Progetto di Ricerca Operativa

BIKE SHARING

Anno accademico: 2017/2018

Studenti: Gatti Matteo (289639), Zanella Davide (293074)

Corso di Laurea in Ingegneria Informatica

Descrizione del problema

Una ditta di bike-sharing consente ai propri utenti di recuperare la bicicletta in uno dei P parcheggi di biciclette di sua proprietà e di lasciarle poi in qualsiasi posizione nell'area urbana. La ditta alla sera deve inviare un furgoncino in grado di trasportare fino a K biciclette per il recupero delle bici e la loro ricollocazione nei vari parcheggi. Si suppone di conoscere:

- il numero S delle biciclette da recuperare;
- per ogni parcheggio i, i = 1, . . . , P, il numero massimo Ri di biciclette collocabili in quel parcheggio;
- i tempi di percorrenza tra: (i) ogni coppia di bici; (ii) tra ogni bici e ogni parcheggio; (iii) tra coppie di parcheggi.

Si vuole stabilire in quale ordine il furgoncino deve visitare (e recuperare) le bici e i parcheggi e per ciascuna bici in quale parcheggio portarla, il tutto in modo tale da minimizzare il tempo complessivo di recupero di tutte le bici. Si supponga che il furgoncino parta e arrivi in un deposito per il quale sono note le distanze da tutte le bici e da tutti i parcheggi.

Modello Matematico

Insiemi definiti:

- **BICI**: insieme contenente i nodi che rappresentano le biciclette da raccogliere;
- **PARCHEGGI**: insieme contenente i nodi che rappresentano i parcheggi in cui è possibile depositare le bici;
- **DEPOSITO**: insieme contenete il nodo deposito da cui parte il furgone per la raccolta delle bici;
- **ARCHI_BICI**: insieme degli archi che specificano le distanze tra le varie bici;
- **ARCHI_PARCHEGGI**: insieme degli archi che uniscono i vari parcheggi;
- **ARCHI_BICI_PARCHEGGI**: insieme degli archi che uniscono le varie bici con ogni parcheggio;
- **ARCHI_PARCHEGGI_BICI**: insieme duale dell'insieme precedente: ogni parcheggio è collegato a tutte le biciclette;
- **ARCHI_DEPOSITO_BICI**: insieme contenente gli archi che uniscono il nodo deposito con i nodi bici;
- **ARCHI_PARCHEGGI_DEPOSITO**: insieme contenente gli archi che uniscono i nodi parcheggi con il nodo deposito;
- **NODI**: insieme contenente tutti i nodi (unione di BICI, PARCHEGGI e DEPOSITO).

Parametri definiti:

- **P**: numero di nodi di tipo parcheggio;
- **S**: numero nodi di tipo bici;
- **K**: capacità massima del furgone;
- **R{PARCHEGGI}**: capacità massima di ogni parcheggio;
- **d_bici{ARCHI_BICI}**: distanza tra ogni nodo di tipo bici (simmetrica);
- **d_parcheggi{ARCHI_PARCHEGGI}**: distanza tra ogni nodo parcheggio (simmetrica);
- **d_parcheggi_bici{ARCHI_PARCHEGGI_BICI}**: distanza tra un nodo parcheggio e un nodo bici;

- **d_bici_parcheggi{ARCHI_BICI_PARCHEGGI}**: distanza tra un nodo bici e un nodo parcheggio;
- **d_deposito_bici{ARCHI_DEPOSITO_BICI}**: distanza tra il nodo deposito e ogni nodo di tipo bici;
- **d_parcheggi_deposito{ARCHI_PARCHEGGI_DEPOSITO}**: distanza tra i nodi parcheggio e il nodo deposito;
- **n**: numero di nodi contenuto nell'insieme NODI, è il numero di nodi totali presenti nel problema.

Variabili definite:

- **b**: matrice di variabili utilizzate come flag per controllare quali archi bici-bici vengono percorsi dal furgone (in questo caso flag settato quindi variabile uguale a 1);
- **d**: matrice di variabili binarie che indicano se un arco di tipo parcheggio-bici viene percorso;
- **d2**: matrice di variabili che indica se un arco bici-parcheggio viene percorso (è il duale della variabile precedente);
- **parti**: matrice di variabili binarie che indica quale arco deposito-bici viene percorso;
- **ritorna**: matrice di variabili binarie che indica quale arco viene percorso per tornare al deposito dall'ultimo parcheggio visitato;
- **y**: vettore di variabili che servono per tenere traccia iterazione dopo iterazione della quantità di bici caricate sul furgone;
- **x**: vettore di variabili che indica l'ordine in cui vengono visitati i nodi dal furgone.

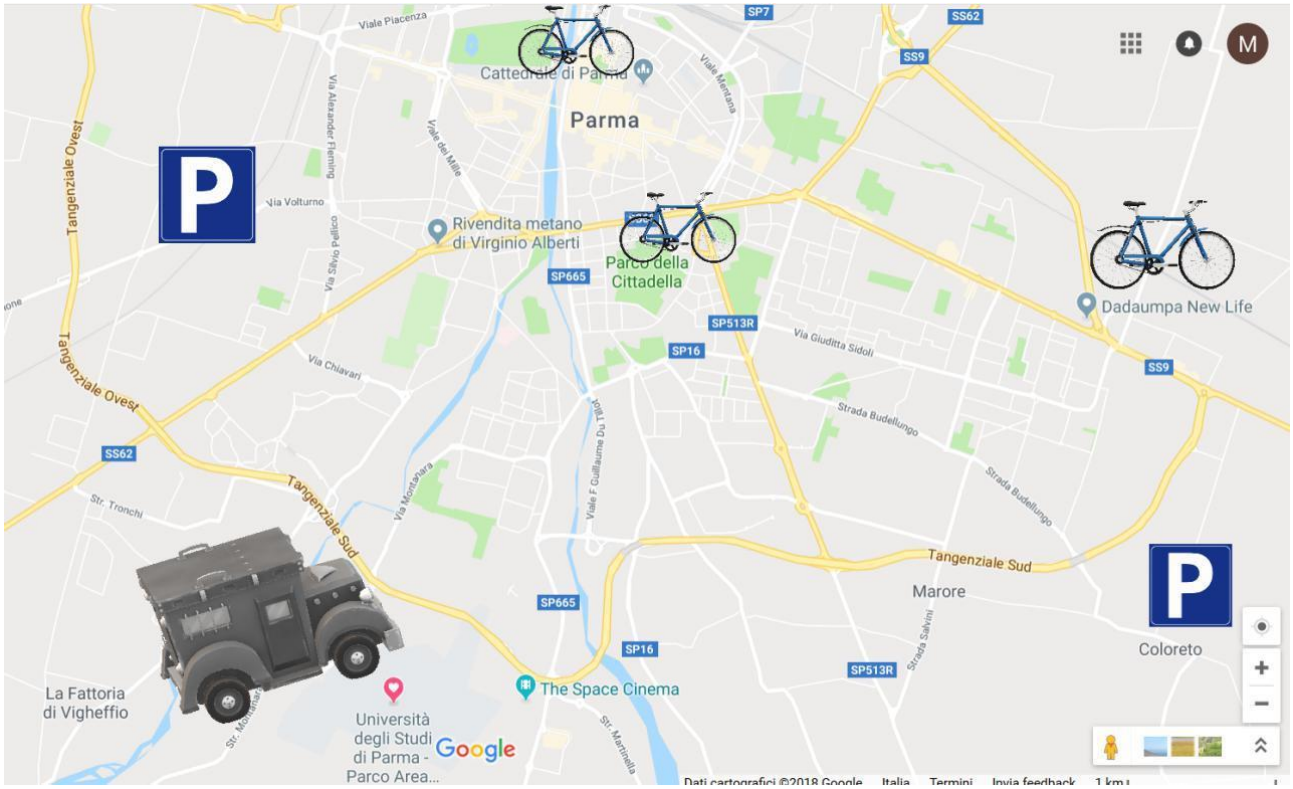
Sono stati definiti inoltre diversi vincoli con i seguenti scopi:

- **partenza**: impone al furgone di percorrere un solo arco deposito-bici;
- **ritorno**: attraverso questo vincolo imponiamo che una volta ultimato il deposito delle biciclette il furgone debba percorrere un solo arco che porti dall'ultimo parcheggio al deposito;
- **archi_ingresso_bici & archi_uscita_bici**: imponiamo che il furgone possa transitare una sola volta da un nodo bici;

- **capacita_parcheggi**: imponiamo che non venga sforata la capacità del parcheggio. Per semplicità abbiamo imposto che ogni parcheggio sia formato da un numero di sotto-parcheggi (aventi dimensione 1) pari alla sua capacità complessiva;
- **archi_ingresso_parcheggi**: controlla che in ogni sotto-parcheggio possa esservi una e una sola bicicletta. Il furgone, inoltre, non deve passare nuovamente nei sotto-parcheggi ove ha già depositato delle biciclette;
- **raccolta_bici**: impone al furgone di raccogliere tutte le biciclette prima di tornare al deposito;
- **capacita_temporale**: con questa serie di vincoli si riesce a tenere controllato nel tempo il numero di bici presenti nel furgone, questa variabile verrà quindi incrementata di 1 ogni volta che visiterà un nodo bicicletta e decrementata di 1 ogni volta che il furgone transiterà in un nodo parcheggio;
- **parcheggi_vuoti**: tramite questo vincolo imponiamo che la capacità del furgone sia pari a 0 nei parcheggi in cui questo non transita, si è reso necessario aggiungere questo vincolo dal momento che nel corso dell'esecuzione di varie prove abbiamo notato che per alcuni parcheggi la variabile y assumeva valori diversi da 0 nonostante il furgone non transitasse attraverso questi nodi;
- **scansione_temporale**: questa serie di vincoli impedisce al problema di creare sotto-cicli imponendo che il valore della variabile venga incrementato di uno ogni volta che transita attraverso un qualsiasi nodo partendo dal valore 0 corrispondente al nodo di deposito fino ad arrivare al massimo al valore n-1.

Esempi e discussione

Esempio 1 (file bikereal.dat):
Abbiamo ideato un piccolo esempio reale. Come si evince dall'immagine, il furgoncino parte dal campus, più precisamente dal deposito sul retro della palazzina1, e deve recuperare tre biciclette lasciate rispettivamente nei pressi della Cattedrale, del parco Cittadella e della discoteca Dadaumpa. I parcheggi sono situati uno in via Volturno e uno a Coloreto. Entrambi i parcheggi possono ospitare al massimo due bici mentre il furgone può ospitare contemporaneamente fino a 3 bici.



Con l'ausilio di Google Maps abbiamo calcolato le **distanze reali** tra il deposito e i parcheggi, tra il deposito e le bici, tra le varie bici, tra i due parcheggi e tra i parcheggi e le bici.

Il furgoncino riesce a recuperare le bici e a ritornare in Università percorrendo solo 25km.
Questo il percorso:

1. Dall'università si dirige verso la bici al Parco Cittadella
2. Dal parco va a depositare la bici a Coloreto

3. Da Coloreto recupera la bici abbandonata al Dadaumpa
4. Dal Dadaumpa va a recuperare la bici posta nei pressi della Cattedrale
5. Dalla Cattedrale va al parcheggio in Via Volturno, deposita entrambe le bici e poi ritorna in Università

Esempio 2 (file bike.dat):
In questo secondo esempio abbiamo 3 parcheggi ognuno con due posti liberi e un totale di 4 bici da recuperare. Le distanze sono completamente casuali. Ogni parcheggio è stato quindi diviso in due sotto-parcheggi di dimensione 1 a distanza nulla tra loro. Abbiamo utilizzato questo esempio con dimensioni ridotte per poter controllare il risultato durante le diverse prove fate per verificare la correttezza dell'esecuzione.

Analisi di sensitività

Modificando alcuni parametri del problema (come ad esempio la capacità del furgone, la distanza tra i diversi tipi di nodi, etc.) cambia ovviamente anche la soluzione.

Considerando il primo esempio, se diminuiamo la capacità del parcheggio in via Volturno da due a uno la distanza percorsa aumenta da 25km a 26km. Il percorso del nostro furgone diventa il seguente:

1. Dall'università si dirige verso la bici alla Cattedrale
2. Dalla Cattedrale il furgone va a recuperare la bici al Dadaumpa
3. Il furgone va a depositare le due bici nel parcheggio a Coloreto
4. Il furgone recupera la bici al parco Cittadella e la porta in Via volturno
5. Infine ritorna alla base

Torniamo alla situazione di partenza e supponiamo che il furgone sia piccolo e abbia un solo posto libero per caricare delle bici. In questo caso il percorso si allunga sensibilmente e diventa pari a 31 (+6km).

Conclusioni

L’algoritmo si comporta in modo corretto sia in un esempio con valori dei parametri casuali sia considerando un esempio realistico. In questo secondo caso è interessante vedere come il cambiamento di alcuni parametri possa cambiare la soluzione avendo più o meno posti in un certo parcheggio o avendo un furgone di capacità meno elevata.

Un’osservazione importante da fare è che al crescere del numero di bici o parcheggi la quantità di dati da definire nel file .dat cresce molto rapidamente. Questo è dovuto al fatto che è necessario definire sia gli archi da un nodo x a un nodo y sia gli archi duali. Ad esempio bisogna definire gli archi da ogni bici a ogni parcheggio e gli archi inversi da ogni parcheggio a ogni bici. La crescita esponenziale dei dati è anche dovuta alla semplificazione che abbiamo fatto per cui ogni parcheggio P è in realtà composto da tanti sotto-parcheggi a capacità uno. Pertanto se un generico parcheggio ha 30 posti liberi è necessario definire 30 sotto-parcheggi p1,1…p1,30 e collegarli tra loro (distanza nulla), collegarli alle varie biciclette, agli altri sotto-parcheggi e al deposito.

Per verificare la correttezza del modello abbiamo svolto diverse prove ottenendo sempre un risultato positivo. Ad esempio abbiamo azzerato la distanza tra le bici per verificare che le raccogliesse tutte subito. Abbiamo calcolato il percorso minimo “su carta” e poi aumentato sensibilmente una delle distanze per verificare che il modello andasse a ricalcolare un secondo percorso ottimo tenendo conto della modifica fatta.