	Espando il nodo M3,1. Dei suoi successori, non aggiungo alla frontiera M2,0 perché è già tra i nodi esplorati. Non sostituisco M3,0 perché nel nuovo percorso avrei un costo più elevato (f = 104)	

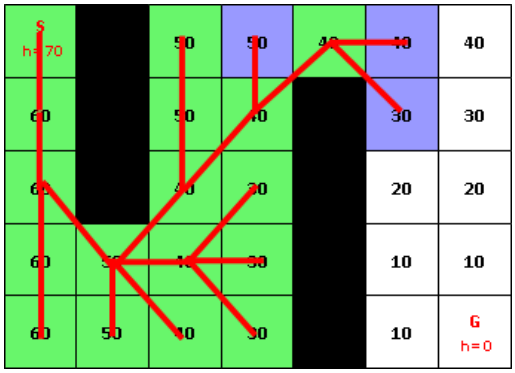
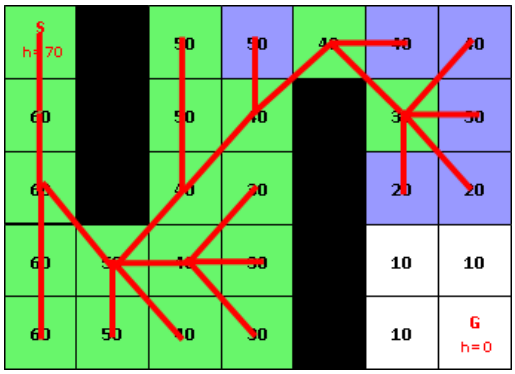
M3,0 - f = 90; predec = M2,0 M4,0 - f = 108; predec = M3,1 M4,1 - f = 94; predec = M3,1 M4,2 - f = 88; predec = M3,1 M2,2 - f = 88; predec = M3,1 M4,3 - f = 88; predec = M3,2 M3,3 - f = 84; predec = M3,2 M2,3 - f = 88; predec = M3,2	M0,0 - f = 70; predec = no M1,0 - f = 70; predec = M0,0 M2,0 - f = 80; predec = M1,0 M3,1 - f = 84; predec = M2,0 M3,2 - f = 84; predec = M3,1	Espando M3,2, quello con il valore di f minore. Non aggiungo M3,1 (già esplorato) Non sostituisco M4,1, M4,2 e M2,2 (già in frontiera con costo più basso)	
M3,0 - f = 90; predec = M2,0 M4,0 - f = 108; predec = M3,1 M4,1 - f = 94; predec = M3,1 M4,2 - f = 88; predec = M3,1 M2,2 - f = 88; predec = M3,1 M4,3 - f = 88; predec = M3,2 M2,3 - f = 88; predec = M3,2	M0,0 - f = 70; predec = no M1,0 - f = 70; predec = M0,0 M2,0 - f = 80; predec = M1,0 M3,1 - f = 84; predec = M2,0 M3,2 - f = 84; predec = M3,1 M3,3 - f = 84; predec = M3,2	Espando il nodo 3,3 ma non sostituisco nulla nella frontiera, perché tutti i successori sono già presenti con un costo inferiore.	
M3,0 - f = 90; predec = M2,0 M4,0 - f = 108; predec = M3,1 M4,1 - f = 94; predec = M3,1 M2,2 - f = 88; predec = M3,1 M4,3 - f = 88; predec = M3,2 M2,3 - f = 88; predec = M3,2	M0,0 - f = 70; predec = no M1,0 - f = 70; predec = M0,0 M2,0 - f = 80; predec = M1,0 M3,1 - f = 84; predec = M2,0 M3,2 - f = 84; predec = M3,1 M3,3 - f = 84; predec = M3,2 M4,2 - f = 88; predec = M3,1	Espando uno dei nodi con costo 88, ad esempio M4,2 N.B. la f = g + h tengo sempre quella di quando ho inserito il nodo nella frontiera. (Ciò mi permetterà di ricostruire il path) Tutti i successori sono già espansi o sono nella frontiera con costo minore => non aggiungo nulla	

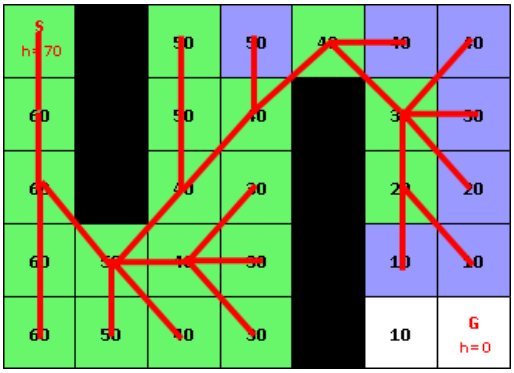
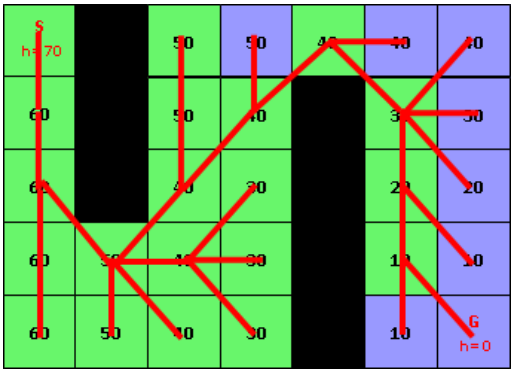
M3,0 - f = 90; predec = M2,0 M4,0 - f = 108; predec = M3,1 M4,1 - f = 94; predec = M3,1 M2,2 - f = 88; predec = M3,1 M2,3 - f = 88; predec = M3,2	M0,0 - f = 70; predec = no M1,0 - f = 70; predec = M0,0 M2,0 - f = 80; predec = M1,0 M3,1 - f = 84; predec = M2,0 M3,2 - f = 84; predec = M3,1 M3,3 - f = 84; predec = M3,2 M4,2 - f = 88; predec = M3,1 M4,3 - f = 88; predec = M3,2	Espando uno dei nodi con costo 88, ad esempio M4,3 Tutti i successori sono già espansi o sono nella frontiera con costo minore => non aggiungo nulla	
M3,0 - f = 90; predec = M2,0 M4,0 - f = 108; predec = M3,1 M4,1 - f = 94; predec = M3,1 M2,3 - f = 88; predec = M3,2 M1,2 - f = 108; predec = M2,2 M1,3 - f = 102; predec = M2,2	M0,0 - f = 70; predec = no M1,0 - f = 70; predec = M0,0 M2,0 - f = 80; predec = M1,0 M3,1 - f = 84; predec = M2,0 M3,2 - f = 84; predec = M3,1 M3,3 - f = 84; predec = M3,2 M4,2 - f = 88; predec = M3,1 M4,3 - f = 88; predec = M3,2 M2,2 - f = 88; predec = M3,1	Espando uno dei nodi con costo 88, ad esempio M2,2. Aggiungo alla frontiera i nodi M1,2 e M1,3	
M3,0 - f = 90; predec = M2,0 M4,0 - f = 108; predec = M3,1 M4,1 - f = 94; predec = M3,1 M1,2 - f = 108; predec = M2,2 M1,3 - f = 102; predec = M2,2	M0,0 - f = 70; predec = no M1,0 - f = 70; predec = M0,0 M2,0 - f = 80; predec = M1,0 M3,1 - f = 84; predec = M2,0 M3,2 - f = 84; predec = M3,1 M3,3 - f = 84; predec = M3,2 M4,2 - f = 88; predec = M3,1 M4,3 - f = 88; predec = M3,2 M2,2 - f = 88; predec = M3,1 M2,3 - f = 88; predec = M3,2	Espando M2,3. Non aggiungo nulla.	

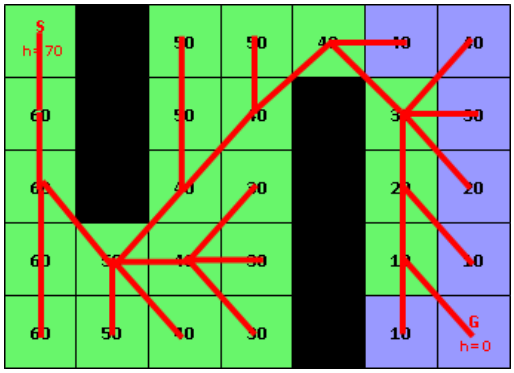
M4,0 - f = **100**; predec = **M2,0** M4,1 - f = 94; predec = M3,1 M1,2 - f = 108; predec = M2,2 M1,3 - f = 102; predec = M2,2	M0,0 - f = 70; predec = no M1,0 - f = 70; predec = M0,0 M2,0 - f = 80; predec = M1,0 M3,1 - f = 84; predec = M2,0 M3,2 - f = 84; predec = M3,1 M3,3 - f = 84; predec = M3,2 M4,2 - f = 88; predec = M3,1 M4,3 - f = 88; predec = M3,2 M2,2 - f = 88; predec = M3,1 M2,3 - f = 88; predec = M3,2 M3,0 - f = 90; predec = M2,0	Espando M3,0. Sostituisco M4,0.	
M4,0 - f = 100; predec = M2,0 M1,2 - f = 108; predec = M2,2 M1,3 - f = 102; predec = M2,2	M0,0 - f = 70; predec = no M1,0 - f = 70; predec = M0,0 M2,0 - f = 80; predec = M1,0 M3,1 - f = 84; predec = M2,0 M3,2 - f = 84; predec = M3,1 M3,3 - f = 84; predec = M3,2 M4,2 - f = 88; predec = M3,1 M4,3 - f = 88; predec = M3,2 M2,2 - f = 88; predec = M3,1 M2,3 - f = 88; predec = M3,2 M3,0 - f = 90; predec = M2,0 M4,1 - f = 94; predec = M3,1	Espando M4,1. Non aggiungo nulla.	

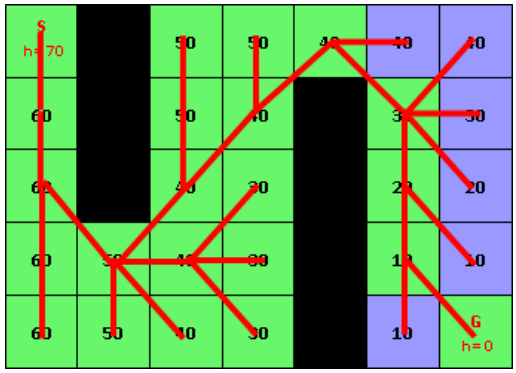
M1,2 - f = 108; predec = M2,2 M1,3 - f = 102; predec = M2,2	M0,0 - f = 70; predec = no M1,0 - f = 70; predec = M0,0 M2,0 - f = 80; predec = M1,0 M3,1 - f = 84; predec = M2,0 M3,2 - f = 84; predec = M3,1 M3,3 - f = 84; predec = M3,2 M4,2 - f = 88; predec = M3,1 M4,3 - f = 88; predec = M3,2 M2,2 - f = 88; predec = M3,1 M2,3 - f = 88; predec = M3,2 M3,0 - f = 90; predec = M2,0 M4,1 - f = 94; predec = M3,1 M4,0 - f = 100; predec = M2,0	Espando M4,0. Non aggiungo nulla.	
M1,2 - f = 108; predec = M2,2 M0,2 - f = 126; predec = M1,3 M0,3 - f = 122; predec = M1,3 M0,4 - f = 116; predec = M1,3	M0,0 - f = 70; predec = no M1,0 - f = 70; predec = M0,0 M2,0 - f = 80; predec = M1,0 M3,1 - f = 84; predec = M2,0 M3,2 - f = 84; predec = M3,1 M3,3 - f = 84; predec = M3,2 M4,2 - f = 88; predec = M3,1 M4,3 - f = 88; predec = M3,2 M2,2 - f = 88; predec = M3,1 M2,3 - f = 88; predec = M3,2 M3,0 - f = 90; predec = M2,0 M4,1 - f = 94; predec = M3,1 M4,0 - f = 100; predec = M2,0 M1,3 - f = 102; predec = M2,2	Espando M1,3. Aggiungo M0,2, M0,3 e M0,4	

M0,2 - f = **118**; predec = **M1,2** M0,3 - f = 122; predec = M1,3 M0,4 - f = 116; predec = M1,3	M0,0 - f = 70; predec = no M1,0 - f = 70; predec = M0,0 M2,0 - f = 80; predec = M1,0 M3,1 - f = 84; predec = M2,0 M3,2 - f = 84; predec = M3,1 M3,3 - f = 84; predec = M3,2 M4,2 - f = 88; predec = M3,1 M4,3 - f = 88; predec = M3,2 M2,2 - f = 88; predec = M3,1 M2,3 - f = 88; predec = M3,2 M3,0 - f = 90; predec = M2,0 M4,1 - f = 94; predec = M3,1 M4,0 - f = 100; predec = M2,0 M1,3 - f = 102; predec = M2,2 M1,2 - f = 108; predec = M2,2	Espando M1,2. Sostituisco M0,2	
M0,2 - f = 118; predec = M1,2 M0,3 - f = 122; predec = M1,3 M0,5 - f = 126; predec = M0,4 M1,5 - f = 120; predec = M0,4	M0,0 - f = 70; predec = no M1,0 - f = 70; predec = M0,0 M2,0 - f = 80; predec = M1,0 M3,1 - f = 84; predec = M2,0 M3,2 - f = 84; predec = M3,1 M3,3 - f = 84; predec = M3,2 M4,2 - f = 88; predec = M3,1 M4,3 - f = 88; predec = M3,2 M2,2 - f = 88; predec = M3,1 M2,3 - f = 88; predec = M3,2 M3,0 - f = 90; predec = M2,0 M4,1 - f = 94; predec = M3,1 M4,0 - f = 100; predec = M2,0 M1,3 - f = 102; predec = M2,2 M1,2 - f = 108; predec = M2,2 M0,4 - f = 116; predec = M1,3	Espando M0,4. Aggiungo M0,5 e M1,5	

M0,3 - f = 122; predec = M1,3 M0,5 - f = 126; predec = M0,4 M1,5 - f = 120; predec = M0,4	M0,0 - f = 70; predec = no M1,0 - f = 70; predec = M0,0 M2,0 - f = 80; predec = M1,0 M3,1 - f = 84; predec = M2,0 M3,2 - f = 84; predec = M3,1 M3,3 - f = 84; predec = M3,2 M4,2 - f = 88; predec = M3,1 M4,3 - f = 88; predec = M3,2 M2,2 - f = 88; predec = M3,1 M2,3 - f = 88; predec = M3,2 M3,0 - f = 90; predec = M2,0 M4,1 - f = 94; predec = M3,1 M4,0 - f = 100; predec = M2,0 M1,3 - f = 102; predec = M2,2 M1,2 - f = 108; predec = M2,2 M0,4 - f = 116; predec = M1,3 M0,2 - f = 118; predec = M1,2	Espando M0,2. Non aggiungo nulla.	
M0,3 - f = 122; predec = M1,3 M0,5 - f = 126; predec = M0,4 M0,6 - f = 154; predec = M1,5 M1,6 - f = 130; predec = M1,5 M2,5 - f = 120; predec = M1,5 M2,6 - f = 124; predec = M1,5	M0,0 - f = 70; predec = no M1,0 - f = 70; predec = M0,0 M2,0 - f = 80; predec = M1,0 M3,1 - f = 84; predec = M2,0 M3,2 - f = 84; predec = M3,1 M3,3 - f = 84; predec = M3,2 M4,2 - f = 88; predec = M3,1 M4,3 - f = 88; predec = M3,2 M2,2 - f = 88; predec = M3,1 M2,3 - f = 88; predec = M3,2 M3,0 - f = 90; predec = M2,0 M4,1 - f = 94; predec = M3,1 M4,0 - f = 100; predec = M2,0 M1,3 - f = 102; predec = M2,2 M1,2 - f = 108; predec = M2,2 M0,4 - f = 116; predec = M1,3 M0,2 - f = 118; predec = M1,2 M1,5 - f = 120; predec = M0,4	Espando M1,5. Aggiungo M0,6, M1,6, M2,5 e M2,6.	

M0,3 - f = 122; predec = M1,3 M0,5 - f = 126; predec = M0,4 M0,6 - f = 154; predec = M1,5 M1,6 - f = 130; predec = M1,5 M2,6 - f = 124; predec = M1,5 M3,5 - f = 120; predec = M2,5 M3,6 - f = 124; predec = M2,5	M0,0 - f = 70; predec = no M1,0 - f = 70; predec = M0,0 M2,0 - f = 80; predec = M1,0 M3,1 - f = 84; predec = M2,0 M3,2 - f = 84; predec = M3,1 M3,3 - f = 84; predec = M3,2 M4,2 - f = 88; predec = M3,1 M4,3 - f = 88; predec = M3,2 M2,2 - f = 88; predec = M3,1 M2,3 - f = 88; predec = M3,2 M3,0 - f = 90; predec = M2,0 M4,1 - f = 94; predec = M3,1 M4,0 - f = 100; predec = M2,0 M1,3 - f = 102; predec = M2,2 M1,2 - f = 108; predec = M2,2 M0,4 - f = 116; predec = M1,3 M0,2 - f = 118; predec = M1,2 M1,5 - f = 120; predec = M0,4 M2,5 - f = 120; predec = M1,5	Espando M2,5. Aggiungo M3,5 e M3,6.	
M0,3 - f = 122; predec = M1,3 M0,5 - f = 126; predec = M0,4 M0,6 - f = 154; predec = M1,5 M1,6 - f = 130; predec = M1,5 M2,6 - f = 124; predec = M1,5 M3,6 - f = 124; predec = M2,5 M4,5 - f = 130; predec = M3,5 M4,6 - f = 124; predec = M3,5	M0,0 - f = 70; predec = no M1,0 - f = 70; predec = M0,0 M2,0 - f = 80; predec = M1,0 M3,1 - f = 84; predec = M2,0 M3,2 - f = 84; predec = M3,1 M3,3 - f = 84; predec = M3,2 M4,2 - f = 88; predec = M3,1 M4,3 - f = 88; predec = M3,2 M2,2 - f = 88; predec = M3,1 M2,3 - f = 88; predec = M3,2 M3,0 - f = 90; predec = M2,0 M4,1 - f = 94; predec = M3,1 M4,0 - f = 100; predec = M2,0 M1,3 - f = 102; predec = M2,2 M1,2 - f = 108; predec = M2,2 M0,4 - f = 116; predec = M1,3 M0,2 - f = 118; predec = M1,2 M1,5 - f = 120; predec = M0,4 M2,5 - f = 120; predec = M1,5 M3,5 - f = 120; predec = M2,5	Espando M3,5. Aggiungo M4,5 e M4,6.	

M0,5 - f = 126; predec = M0,4 M0,6 - f = 154; predec = M1,5 M1,6 - f = 130; predec = M1,5 M2,6 - f = 124; predec = M1,5 M3,6 - f = 124; predec = M2,5 M4,5 - f = 130; predec = M3,5 M4,6 - f = 124; predec = M3,5	M0,0 - f = 70; predec = no M1,0 - f = 70; predec = M0,0 M2,0 - f = 80; predec = M1,0 M3,1 - f = 84; predec = M2,0 M3,2 - f = 84; predec = M3,1 M3,3 - f = 84; predec = M3,2 M4,2 - f = 88; predec = M3,1 M4,3 - f = 88; predec = M3,2 M2,2 - f = 88; predec = M3,1 M2,3 - f = 88; predec = M3,2 M3,0 - f = 90; predec = M2,0 M4,1 - f = 94; predec = M3,1 M4,0 - f = 100; predec = M2,0 M1,3 - f = 102; predec = M2,2 M1,2 - f = 108; predec = M2,2 M0,4 - f = 116; predec = M1,3 M0,2 - f = 118; predec = M1,2 M1,5 - f = 120; predec = M0,4 M2,5 - f = 120; predec = M1,5 M3,5 - f = 120; predec = M2,5 M0,3 - f = 122; predec = M1,3	Espando M0,3. Non aggiungo nulla.	
		Qui potrebbero esserci due passaggi in cui espando M2,6 e M3,6 senza aggiungere nulla, dipende da cosa scelgo di espandere prima Qui è mostrato il caso in cui espando prima il nodo M4,6, che è il goal (vedi sotto)	

M0,5 - f = 126; predec = M0,4 M0,6 - f = 154; predec = M1,5 M1,6 - f = 130; predec = M1,5 M2,6 - f = 124; predec = M1,5 M3,6 - f = 124; predec = M2,5 M4,5 - f = 130; predec = M3,5	M0,0 - f = 70; predec = no M1,0 - f = 70; predec = M0,0 M2,0 - f = 80; predec = M1,0 M3,1 - f = 84; predec = M2,0 M3,2 - f = 84; predec = M3,1 M3,3 - f = 84; predec = M3,2 M4,2 - f = 88; predec = M3,1 M4,3 - f = 88; predec = M3,2 M2,2 - f = 88; predec = M3,1 M2,3 - f = 88; predec = M3,2 M3,0 - f = 90; predec = M2,0 M4,1 - f = 94; predec = M3,1 M4,0 - f = 100; predec = M2,0 M1,3 - f = 102; predec = M2,2 M1,2 - f = 108; predec = M2,2 M0,4 - f = 116; predec = M1,3 M0,2 - f = 118; predec = M1,2 M1,5 - f = 120; predec = M0,4 M2,5 - f = 120; predec = M1,5 M3,5 - f = 120; predec = M2,5 M0,3 - f = 122; predec = M1,3 M4,6 - f = 124; predec = M3,5	Espando M4,6. È il nodo goal => fine. E il costo per arrivare da S a G è 124. E possibile procedere a ritroso lungo i predecessori (come mostrato nella colonna a sinistra) per ricostruire il percorso ottimo da S a G: M0,0 - M1,0 - M2,0 - M3,1 - M2,2 - M1,3 - M0,4 - M1,5 - M2,5 - M3,5 - M4,6.	

Una volta trovato il nodo goal è possibile procedere a ritroso attraverso i nodi predecessori, ricostruendo così il percorso ottimo dal punto di partenza S al punto di arrivo G. La figura A.2 visualizza tale percorso.

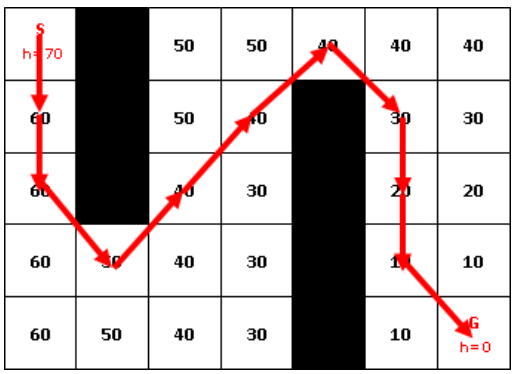


Figura A.2 - Il percorso ottimo da S a G.

Indice