**Immagine che contiene emblema, simbolo, logo, Carattere

Descrizione generata automaticamente**

### Ricerca in Struttura Universitaria

*Caso di Studio di “Ingegneria della Conoscenza”*

Gruppo di lavoro

* **Matteo Castano**, 737890, [m.castano@studenti.uniba.it](mailto:m.castano@studenti.uniba.it)
* **Ugo Gabriele De Santis**, 736572, [u.desantis2@studenti.uniba.it](mailto:u.desantis2@studenti.uniba.it)

<URL repo associato, contenente il materiale completo>

AA 2022-23

### Introduzione

L’obiettivo del caso di studio è quello di semplificare lo spostamento all’interno di una struttura universitaria calcolando il percorso più efficiente tra due posizioni inserite dall’utente.

Il percorso calcolato dipende dalle condizioni reali dell’edificio, perciò dalle posizioni dei nodi che lo compongono, ma anche dal loro stato: infatti vengono considerate eventuali inagibilità, immesse dall’utente allo start del programma (es. pavimenti bagnati). Viene tenuto conto, inoltre, di chi sta utilizzando il sistema (es. l’utilizzo degli ascensori o di alcuni corridoi “riservati” è limitato solo a determinati utenti privilegiati, l’accesso a determinate aule è consentito solo a determinati orari e per determinati utenti).

Per migliorare la ricerca è stata implementata una funzione euristica (i cui dettagli e proprietà saranno esposti successivamente) che stima quanto un nodo è vicino a un goal.

Il sistema permette di effettuare inoltre delle query default, come la ricerca del bagno più vicino.

Lo step successivo all’implementazione del sistema è una analisi relativa alla computazione degli algoritmi analizzati: saranno confrontati l’algoritmo A\* e l’algoritmo Lowest Cost First, entrambi con e senza Mutiple Path Pruning, per capire quanto la funzione euristica sviluppata incida sull’efficienza della ricerca nel nostro dominio.

Struttura dell’edificio

Immagine che contiene testo, schermata, Rettangolo, Policromia

Descrizione generata automaticamenteImmagine che contiene testo, Policromia, Rettangolo, schermata

Descrizione generata automaticamenteLa planimetria dell’edificio è stata codificata nella base di conoscenza tramite i predicati che saranno illustrati nella parte dedicata alla KB ed è strutturata nella seguente maniera:

Immagine che contiene testo, schermata, Rettangolo, quadrato

Descrizione generata automaticamenteImmagine che contiene testo, Rettangolo, schermata, quadrato

Descrizione generata automaticamenteImmagine che contiene testo, schermata, Rettangolo, quadrato

Descrizione generata automaticamente

L’edificio è composto da cinque piani nei quali sono presenti diversi ambienti; essi si distinguono in:

* **Aule per Lezioni** (Rosa): esse sono accessibili in un determinato momento solo dai soggetti interessati alla lezione che si sta tenendo: dato lo scheduling delle lezioni e delle aule, un soggetto può quindi entrare solo se è il professore della lezione che si sta tenendo oppure uno studente che segue il relativo corso.
* **Aule Studio** (Azzurro): esse sono accessibili in qualsiasi momento, ma solamente dagli studenti.
* **Bagni** (Violetto): essi sono accessibili da tutti in qualsiasi momento.
* **Uffici dei professori** (Verde): essi sono accessibili in qualsiasi momento, ma solo dal professore che possiede l’ufficio oppure dagli studenti che seguono un corso tenuto dallo stesso professore.
* **Aree Fumatori** (Arancione): esse sono accessibili da tutti in qualsiasi momento.
* **Scale** (Blu): esse collegano due piani e possono essere utilizzate da qualsiasi individuo.
* **Ascensori** (Viola): essi collegano due (o più, in alcuni casi) piani ma sono accessibili solo dai professori o da alcuni studenti che possiedono un permesso speciale.
* **Corridoi**: essi collegano tutti gli ambienti dello stesso piano, perciò sono collegati a altri corridoi e alle stanze dell’edificio; essi sono divisi in:
  + **Corridoi Normali** (Rosso): accessibili a tutti.
  + **Corridoi Riservati** (Oro): accessibili solo da professori oppure da studenti che hanno un determinato permesso.

Ogni ambiente è accessibile solo da chi ne ha autorizzazione ammesso che non ci siano condizioni che precludano il corretto utilizzo (nel nostro caso sono state previste solo le situazioni anormali di ‘pavimento bagnato’ o di ‘problema generico’).

Utenti

Il KBS è stato progettato per essere utilizzato da due categorie utente: studenti e professori. All’avvio verrà chiesto all’utente di autentificarsi con il suo id\_utente, legato nella KB a una delle due categorie e a specifici permessi.

Il sistema potrebbe essere agilmente esteso per aggiungere ulteriori categorie, caratterizzate da specifici permessi sui relativi accessi (es. custode, visitatore, ecc).

Sommario

Il sistema è implementato mediante l’impiego della libreria **pyswip**, che consente l’utilizzo del relativo interprete Prolog, per interrogare una base di conoscenza costruita in maniera semi-automatica (mediante script python per assiomatizzare i fatti e in maniera manuale per assiomatizzare le regole che descrivono le caratteristiche del nostro dominio).

Le interrogazioni sono fatte per costruire un apposito grafo di ricerca in cui i nodi sono i luoghi dell’edificio, mentre gli archi sono i collegamenti *disponibili* tra due nodi (la disponibilità dipende anche qui dalle situazioni normali/anormali dei nodi, dai permessi e altre situazioni codificate nella Knowledge Base).

La ricerca dei percorsi è stata implementata mediante le classi messe a disposizione dal libro di testo (**AIPython**), in particolare si è utilizzata la classe ‘*Search\_problem\_from\_explicit\_graph*’ del modulo ‘*SearchProblem.py*’ per costruire il grafo di ricerca, in cui i nodi sono banalmente gli ambienti dell’edificio, e le classi ‘*AStarSearcher*’ del modulo ‘*searchGeneric.py*’, ‘*SearcherMPP*’ del modulo ‘*searchMPP.py*’, **(scrivi altre classi nel caso aggiungi)** per risolvere i problemi di ricerca associati al grafo.

Per sfruttare al meglio le potenzialità degli algoritmi forniti dal libro si è sviluppata una funzione euristica, discussa in seguito.

Elenco argomenti di interesse

* **Risoluzione di Problemi Mediante Ricerca**:
  + *LCFS, A\*, Multiple Path Pruning*: utilizzo di una funzione euristica per la stima del percorso più breve basata sulla distanza euclidea; gli algoritmi sono impiegati per il calcolo del percorso migliore tra due punti dell’edificio nelle condizioni esplicitate (autenticazione, permessi, eventuali posti inaccessibili).
* **Rappresentazione e Ragionamento Relazionale**:
  + *KB in Prolog*: codifica le coordinate dei luoghi di interesse nell’edificio, gli individui che hanno accesso alla struttura, i corsi tenuti e tutte le altre informazioni che consentono di determinare, in base ai vincoli del problema iniziale, l’eventuale accesso a un ambiente dell’edificio.
  + *Vincoli di Intergrità*: sono state impiegate nella KB delle clausole speciali per codificare situazioni anomali perciò appena la KB viene caricata viene fatto un check relativo alla **soddisfacibilità** della stessa.

### Risoluzione di Problemi Mediante Ricerca

Sommario

Il KBS implementa la ricerca del cammino più economico all’interno del grafo che rappresenta l’edificio attraverso l’algoritmo A\* in congiunzione con il Multiple Path Pruning. Nella parte relativa alla valutazione saranno esposte eventuali considerazioni in merito al confronto della nostra strategia con diverse alternative quali Lowest Cost First Search, A\* senza MPP, …

In particolare sarà evidenziato come, in alcuni casi limite, strategie come A\* senza MPP presentano un notevole aumento del tempo di computazione al minimo aumento della distanza tra i due punti scelti.

Strumenti utilizzati

Gli strumenti utilizzati sono stati i seguenti: libreria **AIPython** con le relative classi ‘*Search\_problem\_from\_explicit\_graph*’ del modulo ‘*SearchProblem.py*’ per costruire il grafo di ricerca, in cui i nodi sono banalmente gli ambienti dell’edificio, e le classi ‘*AStarSearcher*’ del modulo ‘*searchGeneric.py*’, ‘*SearcherMPP*’ del modulo ‘*searchMPP.py*’ per risolvere i problemi di ricerca associati al grafo.

Decisioni di Progetto

La principale scelta progettuale, oltre alla tipologia di algoritmo utilizzato, è stata la funzione euristica da impiegare. A\* è un algoritmo di ricerca informata, che quindi è notevolmente influenzato dalla bontà della medesima funzione.

Per progettare la funzione euristica ci si è focalizzati sul soddisfare due caratteristiche principali: la prima è ovviamente la **accettabilità**, la seconda è la **consistenza**.

La prima proprietà garantisce che la valutazione della funzione euristica relativa a un nodo restituisca sempre una *sottostima* del costo del miglior percorso fino al gol, la seconda invece ci garantisce che, per ogni arco, l’euristica del nodo di partenza non sia mai maggiore strettamente del costo dell’arco sommato all’euristica del nodo di arrivo; in particolare quest’ultima è essenziale per garantire l’ottimalità della soluzione trovata da A\* con MPP (che altrimenti non sarebbe garantita).

La funzione euristica sviluppata si basa perciò sulla distanza euclidea: nel caso il nodo di partenza si trovi sullo stesso piano del nodo goal, il valore calcolato corrisponderà proprio la distanza euclidea tra i due punti; nel caso in cui invece i due nodi si trovino su piani diversi, si sceglierà come valore il minimo, per ogni metodo per cambiare piano, della somma della distanza euclidea tra il nodo iniziale e il metodo per cambiare piano, il costo dell’utilizzo del suddetto mezzo, e l’euristica del nodo di arrivo calcolata in maniera ricorsiva.

Nel caso ci siano più nodi goal, l’euristica equivarrà al minimo delle singole euristiche calcolate.

Formalmente, sia *n* il nodo di cui calcolare l’euristica, *G* l’insieme dei nodi obiettivo, *d(nodo1,nodo2)* la funzione che calcola la distanza euclidea (come radice quadrata della somma dei quadrati delle differenze delle coordinate), *down(nodo)* (risp. *up(nodo))* funzione che restituisce, dato unmetodo per scendere (risp. salire), il relativo metodo che si trova al piano sottostante (risp. sovrastante), *cost(nodo1,nodo2)* funzione che calcola il costo di utilizzo del relativo metodo di discesa (risp. salita), si ha che:

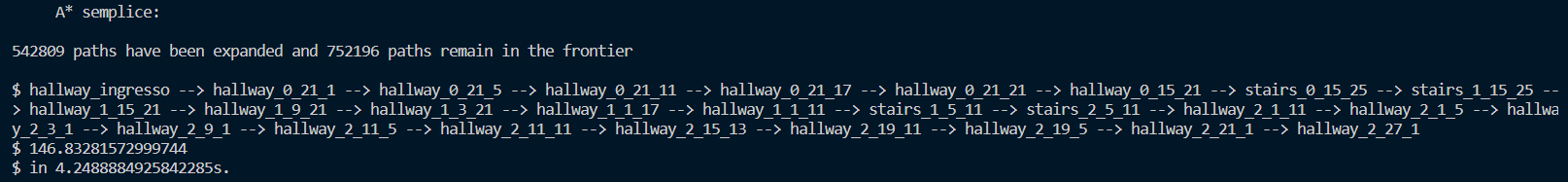
dove:

Notiamo che *h(n)* è una funzione euristica che è **ammissibile**, poiché la distanza euclidea, per disuguaglianza triangolare, è sempre sottostima del costo del percorso migliore nel problema ‘vincolato’, inoltre il calcolo del minimo ci assicura di star andando a prendere sempre la sottostima del percorso più breve possibile. Inoltre questa funzione è anche **consistente** per restrizione di monotonicità, quindi raggiungendo da *n* un qualsiasi nodo *n’*, l’euristica di *n* non sarà mai maggiore rispetto alla somma di *h(n’)+cost(n,n’)*, questo sempre per le proprietà della distanza euclidea.

Valutazione

Il nostro sistema lavora in maniera ottimale se ad A\* viene applicato MPP. Senza quest’ultimo, infatti, A\* è soggetta a notevoli problematiche dovute al fatto che al crescere della dimensione del percorso ottimo il numero di percorsi nella frontiera cresce in maniera esponenziale.

Esponiamo ora alcuni esperimenti per una valutazione empirica:

eseguiamo la ricerca del percorso ottimo (identificandoci come *student\_1* che nel nostro sistema non ha accesso a corridoi riservati e agli ascensori) tra gli ambienti *hallway\_ingresso* e *hallway\_2\_27\_1.* Usando A\* senza MPP, il risultato è il seguente:

la ricerca termina in 4.25 secondi, espandendo 542809 percorsi; inoltre al momento del ritrovamento della soluzione, 752196 percorsi si trovano ancora nella frontiera.

Immagine che contiene schermata, Carattere, testo

Descrizione generata automaticamenteValutiamo ora la ricerca tra *hallway\_ingresso* e *hallway\_2\_29\_5*, che è un ambiente vicino al precedente, perciò comporta uno spostamento minimo:

la ricerca qui impiega 9.34 secondi, più del doppio, espandendo 1207387 percorsi, con 1672021 percorsi ancora nella frontiera al termine dell’algoritmo.

Questo è rappresentativo del fatto che, anche in un problema con un insieme di nodi così piccolo (appena **147 nodi**) e un fattore di ramificazione uscente e entrante di **4 vicini per nodo**, lo spazio di ricerca cresce in maniera esponenziale, rendendo infatti poco utile A\* senza MPP dopo una certa soglia di costo del percorso da rintracciare.

### Rappresentazione e Ragionamento Relazionale

Sommario

Un paragrafo che chiarisca la rappresentazione della conoscenza scelta per KB (modelli di ragionamento / apprendimento), dati, BK, ...

Strumenti utilizzati

Breve: non serve spiegare come funzionano se implementano modelli ben noti   
(basta indicare dei riferimenti bibliografici)

Dilungarsi solo su eventuali modelli/algoritmi originali ideati dal gruppo

Decisioni di Progetto

Configurazione dei componenti (e.g. API/librerie) utilizzati, ad es. parametri, soglie, ecc.   
e di eventuali metodi specifici utilizzati a tale scopo

Valutazione

Paragrafi che richiamino (non spieghino, se standard) le metriche adottate   
+ tabelle sui risultati e loro discussione

### Conclusioni

Un paragrafo che riassuma le valutazioni e delinei possibili sviluppi, ad. es. problematiche non affrontate per questioni di tempo (per eventuali estensioni da parte di altri gruppi).

Riferimenti Bibliografici

[1] ...

[2] ...

[3] ...