UNIVERSITÀ
DI PARMA

Ricollocazione nel bike-sharing

Matteo Gianvenuti

Donato Bruno

Anno accademico 2023/2024

Sommario

Problema

Ricollocazione nel bike-sharing

Un'azienda che gestisce il bike-sharing ha il problema di ricollocare le bici ogni sera negli stalli dove si ha maggiore richiesta. In pratica ci sono N stalli, ogni stallo i ha un certo numero c_i di bici parcheggiate in esso alla sera e ne vuole s_i parcheggiate nello stesso alla mattina. Il recupero delle bici avviene attraverso un furgone che ha una capacità massima di K. Sono poi noti i tempi di percorrenza $t_{i,j}$ tra le diverse coppie $i, j \in N$ di stalli. Il problema da risolvere consiste nel cercare di pianificare le operazioni di ricollocazione delle bici in un tempo minimo.
Si formuli un modello matematico di questo problema, lo si traduca in AMPL e si risolva e commenti una particolare istanza del problema.

Modello matematico

1. Dati

- N = Numero di stalli.
- NODI = Insieme degli stalli.
- ARCHI = insieme degli archi (strade tra gli stalli).
- K = Capacità del furgone.
- c_i con i ∈ NODI = Numero di bici presenti nell'i-esimo stallo la sera.
- s_i con i ∈ NODI = Numero di bici volute nello stallo i-esimo al mattino.
- $t_{i,j}$ con i, j ∈ NODI con i ≠ j = Tempo di percorrenza da i a j.
- $pc_i := c_i - s_i$ con i ∈ NODI = Indica se nello stallo ci sono troppe bici ($pc_i > 0$) oppure se non ce ne sono abbastanza ($pc_i < 0$). Se è zero le bici sono già a posto, ed i sarà un nodo di transito (il circuito hamiltoniano deve passare da tutti i nodi).

2. Variabili decisionali

- $x_{i,j} \in \{0, 1\}$ ∀ i, j ∈ NODI con i ≠ j = Indica se l'arco i, j fa parte del circuito hamiltoniano.
- $y_i \in \{0, 1, \ldots, N-1\}$ ∀ i ∈ NODI = Indica la posizione del nodo i-esimo nella sequenza di visita dei nodi.
- $f_i \geq 0, \leq K$ ∀ i ∈ NODI = Indica la capacità del furgone nel nodo i-esimo.

3. Vincoli

- Ogni nodo ha un solo arco in ingresso, ovvero $\sum_{i \in NODI, i \neq j} x_{i,j} = 1 \quad \forall j \in NODI$
- Ogni nodo ha un solo arco in uscita, ovvero $\sum_{i \in NODI, i \neq j} x_{j,i} = 1 \quad \forall j \in NODI$

- Non sono presenti sotto-circuiti, ovvero $y_j - y_i \geq x_{i,j} + (1 - N)(1 - x_{i,j}) \quad \forall\, i \neq j, j \neq 1$
- La y del primo nodo è fissata a 0, ovvero $y_1 = 0$
- La f del primo nodo è fissata a pc del primo nodo se pc positivo altrimenti a 0, ovvero $f_1 = pc_1$ se $pc_1 > 0$ altrimenti 0
- Se $x_{i,j} = 1 \Rightarrow f_j - f_i = pc_j$ inevce se $x_{i,j} = 0 \Rightarrow f_j - f_i \in [-K, K]$, ovvero $f_j - f_i \geq pc_j x_{i,j} + (1 - x_{i,j})(-K)$ e $f_j - f_i \leq pc_j x_{i,j} + (1 - x_{i,j})K$

4. Obbiettivo

- Il tempo totale deve essere minimo, ovvero $\min \sum_{i \neq j} t_{i,j} x_{i,j}$

Modello matematico completo

$\min \sum_{i \neq j} t_{i,j} x_{i,j}$	$\forall\, i, j \in NODI, i \neq j$
$\sum_{i \in NODI, i \neq j} x_{i,j} = 1$	$\forall\, j \in NODI$
$\sum_{i \in NODI, i \neq j} x_{j,i} = 1$	$\forall\, j \in NODI$
$y_j - y_i \geq x_{i,j} + (1 - N)(1 - x_{i,j})$	$\forall\, i \neq j, j \neq 1$
$x_{i,j} \in \{0, 1\}$	$\forall\, i, j \in NODI, i \neq j$
$1 \leq y_i \leq N - 1$	$i = 2, \ldots, N$
$y_1 = 0$	
$f_j - f_i \geq pc_j x_{i,j} + (1 - x_{i,j})(-K)$	$\forall\, i, j \in NODI, i \neq j$
$f_j - f_i \leq pc_j x_{i,j} + (1 - x_{i,j})K$	$\forall\, i, j \in NODI, i \neq j$
$0 \leq f_i \leq K$	$\forall\, i \in NODI$

AMPL

1. Dati

```
### INSIEMI ###--------------------------------------------------------------------------

set NODI ordered;
set ARCHI := ( NODI cross NODI );


### PARAMETRI ###-----------------------------------------------------------------------

param n := card(NODI);
param t{ARCHI} default 100000000;
param c{NODI};
param s{NODI};
param pc{i in NODI} := c[i] - s[i];
param k;

```

2. Variabili decisionali

```
### VARIABILI ###-----------------------------------------------------------------------

var x{ARCHI} binary;
var y{NODI} >= 0, <= n-1, integer;
var f{NODI} >= 0, <= k, integer; # capacita del furgone nel nodo i

```

3. Vincoli

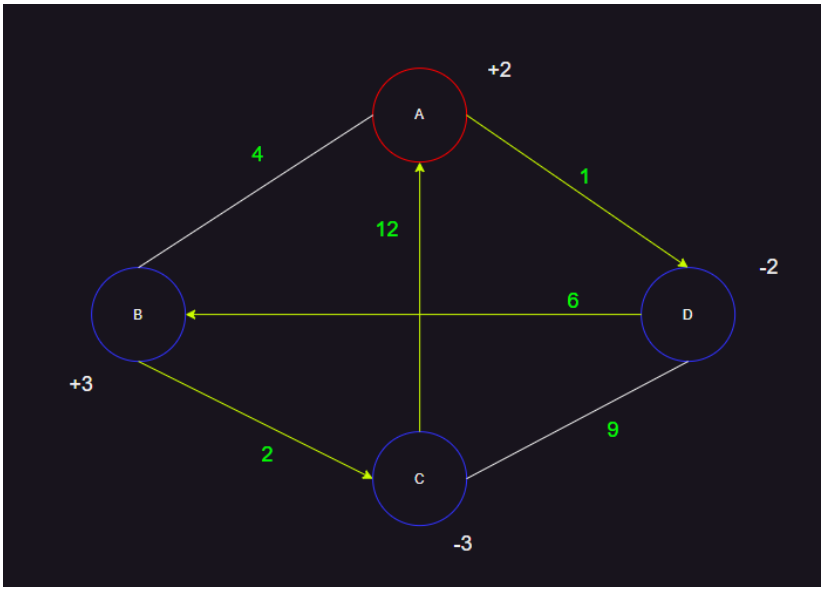
```
### VINCOLI ###-------------------------------------------------------------------------

subject to ingresso{i in NODI} : sum{j in NODI :  (j,i) in ARCHI}
x[j,i] = 1;
subject to uscita{i in NODI} : sum{j in NODI : (i,j) in ARCHI}
x[i,j] = 1;
subject to sequenza{(i,j) in ARCHI : j != first(NODI)} :
y[j]-y[i] >= n*x[i,j]+1-n ;
subject to nodo_partenza : y[first(NODI)]=0;

subject to flusso_1{(i, j) in ARCHI : i != j} :
(f[j] - f[i]) >= (pc[j]*x[i,j] + (1 - x[i,j])*(-k));
subject to flusso_12{(i, j) in ARCHI : i != j} :
(f[j] - f[i]) <= (pc[j]*x[i,j] + (1 - x[i,j])*k);
subject to start : f[first(NODI)] = if(pc[first(NODI)] > 0) then pc[first(NODI)] else 0;

```

4. Obbiettivo

```
41  ### OBIETTIVO ###-----------------------------------------------------------------------------
42
43  minimize distanza_totale : sum{(i,j) in ARCHI}
44  t[i,j]*x[i,j];
45
46
```

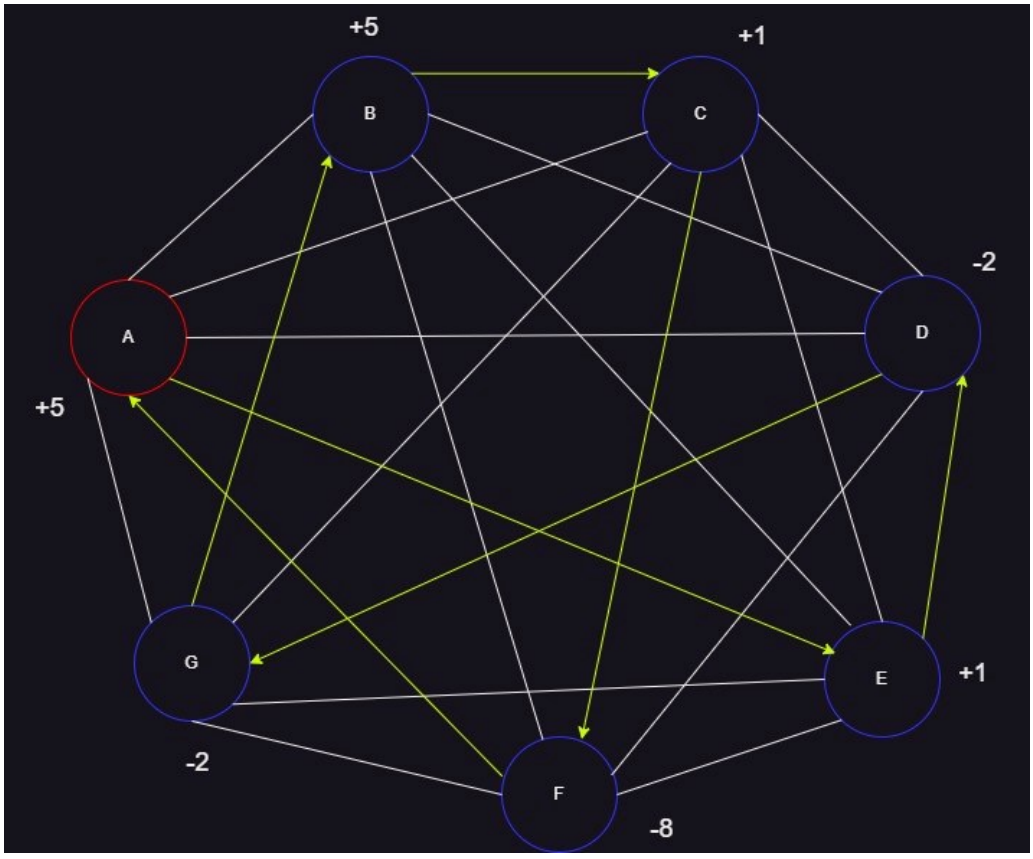
Esempi

1. Un esempio semplice



2. Un esempio più complesso

Un esempio più interessante si può ottenere con sette nodi.



```
t [*,*]
:     a       b       c       d       e       f       g      :=
a   1e+08       1       5       6       1       2       6
b       1   1e+08       3       7       7       5       4
c       5       3   1e+08       8       8       3       5
d       6       7       8   1e+08       2       5       1
e       1       7       8       2   1e+08       6       7
f       2       5       3       5       6   1e+08      11
g       6       4       5       1       7      11   1e+08
;
```

La strada ottimale è:
A -> E -> D -> G -> B -> C -> F -> A

Tutte le altre:
A -> (C, E) -> (G, D) -> B -> F -> A
A -> (D, E) -> (G, C) -> B -> F -> A
A -> (G, E) -> (D, C) -> B-> F -> A
A -> (G, D) -> (B, C, E) -> F -> A
A -> D -> B -> F -> (G, C, E) -> A
A -> G -> B -> F -> (D, C, E) -> A
A -> C -> (D, G) -> B -> E -> F -> A
A -> E -> G -> D -> B -> F -> A

Dove (x1, x2, x3) indica tutti i possibili percorsi attraverso i nodi x1, x2 e x3 (permutazioni).

Risultati del modello in AMPL

```
ampl: include TSPb.run;
Gurobi 4.0.1: optimal solution; objective 16
12 simplex iterations
x [*,*]
:   a   b   c   d   e   f   g    :=
a   0   0   0   0   1   0   0
b   0   0   1   0   0   0   0
c   0   0   0   0   0   1   0
d   0   0   0   0   0   0   1
e   0   0   0   1   0   0   0
f   1   0   0   0   0   0   0
g   0   1   0   0   0   0   0
;
```

```
y [*] :=
a  0
b  4
c  5
d  2
e  1
f  6
g  3
;
```

```
pc [*] :=
a   5
b   5
c   1
d  -2
e   1
f  -8
g  -2
;
```

```
f [*] :=
a  5
b  7
c  8
d  4
e  6
f  0
g  2
;
```

Conclusione

Per semplicità abbiamo utilizzato solo grafi non orientati negli esempi ma è chiaro che il modello è generico e può essere utilizzato anche su grafi orientati purché esista un circuito hamiltoniano ammissibile per il problema e la somma dei pc_i sia zero.