Documentazione del Progetto di Intelligenza Artificiale

Adriano Oliviero

July 2, 2024

Contents

1 Obiettivo del progetto

Il progetto si propone di applicare algoritmi di ricerca ad alcuni dataset disponibili su `https://snap.stanford.edu/data/` per valutare l'efficacia e l'efficienza di tali algoritmi.

2 Descrizione delle metodologie e tecniche adoperate

2.1 Il linguaggio

Per la realizzazione del progetto, ho utilizzato il linguaggio di programmazione Rust.

Questo linguaggio è stato scelto per diversi motivi:

- **Performance**: Rust è un linguaggio di programmazione ad alte prestazioni con controllo della memoria a tempo di compilazione.
- **Semplicità**: Rust è un linguaggio di programmazione moderno con una sintassi pulita e concisa, quasi alla pari di Python.
- **Esperienza**: Ho già esperienza con Rust e ho trovato che sia un linguaggio adatto per progetti di questo tipo.

2.2 Organizzazione del progetto

Il progetto è organizzato come segue:

- `download-datasets.sh` - Script shell per scaricare i dataset automaticamente.
- `run.py` - Script Python per eseguire il progetto, e generare benchmark e grafici.
- `src/` - Directory contenente il codice sorgente in Rust.
- file aggiuntivi

2.3 Compilazione ed esecuzione

Per compilare ed eseguire il progetto, è necessario avere Rust (e cargo) installato sul proprio sistema. Inoltre è necessario scaricare almeno un dataset, come descritto nella sezione dedicata.

Per compilare il progetto, eseguire i seguenti comandi:

```
$ cargo build --release
```

Per eseguire il progetto, eseguire il seguente comando:

```
$ ./target/release/eia <opzioni>
```

Per ottenere la lista delle opzioni disponibili, è possibile eseguire il programma con l'opzione `-h`

2.4 Strutture dati

Per rendere utilizzabili i dataset, ho implementato le seguenti strutture dati:

- `Problem` - Struttura dati contenente il grafo e i dati e le funzioni necessarie per la ricerca:
  - `stato_inziale: State` - Nodo dal quale iniziare la ricerca.
  - `stato_finale: State` - Nodo da raggiungere.
  - `grafo: Graph` - Struttura dati contenente il grafo.
  - `limite: usize` - Limite di profondità per gli algoritmi di ricerca limitata.
  - `goal_test(&self, stato) -> bool` - Funzione per verificare se il nodo obiettivo è stato raggiunto.
  - le funzioni di ricerca, delle quali parlerò più avanti.
- `Graph` - Struttura dati contenente una astrazione del grafo:
  - `gtype: String` - Tipo del grafo (direzionato, non direzionato o con pesi).
  - `nodi: Vec<Node>` - I nodi del grafo.
  - `edge_count: u32` - Il numero di archi del grafo, utile per essere sicuri che il caricamento del dataset sia avvenuto correttamente.
  - `load_dataset(dataset_path)` - Legge il file del dataset e costruisce il grafo.
- `Node` - Struttura dati per rappresentare un nodo del grafo:
  - `stato: State` - Stato corrispondente al nodo.
  - `azioni: Vec<Action>` - Azioni possibili dal nodo.
  - `genitore: Node` - Nodo genitore, utile per risalire il percorso.
  - `costo_camm: usize` - Costo del cammino partendo dal nodo iniziale per raggiungere il nodo.
  - `profondità: usize` - Profondità del nodo rispetto al nodo iniziale.
- `Action` - Struttura dati per rappresentare un'azione:
  - `risultato: State` - Stato risultante dall'azione.
  - `costo: i32` - Costo dell'azione.
- `State: u32` - Un semplice alias per rendere più leggibile il codice.

2.5 Algoritmi di ricerca

Gli algoritmi di ricerca che ho scelto di implementare sono:

- **Tree Search (tree-search)** - Ricerca semplice, per default disattivata a causa della sua eccessiva inefficienza.
- **Breadth-First Search (breadth-first)** - Ricera in ampiezza.
- **Uniform-Cost Search (uniform-cost)** - Ricerca a costo uniforme.
- **Depth-Limited Search (depth-limited)** - Ricerca in profondità limitata.
- **Iterative Deepening Depth-First Search (iterative-deepening)** - Ricerca in profondità iterativa.
- **Bidirectional Search (bi-directional)** - Ricerca bidirezionale.

Tutti gli algoritmi sono compatibili sia con grafi direzionati che non direzionati, e con grafi pesati.

3 Dataset

I dataset utilizzati sono stati scaricati dal sito dell'università di Stanford (`https://snap.stanford.edu/data/`). Per scaricare i dataset, è possibile procedere manualmente recandosi alle reciproche pagine sul sito, oppure utilizzare lo script shell fornito nel progetto:

```
$ chmod +x ./download-datasets.sh
$ ./download-datasets.sh
```

Lo script utilizza `wget` ed è scritto per sistemi UNIX & UNIX-like.

3.1 Dataset Utilizzati

Nome	Nodi	Archi	Tipologia	Dimensione
soc-sign-bitcoin-alpha	3783	24186	Con pesi	152KB
email-Enron	36692	183831	Non direzionato	1.1MB
com-Youtube	1134890	2987624	Non direzionato	11MB
roadNet-CA	1965206	2766607	Direzionato	18MB
as-Skitter	1696415	11095298	Non direzionato	33MB
cit-Patents	3774768	16518948	Direzionato	85MB
com-LiveJournal	3997962	34681189	Non direzionato	124MB

Table 1: Dataset Utilizzati

I dataset contengono alcune informazioni nelle prime righe. Sono in formato `txt` con compressione `.gz` e le proprie righe sono formate da due numeri (Nodo Sinistro e Nodo

Destro), ad eccezione del dataset soc-sign-bitcoin-alpha, che è in formato `csv` con le colonne:

- **SOURCE** (id del nodo Sinistro),
- **TARGET** (id del nodo Destro),
- **RATING** (il costo delle azioni),
- **TIME** (non rilevante).

4 Risultati sperimentali

I risultati ottenuti sono stati valutati in termini di efficacia ed efficienza, come descritto di seguito.

Inoltre, sono stati generati grafici automaticamente dallo script Python fornito nel progetto. È possibile visualizzarli nel documento grafici.pdf

4.1 email-Enron.txt.gz

Tipo di Grafo: Directed

Durata caricamento: 1.342s

Nodi cercati: 2052 e 4826

Algoritmo	Risultato	Profondità	Costo	Tempo
bi-directional	Trovato	3	0	8.165407s

Table 2: email-Enron.txt.gz

5 Codice Sviluppato

Il codice sviluppato è stato consegnato insieme alla documentazione del progetto e può essere consultato nei file allegati.

Alternativamente è possibile trovare il codice sorgente su GitHub al seguente indirizzo:

ad-oliviero/progetto_eia

References

[1] Jure J. Leskovec et al. “Community Structure in Large Networks: Natural Cluster Sizes and the Absence of Large Well-Defined Clusters”. In: *Internet Mathematics*. Vol. 6. 1. 2009, pp. 29–123.

[2] Benjamin Klimmt and Yiming Yang. “Introducing the Enron corpus”. In: *CEAS conference*. 2004.

[3] Srijan Kumar et al. “Edge weight prediction in weighted signed networks”. In: *Data Mining (ICDM), 2016 IEEE 16th International Conference on*. IEEE. 2016, pp. 221–230.

[4] Srijan Kumar et al. “Rev2: Fraudulent user prediction in rating platforms”. In: *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM. 2018, pp. 333–341.

[5] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. “Graphs over Time: Densification Laws, Shrinking Diameters and Possible Explanations”. In: *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 2005.

[6] Jaewon Yang and Jure Leskovec. “Defining and Evaluating Network Communities based on Ground-truth”. In: *ICDM*. 2012.