

POLITECNICO DI TORINO

DIPARTIMENTO DI INGEGNERIA GESTIONALE E DELLA
PRODUZIONE

Corso di Laurea in Ingegneria Gestionale
Classe L8 – Ingegneria dell'Informazione



Relazione dell'Esame Finale

Sviluppo di un software gestionale per assistenza hotspot Wi-Fi pubblici della città di New York

Relatore

Prof. Fulvio Corno

Candidato

Eugenio Simone Errigo

A.A. 2018/2019

Indice

Copia integrale della proposta	3
Studente proponente.....	3
Titolo della proposta	3
Descrizione del problema proposto.....	3
Descrizione della rilevanza gestionale del problema	3
Descrizione dei data-set per la valutazione.....	3
Descrizione preliminare degli algoritmi coinvolti	4
Descrizione preliminare delle funzionalità previste per l'applicazione software.....	4
Descrizione dettagliata del problema affrontato	5
Descrizione del data-set utilizzato per l'analisi	7
Descrizione ad alto livello delle strutture dati e degli algoritmi utilizzati	8
Diagramma delle classi delle parti principali dell'applicazione	11
Alcune videate dell'applicazione realizzata	12
Risultati sperimentali ottenuti	14
Conclusioni.....	15
Bibliografia.....	16

Copia integrale della proposta

Studente proponente

s223412 Errigo Eugenio Simone

Titolo della proposta

Assistenza hotspot Wi-Fi pubblici della città di New York

Descrizione del problema proposto

Si intende creare un'applicazione JavaFX da usare in ambito industriale, in grado di ricercare il cammino di costo minimo all'interno di un grafo che rappresenta le distanze di intervento tra gli hotspot pubblici della città di New York. In particolare, l'azienda è incaricata dai fornitori del servizio Wi-Fi pubblico, interessati alla riparazione/sostituzione di tutti gli hotspot, nell'ambito del rinnovamento della linea. Il manager può ottenere delle stime sul tempo di lavoro, in modo da pianificare i turni, rispetto al tempo disponibile e ai percorsi che lo minimizzano.

Descrizione della rilevanza gestionale del problema

Oggigiorno in azienda vengono usati numerosi software per la pianificazione. Il problema di controllo logistico mira ad ottimizzare i tempi e a migliorare il servizio, riducendone i costi.

La pianificazione dei turni è un problema realmente affrontato nella gestione di un'azienda che, però, ha bisogno di numerosi parametri "umani". L'applicazione produce una stima quanto più veritiera possibile, che, integrata con l'esperienza e con le competenze, fornisce un'ottima approssimazione.

Descrizione dei data-set per la valutazione

Il data-set "NYC Wi-Fi hotspot location" contiene i dati reali sulla distribuzione degli hotspot nella città di New York. I dati sono aggiornati ad Ottobre 2018 e sono resi disponibili da NYC Open data sul sito www.kaggle.com.

Il database è costituito da una sola tabella, con molteplici campi: l'hotspot è identificato da un id e SSID, dalla sua posizione (latitudine e longitudine, città, distretto), il nome del provider, il tipo di servizio, e molteplici altri dati non utili ai nostri fini. Il dataset è in formato CSV (Comma-Separated Values).

Descrizione preliminare degli algoritmi coinvolti

L'applicazione implementa un algoritmo ricorsivo con lo scopo di trovare il cammino ottimale, ossia la sequenza di riparazioni che minimizza la distanza da percorrere per raggiungere tutti gli hotspot dell'area selezionata. Il problema proposto è una applicazione del Traveling Salesman Problem.

Il grafo rappresenta la mappatura degli hotspot (vertici), ed un arco collega due hotspot se il peso dell'arco è minore della disponibilità del tecnico di spostarsi (input utente).

Vengono effettuati controlli sulla coerenza e integrità dei dati, con la presenza di messaggi di errore in caso di fallimento.

L'applicazione è implementata secondo il pattern MVC (Model-View-Controller).

Descrizione preliminare delle funzionalità previste per l'applicazione software

L'applicazione JavaFX permette all'utente di relazionarsi con un'interfaccia grafica, tramite cui scegliere il provider a cui fornire assistenza ed un distretto di New York (o tutta la città). Infine, è possibile impostare la distanza massima (in km) tra i luoghi di intervento (hotspot), la probabilità di guasto e la possibilità di avere una stima approssimata.

Il risultato ottenuto sarà visibile su una nuova interfaccia e conterrà l'elenco ottimale di hotspot da attraversare per ogni area (componente connessa), con delle stime sulla distanza e sul tempo: l'addetto considererà quanti tecnici impiegare per l'area.

Inoltre, in caso di errore, l'utente è in grado di avere un feedback su quanto accaduto.

Descrizione dettagliata del problema affrontato

Il problema analizzato è un'applicazione del Traveling Salesman Problem (Problema del commesso viaggiatore). Spesso abbreviato TSP, è uno dei maggiori casi di studio nell'ambito della ricerca operativa; fa parte dei problemi di ottimizzazione combinatoria, il cui scopo è ricercare una soluzione ottima in un insieme discreto di soluzioni ammissibili. L'azienda ricorre spesso ad algoritmi di questo genere per la pianificazione, coadiuvata dall'utilizzo di strumenti informatici per far fronte alla mole di informazioni da elaborare.

L'obiettivo generale del TSP è individuare il percorso di minore lunghezza che un commesso viaggiatore deve eseguire per visitare tutte le città di un'area selezionata una sola volta, eventualmente tornando alla città di partenza.

L'algoritmo lavora su un grafo, una struttura dati costituita da un numero V di nodi e un numero E di archi che li collegano, ognuno con un peso w . Nel caso del TSP, l'insieme dei vertici V corrisponde ai luoghi di intervento nell'area considerata e l'insieme degli archi E rappresentano il collegamento "in linea d'aria" tra essi, con peso pari alla distanza. Il TSP si dice asimmetrico se il grafo è orientato, mentre si dice simmetrico se il grafo è non orientato.

In altre parole, il TSP consiste nel determinare, se esiste, un *ciclo hamiltoniano* di costo minimo sul grafo. Un ciclo hamiltoniano è un ciclo che visita tutti i nodi del grafo una e una sola volta. Determinare se questo esista è un problema NP-completo, ossia non deterministico risolvibile in tempo polinomiale.

Modello matematico:

$$\begin{aligned} & \min \sum_{i,j \in V, i \neq j} w_{ij} x_{ij} \\ & s.t. \\ & \sum_{i \in V} x_{iv} = 1 \quad \forall v \in V \\ & \sum_{j \in V} x_{vj} = 1 \quad \forall v \in V \\ & \sum_{i \in Q} \sum_{j \in V \setminus Q} x_{ij} \geq 1 \quad \forall Q \subset V, |Q| \geq 1 \\ & x_{ij} = \begin{cases} 1 & \text{se } (i,j) \in E \\ 0 & \text{altrimenti} \end{cases} \end{aligned}$$

Detta x_{ij} la generica variabile binaria che rappresenta l'appartenenza o meno di un arco al circuito e w_{ij} il peso dell'arco per andare dal nodo i al nodo j , la funzione obiettivo minimizza la somma dei costi degli archi scelti, sottoposti ai vincoli:

- di *assegnazione*: in ogni nodo, entra ed esce un solo arco.

- di *eliminazione dei sotto-cicli*: comunque si scelga un sottoinsieme proprio di nodi Q in V , deve esistere almeno un arco che colleghi un nodo in Q con un nodo non appartenente a Q .

Il TSP della nostra analisi prende forma dai parametri inseriti dall'utente nell'interfaccia di avvio dell'applicazione: in base alla percentuale di guasto, al provider e al distretto della città di NY scelti, si costruisce un grafo che vede come nodi gli hotspots pubblici dell'area considerata, collegati tra loro da un arco se a distanza non maggiore della massima disponibilità a spostarsi dell'operaio (input utente).

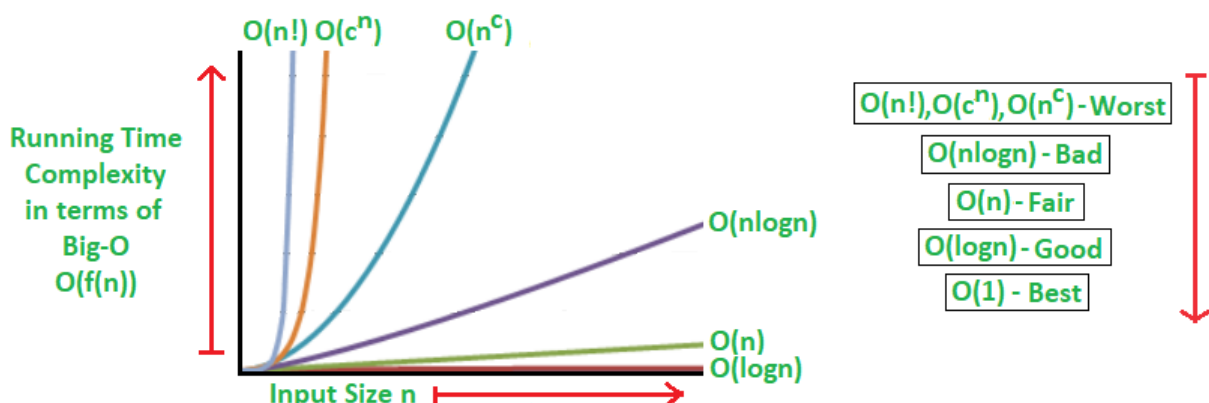
Trovare una soluzione a questo problema non è così banale, come è evidente dal numero di vincoli tanto grande all'aumentare del numero di nodi nel grafo. Ancora oggi non esiste una soluzione esatta ottenibile in tempo polinomiale, perciò spesso si ricorre ad approssimazioni, che rilassano alcuni vincoli.

L'approccio "brute-force" o dell'enumerazione totale, scelto per risolvere il problema in analisi, consiste nell'elaborazione di tutte le possibili permutazioni dei cammini fino a trovare quello di costo minimo. L'idea di tenere traccia, per ogni nodo visitato, dei nodi attraversabili in seguito è implementata da un algoritmo ricorsivo, che permette di richiamare la stessa funzione sul grafo più volte, aggiungendo e rimuovendo i nodi in modo da trovare il percorso migliore. Il *migliore* sarà quello con costo più basso, ossia somma del peso degli archi attraversati minore.

Tuttavia, il "brute-force" non è sicuramente la soluzione più efficiente poiché è per definizione un algoritmo fattoriale, ossia la funzione ricorsiva è richiamata $(V - 1)!$ volte, un numero crescente in base al numero di nodi del grafo. Le scelte effettuate per rendere praticabile questo tipo di operazione sono: utilizzare grafi con numero di nodi basso o, se impossibile, applicare delle euristiche, che hanno un'alta probabilità di produrre una buona soluzione velocemente.

Nel caso in cui l'algoritmo ricorsivo impiega un tempo relativamente elevato, o se è l'utente a sceglierlo in input, il TSP è risolto attraverso un algoritmo di approssimazione, che restituisce una soluzione subottimale ammissibile in tempo polinomiale. L'algoritmo 2-*approssimato* utilizzato garantisce che il peso totale C del cammino individuato sia minore di 2 volte il peso della soluzione ottima (OPT): $C \leq 2 * OPT$.

I due approcci hanno tempi di esecuzione diversi, che variano in base al numero di vertici del grafo. La complessità computazionale risulta molto elevata $O(V!)$ nel caso dell'algoritmo ricorsivo, mentre ha un tempo $O(|V|^2 \log|V|)$ minore per l'approssimazione.



Descrizione del data-set utilizzato per l'analisi

L'applicazione è implementata e testata con i dati del servizio hotspot Wi-Fi pubblico della città di New York. Il data-set, di grandi dimensioni, è scaricabile dal sito <https://www.kaggle.com/new-york-city/nyc-wi-fi-hotspot-locations>, concesso (e aggiornato periodicamente) da NYC Open Data. Esso contiene tutte le informazioni utili circa gli hotspots pubblici e la loro localizzazione.

I dati sono forniti in formato CSV, vengono convertiti per un migliore utilizzo in SQL.

Il data-set contiene un'unica tabella con molteplici campi, che identificano un hotspot:

- ID (chiave primaria);
- Borough: distretto di New York a cui appartiene;
- Type: tipo di WiFi (libero, limitato, ecc);
- Provider: fornitore del servizio;
- Name: nome del luogo fisico;
- Location: indirizzo;
- Latitude;
- Longitude;
- City;
- Remarks: commenti sul funzionamento dell'hotspot;
- SSID: nome di identificazione agli utenti.

I dati sono reali e molto specifici spazialmente. Il database è arricchito da numerosi altri campi, che risultano ridondanti o poco utili all'applicazione in esame.

Descrizione ad alto livello delle strutture dati e degli algoritmi utilizzati

L'applicazione esegue più operazioni. La principale struttura dati su cui lavora è un grafo, che rappresenta gli hotspot e i loro collegamenti (vedi sopra).

E' implementato il pattern MVC (Model-View-Controller), che sfruttando i principi della programmazione a oggetti, consente un'architettura software separata in 3 ruoli principali:

- il model (classe Model) fornisce i metodi per accedere ai dati utili all'applicazione;
- il view (interfaccia) visualizza i dati e si occupa dell'interazione con l'utente;
- il controller (classi xController) riceve i comandi dell'utente attraverso il view e li esegue utilizzando il model.

A esso è associato un secondo pattern, DAO (Data Access Object), che permette di accedere ai dati sul database attraverso query all'interno di metodi e renderli fruibili per l'applicazione. Attraverso la classe DAO, infatti, in fase di inizializzazione, vengono prelevate le informazioni da inserire nelle tendine dell'interfaccia e quelle per popolare il grafo.

La persistenza dei dati è ottenuta mediante il pattern ORM (Object-Relational Mapping), che favorisce l'integrazione col database senza errori o ripetizioni.

Avviata l'applicazione, all'utente è richiesto di scegliere i parametri per la costruzione della struttura dati; in caso di errori o parametri mancanti, il sistema avvisa l'utente con un messaggio. Solo dopo la verifica di tutti i parametri, il grafo viene creato immediatamente.

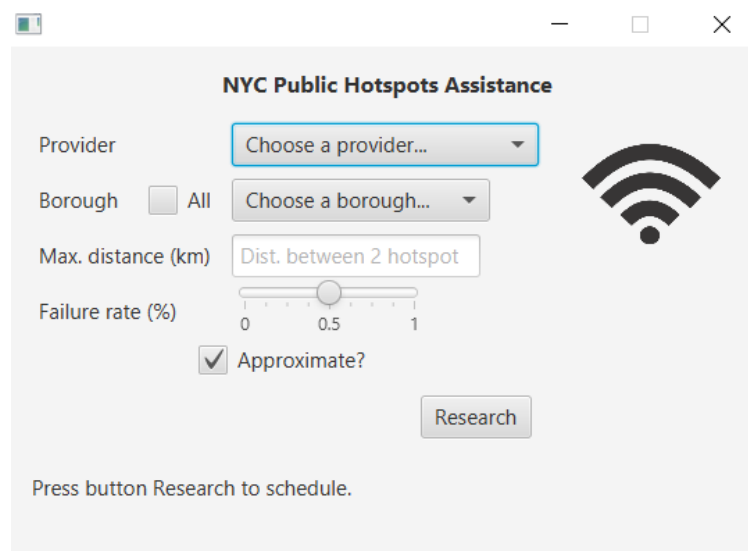


Figura 1. Finestra di avvio

Per la creazione della struttura dati è stata utilizzata la libreria *jgrapht*, contenente delle classi per la generazione di grafi. I dati da inserire sono prelevati dal db attraverso metodi della classe *HotspotDAO*, in base al provider scelto, al distretto (o tutta l'area), alla percentuale di guasto (viene eliminato un numero proporzionale di hotspot in maniera casuale) e alla distanza tra due hotspot (calcoli effettuati con la libreria *SimpleLatLng*). E' così generato un grafo semplice e pesato, con numero di vertici variabile e numero di archi vincolato dalla distanza massima di spostamento dell'operaio, che spesso lo rende piuttosto disconnesso.

Alla pressione del bottone “Research”, viene avviato l’algoritmo vero e proprio. Le operazioni avvengono distintamente su ciascuna delle aree create per facilitare lo spostamento dell’operatore; queste aree rappresentano le componenti connesse del grafo, dove cioè hotspot vicini sono raggiungibili entro la disponibilità a spostarsi inserita nella finestra iniziale.

Nel caso non sia stata scelta l’approssimazione, le operazioni sulla componente connessa avvengono nella classe *TSP*, che implementa un algoritmo ricorsivo per la ricerca del cammino di peso minimo tra i nodi attraverso la valutazione di tutti i possibili cammini. Un algoritmo si dice ricorsivo se la sua implementazione contiene una chiamata a sè stesso. Lo scopo è appunto la valutazione di infiniti sottoinsiemi di dati; non può quindi mancare una condizione di terminazione (valori particolari) e il *backtracking* (rimette a posto l’elemento provato dopo aver esplorato una soluzione).

```
void recursive (... , level) {  
  
    // E -- sequenza di istruzioni che vengono eseguite sempre  
    // Da usare solo in casi rari (es. Ruzzle)  
    doAlways();  
  
    // A  
    if (condizione di terminazione) {  
        doSomething;  
        return;  
    }  
  
    // Potrebbe essere anche un while ()  
    for () {  
  
        // B  
        generaNuovaSoluzioneParziale;  
  
        if (filtro) { // C  
            recursive (... , level + 1);  
        }  
  
        // D  
        backtracking;  
    }  
}
```

Figura 2. Struttura di un algoritmo ricorsivo

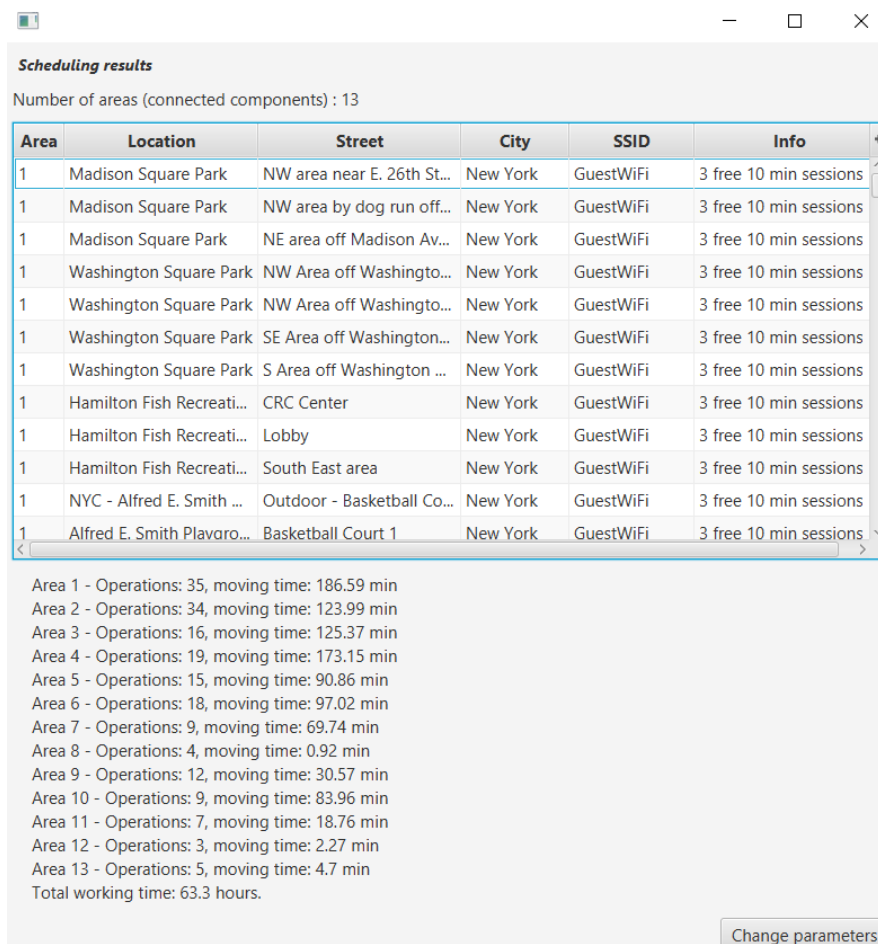
L’algoritmo ricorsivo è esatto, ma oneroso in termini di tempo in base al numero di vertici. Per ovviare a lunghe attese è presente un timer che, allo scadere, passa all’algoritmo approssimato.

L’algoritmo 2-approssimato è avviato quindi quando la ricorsione diventa dispendiosa nel tempo e non più utile o per scelta dell’utente (choice box nella schermata di avvio). I passi sono:

1. Ricercare un albero ricoprente di costo minimo (Minimum Spanning Tree) sul grafo;
2. Attraversare l’albero con una ricerca in profondità (si trova un ciclo euleriano);
3. Determinare il ciclo hamiltoniano che si ottiene visitando i nodi del grafo nell’ordine della loro prima apparizione nel ciclo euleriano.

Il primo passo utilizza la classe *KruskalMinimumSpanningTree* all'interno della libreria *jgraph*, che contiene una semplice implementazione dell'algoritmo di creazione di un MST attraverso i metodi *hashCode()* e *equals()*. Anche per il secondo passo la libreria *jgraph* possiede l'implementazione di un *Depth-first Iterator*, che lavora sul MST. Il terzo passo consiste nella costruzione effettiva del cammino, aggiungendo i vertici e calcolando il costo totale. Questo non risulta ottimo, ma è un buon compromesso tra tempo di processo e ottimalità.

I risultati della ricerca sono mostrati in una nuova finestra. In alto viene indicato il numero di aree in cui viene suddiviso il grafo creato, dunque le componenti connesse. Nella *TableView* (riquadro centrale) è indicata la sequenza di hotspots esatta elaborata dall'algoritmo, per ogni area (numerate da 1, suddivise da una riga vuota per essere più facilmente individuabili). In basso, le statistiche per ogni area: numero di hotspots da visitare ("Operations") e tempo per muoversi tra di essi ("Moving time"). Per fare una stima, è stato supposto un tempo di spostamento di 1 km ogni 10 minuti e un tempo di riparazione di 15 minuti per hotspot. E' da qui che il manager può partire per suddividere i turni dei propri operai e affidare una o più aree di intervento.



Scheduling results

Number of areas (connected components) : 13

Area	Location	Street	City	SSID	Info
1	Madison Square Park	NW area near E. 26th St...	New York	GuestWiFi	3 free 10 min sessions
1	Madison Square Park	NW area by dog run off...	New York	GuestWiFi	3 free 10 min sessions
1	Madison Square Park	NE area off Madison Av...	New York	GuestWiFi	3 free 10 min sessions
1	Washington Square Park	NW Area off Washingto...	New York	GuestWiFi	3 free 10 min sessions
1	Washington Square Park	NW Area off Washingto...	New York	GuestWiFi	3 free 10 min sessions
1	Washington Square Park	SE Area off Washington...	New York	GuestWiFi	3 free 10 min sessions
1	Washington Square Park	S Area off Washington ...	New York	GuestWiFi	3 free 10 min sessions
1	Hamilton Fish Recreati...	CRC Center	New York	GuestWiFi	3 free 10 min sessions
1	Hamilton Fish Recreati...	Lobby	New York	GuestWiFi	3 free 10 min sessions
1	Hamilton Fish Recreati...	South East area	New York	GuestWiFi	3 free 10 min sessions
1	NYC - Alfred E. Smith ...	Outdoor - Basketball Co...	New York	GuestWiFi	3 free 10 min sessions
1	Alfred E. Smith Plavoro...	Basketball Court 1	New York	GuestWiFi	3 free 10 min sessions

Area 1 - Operations: 35, moving time: 186.59 min
Area 2 - Operations: 34, moving time: 123.99 min
Area 3 - Operations: 16, moving time: 125.37 min
Area 4 - Operations: 19, moving time: 173.15 min
Area 5 - Operations: 15, moving time: 90.86 min
Area 6 - Operations: 18, moving time: 97.02 min
Area 7 - Operations: 9, moving time: 69.74 min
Area 8 - Operations: 4, moving time: 0.92 min
Area 9 - Operations: 12, moving time: 30.57 min
Area 10 - Operations: 9, moving time: 83.96 min
Area 11 - Operations: 7, moving time: 18.76 min
Area 12 - Operations: 3, moving time: 2.27 min
Area 13 - Operations: 5, moving time: 4.7 min
Total working time: 63.3 hours.

Change parameters

Figura 3. Esempio di schermata finale

Nel caso si volessero modificare i parametri, o effettuare una nuova analisi, in basso a destra il pulsante *Change parameters* permette di tornare alla schermata iniziale.

Diagramma delle classi delle parti principali dell'applicazione

Un diagramma delle classi rappresenta le entità (classi) dell'applicazione e le relazioni tra di esse.

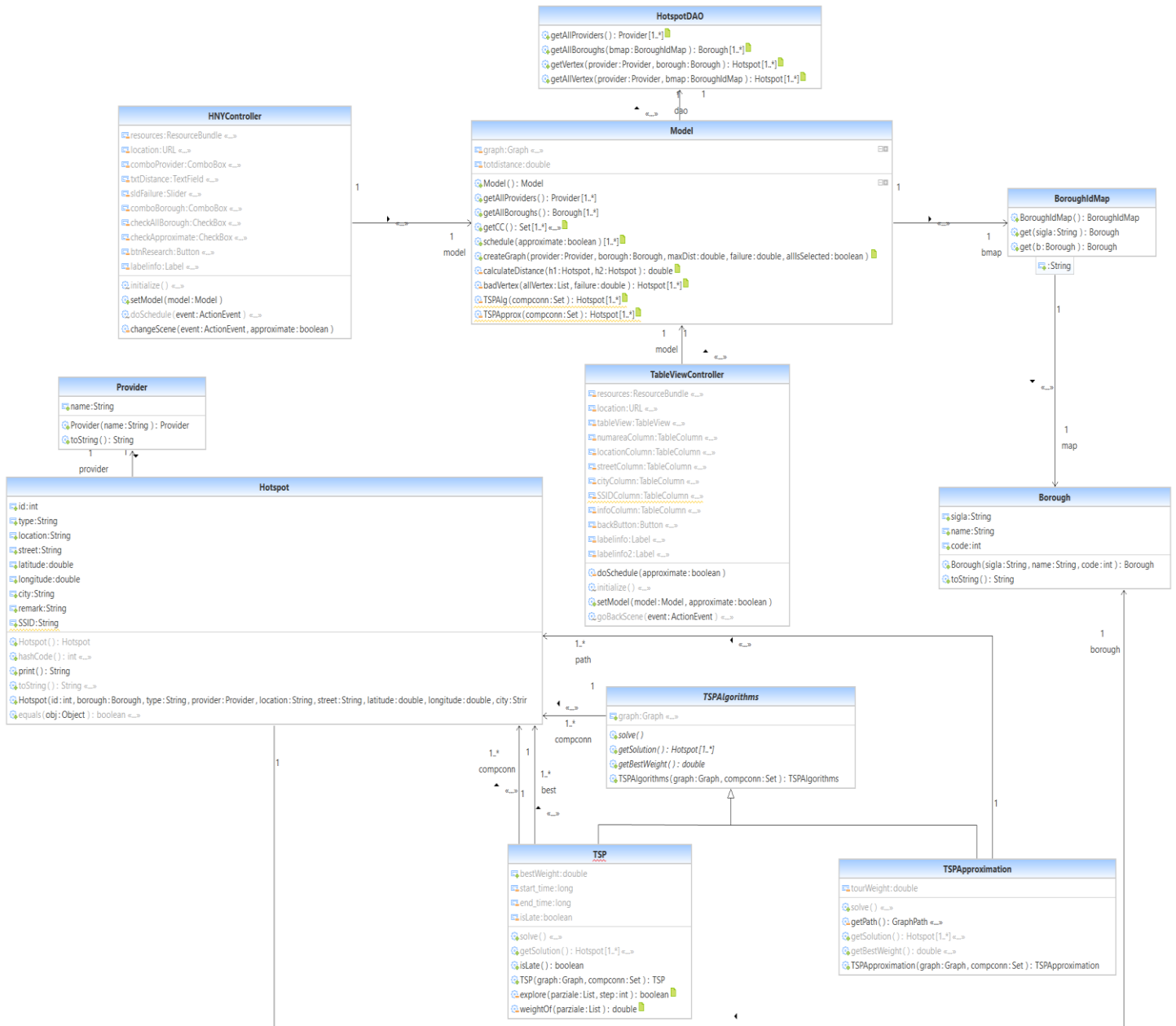


Figura 4. Diagramma delle classi

Alcune videate dell'applicazione realizzata

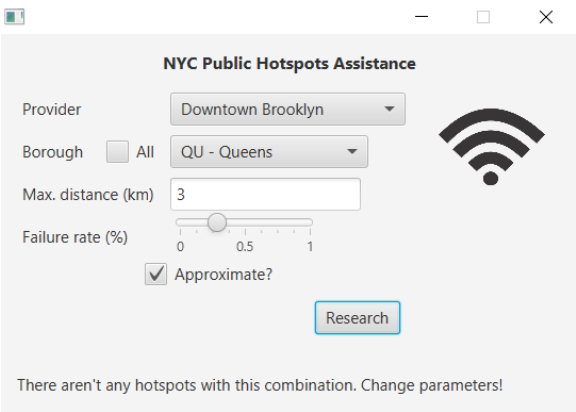


Figura 5. Esempio di segnalazione errore a causa di combinazione Provider-Borough non compatibile.

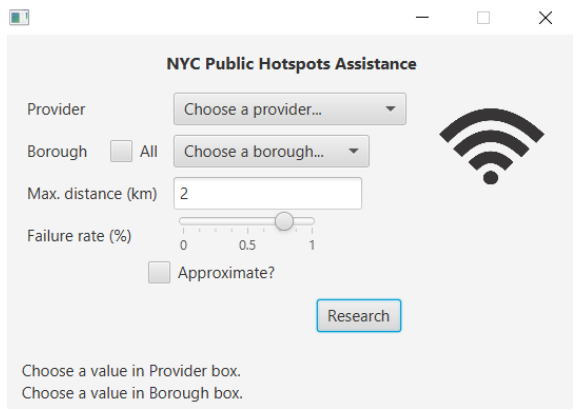


Figura 6. Esempio di segnalazione errore a causa di mancata scelta di Provider e Borough.

Scheduling results

Number of areas (connected components) : 1

Area	Location	Street	City	SSID	Info
1	Governors Island	Motor Pool Bldg. 925 R...	New York	Governors Isl...	Free - up to 5 mbs
1	Governors Island	NYC Grow	New York	Governors Isl...	Free - up to 5 mbs
1	Governors Island	Ball Field Backstop	New York	Governors Isl...	Free - up to 5 mbs
1	Governors Island	Camping	New York	Governors Isl...	Free - up to 5 mbs
1	Governors Island	Liggett F Food Court	New York	Governors Isl...	Free - up to 5 mbs
1	Governors Island	Liggett F Clayton	New York	Governors Isl...	Free - up to 5 mbs
1	Governors Island	Liggett C Clayton	New York	Governors Isl...	Free - up to 5 mbs
1	Governors Island	Liggett i Food Court	New York	Governors Isl...	Free - up to 5 mbs
1	Governors Island	Nolan Bldg 17 Rooftop	New York	Governors Isl...	Free - up to 5 mbs
1	Governors Island	Nolan Bldg 20A Back	New York	Governors Isl...	Free - up to 5 mbs
1	Governors Island	Building 108 Carder Rd	New York	Governors Isl...	Free - up to 5 mbs
1	Governors Island	Building 110 Ferrv	New York	Governors Isl...	Free - up to 5 mbs

Area 1 - Operations: 13, moving time: 32.05 min
Total working time: 4.03 hours.

Change parameters

Figure 7-8. Esempio parametri correttamente inseriti e risultato

NYC Public Hotspots Assistance

Provider: SPECTRUM

Borough: ☐ All ☒ SI - Staten Island

Max. distance (km): 2.2

Failure rate (%): ☐ Approximate?

Press button Research to schedule.

Scheduling results

Number of areas (connected components) : 5

Area	Location	Street	City	SSID	Info
1	Greenbelt Recreation ...	Parking area and ballfield	Staten Island	GuestWiFi	3 free 10 min sessions
0					
2	Midland Beach and Fra...	Along Boardwalk and B...	Staten Island	GuestWiFi	3 free 10 min sessions
2	Midland Beach and Fra...	Along Boardwalk and B...	Staten Island	GuestWiFi	3 free 10 min sessions
0					
3	Clove Lakes Park - Lak...	Front area of the Lake C...	Staten Island	GuestWiFi	3 free 10 min sessions
3	Staten Island Zoo	Area off Broadway by E...	Staten Island	GuestWiFi	3 free 10 min sessions
3	Clove Lakes Park - Lak...	Front area of the Lake C...	Staten Island	GuestWiFi	3 free 10 min sessions
0					
4	Snug Harbor Cultural ...	North area off Richmon...	Staten Island	GuestWiFi	3 free 10 min sessions
4	Snua Harbor Cultural ...	Buildina G - SE towards ...	Staten Island	GuestWiFi	3 free 10 min sessions

Area 1 - Operations: 1, moving time: 0.0 min
 Area 2 - Operations: 2, moving time: 23.51 min
 Area 3 - Operations: 2, moving time: 19.52 min
 Area 4 - Operations: 4, moving time: 5.82 min
 Area 5 - Operations: 1, moving time: 0.0 min
 Total working time: 4.06 hours.

Figure 9-10. Esempio schedulazione con algoritmo approssimato

NYC Public Hotspots Assistance

Provider: ALTICEUSA

Borough: ☐ All ☒ BX - Bronx

Max. distance (km): 1

Failure rate (%): ☐ Approximate?

Press button Research to schedule.

Scheduling results

Number of areas (connected components) : 14

Area	Location	Street	City	SSID	Info
1	Pelham Bay Park	IN PARK POLE 6/P/N/O ...	Bronx	GuestWiFi	3 free 10 min sessions
1	Pelham Bay Park	IN PARK POLE 4/P/N/O ...	Bronx	GuestWiFi	3 free 10 min sessions
1	Pelham Bay Park	C/O MIDDLETOWN RD ...	Bronx	GuestWiFi	3 free 10 min sessions
1	Pelham Bay Park	C/O MIDDLETOWN RD ...	Bronx	GuestWiFi	3 free 10 min sessions
1	Pelham Bay Park	MIDDLETOWN RD 1/P/...	Bronx	GuestWiFi	3 free 10 min sessions
1	Pelham Bay Park	C/O MIDDLETOWN RD ...	Bronx	GuestWiFi	3 free 10 min sessions
1	Pelham Bay Park	IN PARK POLE 6/P/N/O ...	Bronx	GuestWiFi	3 free 10 min sessions
0					
2	Poe Park	Playground area	Bronx	GuestWiFi	3 free 10 min sessions
2	Poe Park	left side of bandshell	Bronx	GuestWiFi	3 free 10 min sessions
2	Poe Park	right side of bandshell	Bronx	GuestWiFi	3 free 10 min sessions
2	Poe Park	North area facina Poe c...	Bronx	GuestWiFi	3 free 10 min sessions

Area 1 - Operations: 6, moving time: 11.52 min
 Area 2 - Operations: 4, moving time: 2.42 min
 Area 3 - Operations: 1, moving time: 0.0 min
 Area 4 - Operations: 2, moving time: 9.92 min
 Area 5 - Operations: 9, moving time: 15.06 min
 Area 6 - Operations: 3, moving time: 0.12 min
 Area 7 - Operations: 1, moving time: 0.0 min
 Area 8 - Operations: 6, moving time: 14.45 min
 Area 9 - Operations: 3, moving time: 0.0 min
 Area 10 - Operations: 23, moving time: 64.88 min
 Area 11 - Operations: 1, moving time: 0.0 min
 Area 12 - Operations: 8, moving time: 21.61 min
 Area 13 - Operations: 1, moving time: 0.0 min
 Area 14 - Operations: 4, moving time: 5.94 min
 Total working time: 22.43 hours.

Figure 11-12. Esempio schedulazione senza approssimazione

E' possibile trovare il video dimostrativo del software al link: <https://youtu.be/gan-8BJPSCY>

Risultati sperimentali ottenuti

Provider	Borough	Max dist. [km]	Approx.	Num. aree	Tot. ore [h]	Num. operatori/turni
ALTICEUSA	All	1	No	26	36.69	5
ALTICEUSA	All	2.5	Si	10	27.85	4
ALTICEUSA	Bronx	3	No	2	16.34	2
ALTICEUSA	Brooklin	1.5	Si	9	17.99	3
ALTICEUSA	Manhattan	4	No	/	/	/
ALTICEUSA	Queens	4	No	/	/	/
ALTICEUSA	S. Island	4	No	/	/	/
LinkNYC	All	1.3	No	9	76.82	10
LinkNYC	All	1.3	Si	7	73.97	10
LinkNYC	Queens	2	Si	2	34.76	5
LinkNYC	Brooklin	5	No	1	26.01	4
Harlem	All	4	No	1	21.32	3
Fiberless	Manhattan	3	Si	1	3.96	1
Fiberless	All	1.8	No	1	4.07	1
SPECTRUM	Bronx	2.6	No	/	/	/
SPECTRUM	Brooklin	2.6	No	3	9.57	2
SPECTRUM	S. Island	1.2	Si	6	8.69	1
SPECTRUM	All	5	Si	3	21.14	3
SPECTRUM	All	5	Si	3	26.58	4
Downtown Brooklin	Brooklin	1	No	1	13.76	2
AT&T	Queens	4	Si	2	0.5	1
AT&T	All	4	No	6	5.64	1
QPL	All	10	No	1	38.98	5
Chelsea	Manhattan	1.5	No	1	4.34	1
Manhattan Down Alliance	All	1	Si	1	5.1	1

Transit Wireless	All	10	No	1	16.19	2
------------------	-----	----	----	---	-------	---

NB. Probabilità di guasto fissa al 50%.

Conclusioni

Dalla valutazione dei risultati, si evince che il parametro “Distanza massima” è il più discriminante in termini di tempi di esecuzione dell’applicazione, connettività del grafo e onerosità dell’intervento. Infatti, come già evidenziato, il calcolo del TSP dipende dal numero di nodi e di archi del grafo, il quale è costruito sulla base dei collegamenti tra nodi, determinati proprio dalla disponibilità dell’utente a spostarsi.

Inoltre, la probabilità di guasto inserisce un’alta variabilità, poichè la scelta degli hotspots da riparare avviene con un algoritmo randomico; più è alta, maggiore è la dimensione del grafo e maggiore è la complessità. Anche il punto di partenza (e di arrivo) dell’operatore è deciso in maniera causale, non conoscendo la sede dell’azienda, ed è quindi da aggiungere per un eventuale stima completa. L’applicazione produce risultati diversi ad ogni ricerca, proprio a causa della casualità degli hotspots scelti; questa scelta consente di simulare una situazione quanto più simile alla realtà.

Anche il numero di aree di intervento è dipendente dal cambiamento della “Distanza massima”: esse altro non sono che le componenti connesse, quindi per una distanza piccola è più probabile che il grafo risulti suddiviso in più aree. E’ preferibile scegliere distanze maggiori, per impiegare meno operatori e spendere minor tempo. Per facilità, è stata scelta l’approssimazione di distanza “in linea d’aria”, e non effettiva; è possibile integrare nel database una mappa delle strade della città per effettuare una ricerca precisa.

Il parametro “Ore totali” è solo indicativo, poichè composto dalla somma del tempo di riparazione per tutti gli hotspots e dal tempo per spostarsi tra di essi; il manager calcolerà le turnazioni, piuttosto, sulla base del numero di operazioni per area e sul tempo di spostamento all’interno dell’area. Le stime dei tempi non considerano la variabilità, ma sono un valore medio. Il numero di operatori o turni di lavoro è ottenuto dividendo le ore totali per 8 ore di lavoro per turno.

L’efficienza degli algoritmi può essere ulteriormente migliorata, adottando la programmazione dinamica: il problema è suddiviso in sottoproblemi, che vengono valutati ed esclusi secondo una struttura ad albero, fino a giungere ad un’unica soluzione finale. Lo studio degli algoritmi solutori del TSP è un punto di forza, alla base di alcune applicazioni di navigazione utilizzate oggi (Google Maps).

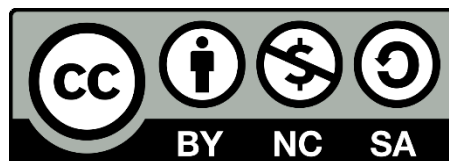
L’applicazione è il punto di partenza per sviluppare sistemi più complessi. L’obiettivo non è fornire dati precisi, ma effettuare una simulazione quanto più simile alla realtà, migliorando il servizio reso ai provider. L’azienda inoltre può decidere di quante assunzioni ha bisogno per soddisfare il carico di lavoro in ingresso.

Bibliografia

“Lucidi delle lezioni”, <https://elite.polito.it/teaching/current-courses/164-03fyz-tecn-progr?showall=&start=3>

“Algoritmo di approssimazione”,
https://it.wikipedia.org/wiki/Algoritmo_di_approssimazione

“The Trials And Tribulations Of The Traveling Salesman”, <https://medium.com/basecs/the-trials-and-tribulations-of-the-traveling-salesman-56048d6709d>



Quest'opera è distribuita con licenza Creative Commons Attribuzione - Non commerciale - Condividi allo stesso modo 4.0 Internazionale.

Copia della licenza consultabile al sito web: <http://creativecommons.org/licenses/by-nc-sa/4.0/>.