MDMM Project

Matino Ruben
Ritacco Antonio

June 2020

1 Problema (N. 4)

A shipping company has to send a set of orders over a multi-modal transportation network (trains + trucks), represented as a directed graph where each arc has a unit transportation cost and a maximum capacity (total amount of goods that can travel along that arc). Train arcs have far lower transportation cost, have stricter capacity (railways are few w.r.t. roads) and are "longer" (only reach a subset of nodes), while truck arcs have a higher transportation cost but larger capacity and connect all nodes. Each order is represented by a given amount of goods, a starting node and an ending node. To avoid possible misservices to users, all the goods pertaining to the same order have to be shipped together, i.e., along a single path; furthermore, the length of the path must not exceed a prescribed constant k (nor very small, but not very large either). The problem is to find a minimum-cost set of paths with the required property that allow to satisfy all the orders while satisfying the arc capacity constraints.

2 Modello matematico

Il modello matematico che ci è sembrato più adeguato è il Minimum Cost Multicommodity Flow adattandolo al nostro caso ed assumendo che dato un arco esso possa essere o solo di tipo stradale o solo di tipo ferroviario. Inoltre per soddisfare l'unicità dei cammini abbiamo aggiunto un vincolo (6) in cui imponiamo che dato un ordine, ogni nodo appartenente a quel cammino possa avere al più un arco uscente.

2.1 Parametri

- N è l'insieme dei nodi.
- A è l'insieme degli archi.
- H è l'insieme degli ordini
- u_{ij} capacità del collegamento $(i, j) > 0$

- c_{ij} costo del collegamento (i, j)
- d_h quantita' merce relativa ad ordine h

2.2 Variabili Decisionali

- $y_{ij}^h \in \{0, 1\}$ arco di attivazione del collegamento (i, j) relativa all'ordine h

2.3 Modello

$$min(\sum_{(i,j)\in A}\sum_{h=1}^{H} c_{ij} \cdot y_{ij}^h) \tag{1}$$

$$(\sum_j y_{ji}^h) - (\sum_j y_{ij}^h) = b_i^h, \ \ \forall i \in N, h = 1, ...H \tag{2}$$

$$\sum_{(h=1)}^{H} y_{ij}^h \cdot d^h \leq u_{ij}, \ \ \forall (i, j) \in A \tag{3}$$

$$\sum_{(i,j)} y_{ij}^h \leq K, \ \ h = 1, ...H \tag{4}$$

$$\sum_j y_{ij}^h \leq 1, \ \ \forall i \in N, h = 1, ...H \tag{5}$$

$$y_{ij}^h \in \{0, 1\}, \ \ \forall (i, j) \in A, h = 1, ...H \tag{6}$$

- In (1) è presentata la funzione obiettivo. Questa consiste nella minimizzazione, per ogni ordine h, del costo di instradare la merce lungo una tratta composta possibilmente da ferrovie e/o strade.
- In (2) è presentato il vincolo di bilancio di flusso per ogni ordine $h \in H$.

$$b_i = \begin{cases} -1, i = origine \\ 1, i = destinazione \\ 0, altrimenti \end{cases}$$

- Le disequazioni (3) rappresenta i vincoli sulla capacita' degli archi.
- La disequazione (4) modella il vincolo di lunghezza massima del cammino $\leq K$.
- Infine, la disequazione (5) modella il vincolo di unicita' del cammino per ogni commodity.
- In (6) è invece presente il vincolo di interezza per le attivazioni degli archi.

3 Generatore di istanze

La funzione *instance_generator* all'interno di **instance.py** viene utilizzata per la generazione dei files .cmpl mentre la funzione *generator* all'interno dello script file **generator.py** si occupa della generazione dei parametri necessari all'istanza per essere creata generando direttamente il file .cmpl per ogni istanza del problema, con annesso modello matematico. Ogni istanza generata è infatti pronta per essere data in pasto ad un solutore che accetti istanze in formato .cmpl. I nostri test presentati nel paragrafo successivo sono stati effettuati utilizzando i solver *glpk* e *cbc*. In questa sezione ci proponiamo di svolgere, quindi, i punti 2 e 3 del progetto.

3.1 Istance.py

Per ottenere le istanze del problema abbiamo deciso di implementare, utilizzando il linguaggio python, un generatore di file .cmpl che tenesse conto delle specifiche del problema per generare delle istanze realistiche. Il problema richiede infatti, che le strade colleghino ogni coppia di nodi e che le ferrovie siano presenti solo sui collegamenti "lunghi". Il nostro generatore, data una coppia (i, j) crea un arco ferroviario tra i e j solo quando la seguente condizione è verificata:

$$random.uniform[0,1] < \frac{|i-j|}{Nodes-1} \cdot perc\_att\_ferrovie \tag{7}$$

Dove, Nodes è il numero di nodi e $perc\_att\_ferrovie$ è un parametro scelto dall'utente in fase di creazione dell'istanza e compreso tra 0 ed 1. In questo modo, la probabilità che venga a crearsi un collegamento ferroviario tra i e j è tanto più alta quanto più i due nodi sono lontani. La probabilità è uguale ad 1 quando i due nodi sono il primo (1) e l'ultimo (Nodes). L'utente tramite il paramentro $perc\_att\_ferrovie$ è in grado di controllare ulteriormente il numero di ferrovie che possono venire a crearsi.
Il nostro generatore, genera quindi per ogni coppia di nodi, una strada e con probabilità variabile una ferrovia. Per scoraggiare l'utilizzo indiscriminato di strade su collegamenti lunghi (visto che le specifiche del problema indicavano che ogni coppia di nodi dovesse essere collegata almeno da una strada), abbiamo moltiplicato il costo del collegamento (stradale o ferroviario) per $|i-j|$ così che utilizzare più collegamenti, ma corti, abbia un costo generalmente inferiore.
Visto che le specifiche richiedevano che i costi e le capacità delle ferrovie fossero minori delle strade, il generatore di istanze genera i costi e le capacità in maniera randomica ma all'interno di intervalli predefiniti, così da impedire che esista un collegamento stradale tra due nodi con un costo (capacità) minore della corrispettiva ferrovia (se creata).
Altro parametro a cui abbiamo fatto attenzione è K ossia il numero massimo di collegamenti servibili in un solo percorso. Nel nostro generatore K è generato 4 volte per ogni istanza tenendo conto del numero di nodi e facendo in modo di avere K crescenti a partire da valori bassi sino a valori leggermente più alti. In questo modo abbiamo potuto osservare come per una stessa istanza, K

influezzasse il risultato in termini di costo minimo e tempo di esecuzione.
Infine, il numero di ordini H è generato come : $\lceil \frac{Nodes}{2} \rceil$

3.2 Generator.py

Per generare una sequenza di istanze .cmpl basta lanciare lo script generator.py con i seguenti parametri:

- -nodes : Numero di nodi della rete, default [10,50,100]
- -patt : percentuale di attivazione delle ferrovie, default 0.9

Non viene esposto all'utente la possibilità di cambiare i costi e e le capacità minime e massime, così come la demand. Dopo alcuni tests ci è sembrato ragionevole utilizzare i seguenti valori di min e max come range per la generazione di valori random:

- min_demand : 50
- max_demand : 100
- min_costs_strade : 50
- max_costs_strade : 75
- min_costs_ferrovie : 25
- max_costs_ferrovie : 45

Riguardo le capacità delle ferrovie abbiamo scelto di utilizzare il valore *max_demand* come **capacità minima ferroviaria** (così da non rischiare la creazione di ferrovie potenzialmente non utilizzabili per nessun ordine), mentre per la **capacità massima ferroviaria** abbiamo moltiplicato la capacità minima per 1.5. Riguardo le strade, la **capacità minima stradale** è ottenuta sommando 1 alla capacità massima delle ferrovie, mentre la **capacità massima stradale** è ottenuta raddoppiando la capacità minima.

4 Implementazione e risultati

Per testare come il generatore crei istanze effettivamente più difficili da risolvere al variare dei parametri esposti all'utente, abbiamo lanciato una serie di esperimenti con i due solver : *cbc* e *glpk* ed usando i seguenti parametri del generatore:

- Numero di nodi : 10, 50, 100
- Percentuale di attivazione delle ferrovie : 0.1, 0.5, 0.9

In totale abbiamo generato 90 diverse istanze .cmpl (per la stessa combinazione di $Nodes$ e $perc\_att\_ferr$ ci sono 3 o 4 istanze con gli stessi dati di input e solo K diverso i cui risultati (completi) sono mostrati in section 8. Per ogni K, per ogni $\#N$ (numero di nodi) e per ogni $\%Ferr$ (percentuale di attivazione delle ferrovie) abbiamo generato 3 diverse istanze in modo tale da avere una stima di quanto in media, il solutore performa su una determinata tipologia di istanza di problema. In generale, abbiamo notato che utilizzando il solver *glpk* la soluzione ottima viene trovata in un tempo inferiore rispetto a quella trovata con il solver *cbc*, fatta eccezione per casi sporadici. Come tempo massimo per la risoluzione del problema abbiamo utilizzato 300 secondi e le caratteristiche del pc sul quale abbiamo lanciato le esecuzioni sono le seguenti:

- S.O. : Windows 10
- Processore : intel core i7-7700HQ 4 core (8 Threads)
- Ram : 16gb ddr4
- S.video : gtx 1060m

Gli script python utilizzati per la creazione delle istanze e per la loro valuzione sono tutti nella cartella *src/*. I dati generati in formato *.cmpl* e le soluzioni salvate in formato *.xml* sono invece nella cartella *dati/*.
La percentuale di attivazione delle ferrovie influenza molto la facilità con la quale è trovata la soluzione ottima. Con 100 nodi e percentuale di attivazione = 0.1 come visibile in table 1, nessuno dei due solvers riesce a trovare la soluzione ottima in un tempo ragionevole tranne che per il caso con $k = 2$ e come solver *glpk*, questo perchè un K così basso rispetto al numero di possibili collegamenti riduce di molto lo spazio di ricerca. Con una percentuale di attivazione delle ferrovie = 0.5 e = 0.9 le istanze da 100 nodi, vengono invece solo risolte da *glpk* come visibile in table 2 e table 3 e con tempi generalmente inferiori. Il parametro k che controlla il numero massimo di archi attivabili per un singolo ordine, abbiamo notato che non influisce molto una volta che diventa sufficientemente grande. I miglioramenti più importanti in termini di costo minimo li abbiamo verificati quando si passava da $k = 2$ a $k = 6$. Probabilmente aumentando la distanza minima tra i nodi sorgente e nodi destinazione avremmo potuto vedere miglioramenti più consistenti in termini di costo minimo all'aumentare di k, ma al momento la generazione dei bilanci degli ordini è lasciata random.
Per quanto riguarda le esecuzioni fallite con 100 nodi (table 1), riducendo il numero di collegamenti stradali ,passando quindi da un grafo completo ad uno sparso (impostando la probabilità di creare un collegamento stradale da 1 ad 0.1), il solver *glpk* riesce in poco tempo a trovare la soluzione ottima. Non abbiamo però incluso questa serie di esperimenti, poichè le istanze non sarebbero state aderenti alla formulazione del problema.

Osservando le tabelle è possibile notare un notevole incremento del tempo di calcolo al crescere delle dimensioni del problema. Per problemi di piccole dimensioni il solver si comporta bene ottenendo soluzioni ottime in tempi accettabili

(meno di 300 secondi); tuttavia aumentando le dimensioni del problema, ed in particolare utilizzando 100 nodi, il tempo di calcolo cresce in maniera evidente, impendendo al solutore di trovare soluzioni ottime restando nel time-limit. Il gap ottenuto nelle istanze non risolte ottimamente è particolarmente elevato quando le istanze sono caratterizzate da un basso numero di collegamenti ferroviari.

5 Riformulazione e Risultati : Alzare il lower bound

Per tentare di fornire un lower-bound migliore (più alto), abbiamo rilassato la formulazione del problema spostando i vincoli di capacità nella funzione obiettivo. La soluzione ottima del problema rilassato è un bound duale al valore ottimo del problema originale. Il duale Lagrangeano del nostro modello è definito a partire dal rilassamento Lagrangeano dei vincoli complicanti

$$\sum_{(h=1)}^{H} y_{ij}^h \cdot d^h \leq u_{ij}, \ \ \forall (i,j) \in A \tag{8}$$

ovvero:

$$\phi(w) = min(\sum_{(i,j)\in A} \sum_{h=1}^{H} c_{ij} \cdot y_{ij}^h + \sum_{(i,j)\in A} w_{ij} \cdot (\sum_{h=1}^{H} y_{ij}^h \cdot d^h - u_{ij}) \tag{9}$$

$$(\sum_j y_{ji}^h) - (\sum_j y_{ij}^h) = b_i^h, \ \ \forall i \in N, h = 1, ...H \tag{10}$$

$$\sum_{(i,j)} y_{ij}^h \leq K, \ \ h = 1, ...H \tag{11}$$

$$\sum_j y_{ij}^h \leq 1, \ \ \forall i \in N, h = 1, ...H \tag{12}$$

$$y_{ij}^h \in \{0,1\}, \ \ \forall (i,j) \in A, h = 1, ...H \tag{13}$$

dove le variabili w_{ij} sono i moltiplicatori lagrangeani (non negativi) associati ai vincoli rilassati. Se z è il valore della funzione obiettivo del nostro modello all'ottimo, vale la relazione:

$$\phi^*(w) \leq z \ \ \forall w \geq 0 \tag{14}$$

dove $\phi^*(w)$ è la soluzione ottima del lagrangeano.
Nel nostro caso non abbiamo utilizzato algoritmi di ottimizzazione per i moltiplicatori lagrangeani, ma li abbiamo scelti provandone solo alcuni valori ed in modo che fossero costanti per tutti gli archi. In particolare, i valori testati nel rilassamento sono stati: 1, 0.5, 0.4 . Questa scelta è stata guidata dal fatto che le capacità e i costi degli archi per le istanze sono tutte dello stesso ordine

di grandezza (per archi della stessa lunghezza). Lo script *lagrange.py*, legge la soluzione del problema duale in .xml e la sostituisce nella formulazione originale del problema utilizzando il corrispondente file .cmpl.

Come è possibile vedere in table 4 risolvendo il duale lagrangeano si trovano lower-bound con valori maggiori rispetto a quelli trovati utilizzando la formulazione originale (che viene rilassata trasformando i vincoli binari in vincoli reali). Per la maggior parte delle istanze dove il gap era abbastanza alto (>3%) lo stesso è stato ridotto notevolmente mentre per le istanze con gap già abbastanza piccolo il guadagno non è stato molto importante. In particolare il gap era già abbastanza piccolo nella versione originale in tutte le istanze con una percentuale di attivazione delle ferrovie elevata. Riguardo al parametro K abbiamo notato che nelle istanze dove la percentuale di attivazione delle ferrovie era piccola, usare un K troppo grande ha effetti negativi sulla risoluzione del problema in quanto lo spazio di ricerca viene inutilmente ingrandito. Questo fatto è confermato anche dalle soluzioni trovate nel problema duale che oltre un certo K smettono di scendere nei valori. Lo script *eval_ activation.py* prende in input una serie di soluzioni in formato .xml e ne estrae il numero di ferrovie e strade attivate così come il numero di ordini da soddisfare. Nelle istanze dove la riduzione del gap è stata importante (prime quattro istanze in table 4) il numero di ferrovie attivate dalle soluzioni rilassate è leggermente superiore a quelle utilizzate nelle soluzioni originali, mentre il numero di strade attivate viene quasi dimezzato. Questo accade perchè nel problema rilassato è possibile sforare i vincoli di capacità e quindi utilizzare da più ordini lo stesso collegamento ferroviario per poi smistare gli ordini utilizzando poche strade che collegano i nodi destinazione con i nodi raggiunti da ferrovie.
Quindi al fine di ridurre i costi totali di gestione degli ordini è molto importante aumentare la capacità dei collegamenti ferroviari senza necessariamente attivarne troppi.

6 Riformulazione e Risultati : Abbassare l'upper bound

6.1 Valid Inequalities

Visto che tutte le istanze nelle quali non riuscivamo ad ottenere un gap = 0, avevano tutte esaurite abbiamo pensato di aggiungere al modello un vincolo che impedisca al solutore di cercare tra le soluzioni che avessero un ciclo di lunghezza 3. Più formalmente:

$$y^h_{i_1 i_2} + y^h_{i_2 i_3} + y^h_{i_3 i_1} \leq 2, \;\; \forall (i_1, i_2), (i_2, i_3), (i_3, i_1) \in A, h = 1, ...H, i_1 < i_2, i_1 < i_3 \tag{15}$$

Purtroppo, questo vincolo che in teoria avrebbe dovuto sveltire la risoluzione del problema, ha fatto aumentare di troppo i tempi di creazione del modello e non ci ha reso possibile vedere quanto avremmo potuto guadagnare in termini di riduzione del gap lavorando sull'upper bound.

Abbiamo poi tentato di rimuovere la valid inequalities di cui sopra e sostituirla con le seguenti :

$$\sum_{h \in Orders} y_{ii}^h \leq 0 \; \forall i \in N \tag{16}$$

$$y_{i_1 i_2}^h + y_{i_2 i_1}^h \leq 1 \quad \forall (i_1, i_2) \in A, h = 1, ...H, i_1 < i_2, \tag{17}$$

In particolare, eq. (16) impedisce la creazione nel modello di collegamenti tra nodi origine e destinazione se il nodo è lo stesso, mentre eq. (17) impedisce la creazione di percorsi che usano lo stesso collegamento per una data commodity, ossia impedisce la generazione di soluzioni aventi cicli di lunghezza 2.
Questa soluzione abbiamo immaginato potesse portare ad un branch & bound più efficace perchè scartavamo a priori soluzioni che sapevamo non potessero essere corrette. In realtà, sebbene questa volta il modello venga correttamente creato ed eseguito, in nessuna istanza è stato apprezzato un miglioramento del gap. I risultati e le istanze generate delle esecuzioni sono comunque visionabili nella cartella **dati1NEWVALID** ma non li abbiamo riportati in tabella nella relazione visto che non è stata una strategia che ha portato migliorie.

6.2 Column generation

In tutte le istanze nelle quali è stato riscontrato un gap>0, il solver non è riuscito ad ottenere una soluzione ottima nei 5 minuti di time limit. Visto che le istanze con $K = 12$ e $K = 25$ non mostravano risultati differenti rispeto a $K = 7$, abbiamo deciso di concentrare gli sforzi solamente sulle istanze con $K = 2$ e $K = 7$. Per cercare di ridurre quanto più possibile il tempo di esecuzione abbiamo quindi adottato un approccio di risoluzione incrementale, partendo dalla più piccola soluzione ammissibile e aggiungendo di volta in volta nodi secondo una semplice euristica che aggiunge i nodi a costo minimo.
Problema ridotto
Il primo tentativo è constato nel tentare di risolvere il sotto-problema caratterizzato dall'uso dei soli nodi presenti nelle commodities. Le soluzioni a questo problema certamente sono ammissibili nel problema originale e questo approccio, utilizzato per le sole istanze nelle quali avevamo un gap > 0 (10 in totale) ha migliorato il gap tra la soluzione del problema ridotto e il lowerbound del problema originale in una sola tra le 10 istanze analizzate. In table 5 sono visibili i risultati dell'analisi del problema ridotto. La colonna LB, UB e GAP indicano rispettivamente il lowerbound, l'upperbound e il GAP della risoluzione del problema originale. La colonna ColGen UB e GAP[ColGen] indicano invece la soluzione ottenuta nel problema ridotto e il gap calcolato rispetto al lowerbound del problema originale. In 3 tra le 10 istanze analizzate, la soluzione ottenuta nel problema ridotto è anche soluzione ottima (gap 0, in giallo nella tabella) ma non ha ridotto il gap rispetto al lower bound del problema originale. In una, invece, la soluzione ottenuta, oltre ad essere ottima nel problema ridotto ha anche dimezzato il gap nel problema originale (in verde in tabella).

Problema ridotto e aggiunta 10F
Per migliorare ulteriormente il gap, almeno per le soluzioni nelle quali avevamo ottenuto gap 0 nel problema ridotto (ma poi estesa anche alle altre), abbiamo pensato ad una strategia euristica di aggiunta di nodi al problema ridotto. In particolare, data una istanza contenente solo i nodi partenza e destinazione delle varie commmodities, abbiamo aggiunto i 10 nodi meno costosi tra quelli che collegano le coppie di nodi partenza e destinazione. Visto che ogni coppia di nodi può essere collegata sia da strade che da ferrovie i 10 nodi meno costosi sono stati aggiunti una volta secondo il criterio di costo delle strade e una volta secondo il costo delle ferrovie. I risulati sono visibili in table 6 dove al solito, oltre la colonna indicante l'istanza e il corrispettivo lowerbound, upperbound e gap, ci sono le colonne indicanti il nuovo upperbound aggiungendo al problema ridotto, i dieci nodi non ancora introdotti che collegano i nodi già aggiunti ([S] considerando i costi stradali e [F] i costi ferroviari). Adottando questa strategia abbiamo riscontrato un miglioramento sostanziale nell'istanza che già nel problema ridotto avevamo migliorato rispetto al problema originale (da 3 a 2 di gap vs 6 nel problema originale) e anche in un istanza che nel problema ridotto non aveva ottenuto un miglioramento ma che aggiugnendo ulteriori 10 nodi ottiene un miglioramento da 2.2 del problema originale a 1.7 nel caso in cui vengano aggiunti i 10 nodi minimizzando il costo dei collegamenti ferroviari. Inoltre, in tutti i casi da noi analizzati l'upperbound migliora all'aumentare del numero di nodi inseriti (provando ad aggiungerli uno per volta fino a 10).
Problema ridotto e aggiunta 5S e 15F
Riguardo le due istanze che continuano ad ottenere gap 0 nel problema ridotto ma upperbound peggiore del problema originale abbiamo pensato di aggiungere ulteriori nodi seguendo una strategia leggermente diversa. Invece di aggiungere i nodi in base al solo costo delle strade che li collegano o delle ferrovie, abbiamo deciso di aggiungerne 20 includendo i nodi che collegati dalle migliori 5 strade a costo minimo e le migliori 15 ferrorive a costo minimo. I risultati di questo tentativo sono stati soddisfacenti in quanto le due istanze ancora non migliorate hanno beneficiato di una riduzione sostanziale dell'upperbound migliorando il gap rispetto al problema originale come visibile in table 7. Oltre a questa strategia di aggiunta di 20 nodi di cui 15 a costo minimo per i collegamenti ferroviari e 5 per quelli stradali, abbiamo anche provato con 5S + 10F e 5S + 15F ma non abbiamo ottenuto risultati migliori e riportiamo nella relazione solo quelli con la strategia 5S + 15F. Nelle 6 istanze restanti in cui non abbiamo riscontrato miglioramenti abbiamo provato ad aggiungere i nodi secondo gli schemi 10S+10F, 25S e 25F. Tutte e tre i tentativi però non hanno portato alcun miglioramento del GAP ottenendo inoltre, GAP > 0 anche nel problema ridotto. Tutti i test, anche quelli non esplicitamente riportati in tabella nella relazione sono però visionabili nel file $RisultatiCG.xlxs$.

7 Conclusioni

Le istanze da noi generate, se considerate tutte quelle con 10 e 50 nodi e tra quelle con 100 nodi solo quelle con $k = 2e7$ (visto che $k = 12e25$ non mostra-

vano alcuna differenza nelle soluzioni), sono in totale 72. Di queste 72 istanze, 10 hanno ottenuto un gap > 0 e sulle quali abbiamo concentrato l'analisi nei punti descritti in section 5 tentando di diminuire il GAP alzando il lowerbound tramite rilassamento lagrangeano e in section 6 abbasando l'upperbound tramite column generation. La strategia di alzare il lowerbound, come implementata in questa relazione, sebbene abbia diminuito ex-post il gap non è una strategia che migliora la risoluzione effettuata dal solutore. La strategia di abbassare l'upperbound, come descritta in questa relazione e in section 6.2, ha invece migliorato le prestazioni del solutore migliorando il gap grazie ad un abbassamento dei tempi di calcolo adottando una strategia di scelta migliore dei nodi da utilizzare. Tra le 10 istanze che non eravamo riusciti a risolvere bene, quest'ultima strategia ha portato un considerevole miglioramento in 4, mentre le altre 6 non ne hanno beneficiato. Nel caso in cui, il solutore non fornisca nei tempi prescritti una soluzione accettabile, è consigliabile effettuare un tentativo di risoluzione del problema ridotto e aggiunta dei nodi come descritto in section 6.2 e aggiunendo al problema ridotto contenente i soli nodi partenza e destinazione delle commodity i 5 migliori nodi che minimizzano il costo stradale e i 15 migliori nodi che minimizzano il costo ferroviario (per istanze fino a 100 nodi).

8 Tabelle e Immagini

#N	%Ferr	K	Orders	Solver	Sol_avg	Sol_std	LB_avg	LB_std	Time_avg	Gap_avg	Gap_Std
10	0,1	2	5	glpk	819	153,5	819	153,5	0s	0%	0%
10	0,1	2	5	cbc	819	153,5	819	153,5	0,1s	0%	0%
10	0,1	3	5	glpk	805	148,7	805	148,7	0s	0%	0%
10	0,1	3	5	cbc	805	148,7	805	148,7	0,08s	0%	0%
50	0,1	2	25	glpk	18262,3	1889,4	18262,3	1889,4	11,7s	0%	0%
50	0,1	2	25	cbc	18262,3	1889,4	18262,3	1889,4	33,9s	0%	0%
50	0,1	4	25	glpk	15852,3	1689,2	15798,6	1613,4	106,6s	0.3%	0.4%
50	0,1	4	25	cbc	15841,3	1673,7	15841,3	1673,7	38,6s	0%	0%
50	0,1	6	25	glpk	15744,6	1656,2	15744,6	1656,2	20,7s	0%	0%
50	0,1	6	25	cbc	15744,6	1656,2	15744,6	1656,2	33,8s	0%	0%
50	0,1	13	25	glpk	15742,6	1653,4	15742,6	1653,4	17,3s	0%	0%
50	0,1	13	25	cbc	15742,6	1653,4	15742,6	1653,4	31,4s	0%	0%
100	0,1	2	50	glpk	62904	1941,6	61228	2550,9	/	2,6%	2,6%
100	0,1	2	50	cbc	NoSol	NoSol		NoSol		-	-
100	0,1	7	50	glpk	53421,3	1502,6	52272,3	1790,8	/	2,10%	0,70%
100	0,1	7	50	cbc	NoSol	NoSol		NoSol		-	-
100	0,1	12	50	glpk	53530	1323,2	52281,3	1822,5	/	2,30%	1,30%
100	0,1	12	50	cbc	NoSol	NoSol		NoSol		-	-
100	0,1	25	50	glpk	53529,3	1323,4	52281,3	1822,5	/	2,30%	1,30%
100	0,1	25	50	cbc	NoSol	NoSol		NoSol		-	-

Table 1: Risultati delle istanze con percentuale di attivazione ferrovie = 0.1 - media su 3 istanze per tipologia

#N	%Ferr	K	Orders	Solver	Sol_avg	Sol_std	LB_avg	LB_std	Time_avg	Gap_avg	Gap_Std
10	0,5	2	5	glpk	828,3	129,9	828,3	129,9	0s	0%	0%
10	0,5	2	5	cbc	828,3	129,9	828,3	129,9	0,07s	0%	0%
10	0,5	3	5	glpk	810	116,3	810	116,3	0s	0%	0%
10	0,5	3	5	cbc	810	116,3	810	116,3	0,07s	0%	0%
50	0,5	2	25	glpk	15401,6	2124	15401,6	2124	4,9s	0%	0%
50	0,5	2	25	cbc	15401,6	2124	15401,6	2124	28,6s	0%	0%
50	0,5	4	25	glpk	14239,6	1970,1	14239,6	1970,1	26,5s	0%	0%
50	0,5	4	25	cbc	14239,6	1970,1	14239,6	1970,1	29,7s	0%	0%
50	0,5	6	25	glpk	14210,6	1965,3	14210,6	1965,3	15,3s	0%	0%
50	0,5	6	25	cbc	14210,6	1965,3	14210,6	1965,3	27s	0%	0%
50	0,5	13	25	glpk	14210,6	1965,3	14210,6	1965,3	14s	0%	0%
50	0,5	13	25	cbc	14210,6	1965,3	14210,6	1965,3	27,17s	0%	0%
100	0,5	2	50	glpk	54272,6	6806,6	54272,6	6806,6	132,4s	0%	0%
100	0,5	2	50	cbc	NoSol	NoSol		NoSol		-	-
100	0,5	7	50	glpk	48818,6	4085,1	48538,3	3886,7	/	0,5%	0,4%
100	0,5	7	50	cbc	NoSol	NoSol		NoSol		-	-
100	0,5	12	50	glpk	48782,3	4069,7	48542	3888,2	/	0,46%	0,36%
100	0,5	12	50	cbc	NoSol	NoSol		NoSol		-	-
100	0,5	25	50	glpk	48782,3	4069,7	48542	3888,2	/	0,46%	0,36%
100	0,5	25	50	cbc	NoSol	NoSol		NoSol		-	-

Table 2: Risultati delle istanze con percentuale di attivazione ferrovie = 0.5 - media su 3 istanze per tipologia

#N	%Ferr	K	Orders	Solver	Sol_avg	Sol_std	LB_avg	LB_std	Time_avg	Gap_avg	Gap_Std
10	0,9	2	5	glpk	753,6	164,4	753,6	164,4	0s	0%	0%
10	0,9	2	5	cbc	753,6	164,4	753,6	164,4	0,08s	0%	0%
10	0,9	3	5	glpk	730	161,3	730	161,3	0s	0%	0%
10	0,9	3	5	cbc	730	161,3	730	161,3	0,06s	0%	0%
50	0,9	2	25	glpk	14584,3	929,5	14584,3	929,5	4,1s	0%	0%
50	0,9	2	25	cbc	14584,3	929,5	14584,3	929,5	29,1s	0%	0%
50	0,9	4	25	glpk	13825,3	666,1	13825,3	666,1	16,4s	0%	0%
50	0,9	4	25	cbc	13825,3	666,1	13825,3	666,1	27,2s	0%	0%
50	0,9	6	25	glpk	13814,6	657,9	13814,6	657,9	7,6s	0%	0%
50	0,9	6	25	cbc	13814,6	657,9	13814,6	657,9	30s	0%	0%
50	0,9	13	25	glpk	13814,6	657,9	13814,6	657,9	8s	0%	0%
50	0,9	13	25	cbc	13814,6	657,9	13814,6	657,9	27,6s	0%	0%
100	0,9	2	50	glpk	50513,3	2112,1	50135,3	2265,4	/	0,7%	1,0%
100	0,9	2	50	cbc	NoSol	NoSol		NoSol		-	-
100	0,9	7	50	glpk	47699,3	2044	47627,3	2117,9	/	0,15%	0,23%
100	0,9	7	50	cbc	NoSol	NoSol		NoSol		-	-
100	0,9	12	50	glpk	47699,3	2044	47627,3	2117,9	/	0,15%	0,23%
100	0,9	12	50	cbc	NoSol	NoSol		NoSol		-	-
100	0,9	25	50	glpk	47699,3	2044	47627,3	2117,9	/	0,15%	0,23%
100	0,9	25	50	cbc	NoSol	NoSol		NoSol		-	-

Table 3: Risultati delle istanze con percentuale di attivazione ferrovie = 0.9 - media su 3 istanze per tipologia

ISTANZA	**LB**	**UB**	**GAP**	**LBLagrange**	**GAP[Lagrange]**	**TimeLagrange**
inst_50_01_4_3.cmpl	18076	18237	0.883	18164	0.4	3.2s
inst_100_01_2.cmpl	59180	60244	1.766	60336	0.153	243.3s
inst_100_01_2_2.cmpl	59680	63644	6.228	64954	2.058	412.9s
inst_100_05_7.cmpl	52252	52700	0.85	52306	0.748	117.7s
inst_100_05_7_3.cmpl	50191	50584	0.777	50373	0.417	119s
inst_100_01_7.cmpl	50936	52563	3.095	51563	1.902	109.7s
inst_100_01_7_2.cmpl	51078	52167	2.088	52026	0.27	106.1s
inst_100_01_7_3.cmpl	54804	55534	1.315	55325	0.376	112.5s
inst_100_09_2.cmpl	48870	50004	2.268	49081	1.846	143.8s
inst_100_09_7.cmpl	45419	45635	0.473	45482	0.335	131.8s

Table 4: In giallo la soluzione trovata dal solutore nel limite dei 300 secondi. La colonna LBLagrange misura la soluzione del problema lagrangeano all'interno della funzione obiettivo del problema originale

ISTANZA	LB	UB	GAP	ColGen UB	GAP[ColGen]
inst_50_01_4_3.cmpl	18076	18237	0.883	19666	8.085
inst_100_01_2.cmpl	59180	60244	1.766	63331	6.554
inst_100_01_2_2.cmpl	59680	63644	**6.228**	61600	**3.117**
inst_100_05_7.cmpl	52252	52700	0.85	55100	5.169
inst_100_05_7_3.cmpl	50191	50584	0.777	53026	5.346
inst_100_01_7.cmpl	50936	52563	3.095	57005	10.646
inst_100_01_7_2.cmpl	51078	52167	2.088	55851	8.546
inst_100_01_7_3.cmpl	54804	55534	1.315	61599	11.031
inst_100_09_2.cmpl	48870	50004	2.268	51423	4.965
inst_100_09_7.cmpl	45419	45635	0.473	48040	5.456

Table 5: In verde l'istanza nella quale il gap (tra upper bound del problema ridotto e lower bound del problema originale) è migliorato. In giallo le istanze nelle quali il gap considerando il lower bound originale non è migliorato ma nelle quali il gap tra UB e LB del problema ridotto è 0, ossia il solutore ha trovato soluzione ottima del problema ridotto.

ISTANZA	LB	UB	GAP	ColGen10[S]	GAP CG10[S]	ColGen10[F]	GAP CG10[F]
inst_50_01_4_3	18076	18237	0.883	18639	3.021	19119	5.455
inst_100_01_2	59180	60244	1.766	62589	5.447	60590	2.327
inst_100_01_2_2	59680	63644	**6.228**	61206	**2.493**	61008	**2.177**
inst_100_05_7	52252	52700	0.85	54405	3.957	54267	3.713
inst_100_05_7_3	50191	50584	0.777	51265	2.095	52004	3.486
inst_100_01_7	50936	52563	3.095	56459	9.782	55325	7.933
inst_100_01_7_2	51078	52167	2.088	54999	7.129	55011	7.149
inst_100_01_7_3	54804	55534	1.315	59992	8.648	58797	6.791
inst_100_09_2	48870	50004	**2.268**	49854	**1.974**	49733	**1.735**
inst_100_09_7	45419	45635	0.473	47357	4.092	47255	3.885

Table 6: In verde le istanza nella quale il gap (tra upper bound del problema ridotto e lower bound del problema originale) è migliorato. In giallo le istanze nelle quali il gap considerando il lower bound originale non è migliorato ma nelle quali il gap tra UB e LB del problema ridotto è 0, ossia il solutore ha trovato soluzione ottima del problema ridotto.

ISTANZA	**LB**	**UB**	**GAP**	**ColGen UBMisto**	**GAP[ColGen]**
inst_50_01_4_3.cmpl	18076	18237	0,883	18204	**0,703**
inst_100_01_2.cmpl	59180	60244	1,766	59835	**1,095**

Table 7: In verde le istanze nella quale il gap (tra upper bound del problema ridotto e lower bound del problema originale) è migliorato aggiungendo 5 strade e 15 ferrovie.