<Titolo: Template Documentazione Progetto>

Gruppo di lavoro

* Nome\_Cognome1, matricola1, [username1@studenti.uniba.it](mailto:username1@studenti.uniba.it)
* Nome\_Cognome2, matricola2, [username2@studenti.uniba.it](mailto:username2@studenti.uniba.it)
* Nome\_Cognome3, matricola3, [username3@studenti.uniba.it](mailto:username3@studenti.uniba.it)
* ...

<URL repo associato, contenente il materiale completo>

AA 202X-2(X+1)

Introduzione

Paragrafo/i sul dominio di interesse

Sommario

Paragrafo sul KBS e su come integri moduli che dimostrino competenze sui diversi argomenti (specificati sotto)

Elenco argomenti di interesse

* Paragrafo su argomento 1 (es. CSP, rappresentazione della conoscenza: clausole di Horn)
* Paragrafo su argomento 2 (es. Ragionamento automatico)
* Paragrafo su argomento 3 (es. Apprendimento e incertezza, ragionamento su KB distribuite)
* ...

(tratti da sezioni diverse del programma, da indicare esplicitamente)

# **Ricerca di Soluzioni in Spazi di Stati**

## **Introduzione**

Per la trattazione di questo argomento, l’obiettivo fissato è stato quello testare diversi algoritmi di ricerca per determinare il percorso più sicuro in una rete stradale urbana, nello specifico della città di Bari. Di fatto, ogni algoritmo è stato testato sulla ricerca di un percorso da Piazza Umberto Primo a Parco Maria Maugeri.

## **Rappresentazione della conoscenza per KB**

In questo caso, la conoscenza è rappresentata utilizzando un grafo **G** che mappa la rete stradale della città di Bari. Ogni nodo del grafo rappresenta un incrocio o un punto di interesse, mentre gli archi collegano questi nodi e rappresentano i segmenti stradali tra i vari punti. Inoltre, ogni arco è stato arricchito con le informazioni relative al **numero di incidenti** avvenuti su quel segmento stradale, tratte da un dataset previamente elaborato grazie a dati forniti da enti regionali che si occupano dell’acquisizione e della rappresentazione di tali informazioni.   
Di seguito un’immagine illustrativa del grafo rappresentativo della rete stradale:

Immagine che contiene mappa, schermata

Descrizione generata automaticamente

## **Librerie utilizzate**

* [***OSMnx***](https://osmnx.readthedocs.io/en/stable/): libreria Python per scaricare, modellare, analizzare e visualizzare facilmente reti stradali e altre funzionalità geospaziali da OpenStreetMap.
* [***NetworkX***](https://networkx.org/)***:*** libreria Python per la creazione e manipolazione di grafi complessi.
* [***Folium***](https://python-visualization.github.io/folium/latest/)***:*** libreria Python utile a visualizzare direttamente su mappa i risultati ottenuti.
* [***Pandas***](https://pandas.pydata.org/): libreria Python utile alla manipolazione dei dati.
* [***Heapq***](https://docs.python.org/3/library/heapq.html): coda con priorità, struttura dati utile negli algoritmi di ricerca implementati.
* [***Geopy***](https://geopy.readthedocs.io/en/stable/): libreria Python utile a manipolare dati geografici.

## **Ricerca non informata**

Per la *ricerca non informata* è stato l’implementato l’***Iterative Deepening Depth First Search.***L'algoritmo **Iterative Deepening Depth First Search (IDDFS)** è un ibrido che combina le caratteristiche della **ricerca in profondità** (DFS) e della **ricerca per livello** (BFS). La sua peculiarità risiede nel fatto che esegue ripetutamente una ricerca in profondità, ma limitando la profondità massima per ogni iterazione. Ad ogni ciclo, la profondità massima viene incrementata, fino a trovare il percorso desiderato o esaurire lo spazio di ricerca.

Nel codice proposto, l'algoritmo è composto da due funzioni principali:

1. **Funzione** dls **(Depth-Limited Search)**:  
   Questa funzione esegue una ricerca in profondità limitata, in cui si esplorano i nodi del grafo a partire dal nodo iniziale (start), fino a una profondità massima specificata. Ogni volta che si esplora un nodo, se non è già stato visitato, viene aggiunto allo stack con il percorso accumulato fino a quel punto. Se il nodo di arrivo (goal) viene raggiunto, la funzione restituisce il percorso; altrimenti, la ricerca continua fino a raggiungere la profondità limite.
2. **Funzione** iterative\_deepening\_dfs:  
   Questa funzione è responsabile dell'iterazione del processo di ricerca in profondità. Ad ogni iterazione, l'algoritmo incrementa la profondità massima da 0 fino al valore massimo di **max\_depth**. Ogni volta che la ricerca raggiunge la profondità massima senza trovare la soluzione, viene incrementata la profondità e la ricerca viene ripetuta. In caso di successo, l'algoritmo restituisce il **percorso trovato** insieme ad alcune statistiche, come la **profondità raggiunta**, il **tempo di esecuzione** e la **memoria utilizzata**.

## **Valutazione**

Dopo aver eseguito l’algoritmo settando il parametro **max\_depth = 50** (per permettere all'algoritmo di esplorare una quantità significativa di soluzioni prima di terminare, senza limitare eccessivamente la profondità di ricerca) i risultati ottenuti sono i seguenti:

* **Percorso più breve trovato (in rosso sulla mappa)**:  
  [270659688, 270388628, 322548994, 322549051, 322550079, 322549607, 322550392, 322550761, 320970977, 330655154, 330655155, 270388358, 320971935, 320971934, 270437927, 353330854, 270389790, 329988604, 329988602, 270655174, 270654641, 1481415203].  
  Ogni elemento all’interno della lista rappresenta l’id del nodo attraversato per arrivare alla soluzione.
* **Percorsi totali esplorati (in blu sulla mappa):** *63991*
* **Profondità raggiunta:** *21*  
  Ciò significa che per navigare dal punto di partenza a quello di arrivo sono necessari bisogna attraversare 21 segmenti (archi) nella rete.
* **Tempo di esecuzione:** *0.0499 secondi*
* **Memoria utilizzata:** 562432 byte

Immagine che contiene mappa, testo

Descrizione generata automaticamente

## **Applicazione Multiple-Path Pruning**

Per ottimizzare IDDFS è stata applicata la tecnica del **Multiple-Path Pruning** per evitare la riesplorazione di percorsi già esplorati. Questa tecnica comporta un miglioramento nell'efficienza della ricerca, in quanto impedisce di esplorare ripetutamente percorsi che non conducono a soluzioni migliori o più rapide.  
Nella **versione precedente**, l'algoritmo eseguiva una ricerca in profondità limitata (DLS) senza tenere traccia in modo efficiente dei percorsi già esplorati. Ciò significava che, in alcuni casi, venivano riesplorati gli stessi percorsi, aumentando il numero di nodi esplorati e il consumo di memoria.

Con la **versione ottimizzata**:

1. **Pruning dei percorsi ridondanti**: L'algoritmo ora utilizza un dizionario visited, per evitare di esplorare nodi già visitati con un percorso più lungo. Questo riduce notevolmente il numero di percorsi esplorati.
2. **Ottimizzazione del consumo di memoria**: Poiché vengono evitati percorsi ripetitivi, la memoria utilizzata è significativamente ridotta rispetto alla versione precedente.
3. **Maggiore efficienza**: La ricerca avviene più velocemente, con un numero minore di percorsi esplorati, migliorando il tempo di esecuzione complessivo.

## **Valutazione**

L’esecuzione dell’algoritmo ottimizzato ha riportato i seguenti risultati:

* **Percorso più breve trovato (in rosso sulla mappa)**:  
  [270659688, 270388628, 322548994, 322549051, 322550079, 322549607, 322550392, 322550761, 320970977, 330655154, 330655155, 270388358, 320971935, 320971934, 270437927, 353330854, 270389790, 329988604, 329988602, 270655174, 270654641, 1481415203].  
  *Uguale a quello trovato precedentemente*
* **Percorsi totali esplorati (in blu sulla mappa):** *8020*Circa l’87% in meno rispetto all’implementazione senza MPP.
* **Profondità raggiunta:** *21*
* **Tempo di esecuzione:** *0.0126 secondi*
* **Memoria utilizzata:** 67168 byte

Immagine che contiene mappa, testo, atlante

Descrizione generata automaticamente

## **Confronti e conclusioni**

|  |  |  |
| --- | --- | --- |
| RISULTATO | IDDFS | IDDFS con MPP |
| Percorso trovato | Stesso | Stesso |
| Profondità raggiunta | 21 | 21 |
| Percorsi esplorati | 63991 | 8020 |
| Tempo di esecuzione | 0.04999 s | 0.0126 s |
| Memoria utilizzata | 562432 byte | 67168 byte |

L'algoritmo ottimizzato ha portato a **un numero significativamente inferiore di percorsi esplorati** (da 63991 a 8020), a un **miglioramento notevole nel tempo di esecuzione** (da 0.0499 a 0.0126 secondi) e a una **riduzione drastica della memoria utilizzata** (da 562432 byte a 67168 byte). Questi miglioramenti sono il risultato diretto dell'implementazione del **Multiple-Path Pruning**, che ha permesso di evitare la riesplorazione dei percorsi già visitati, ottimizzando l'efficienza complessiva dell'algoritmo.

# Sezione Argomento 2

## 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

# Sezione Argomento 3

## 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] ...