POLITECNICO DI MILANO
Scuola di Ingegneria dell'Informazione

POLO TERRITORIALE DI COMO
Corso di Laurea Specialistica in
Ingegneria Informatica

Modelli e metodi di Ottimizzazione per l'instradamento di flussi soggetto a vincoli di equità di suddivisione della banda

Relatore: Prof. Edoardo Amaldi
Correlatore: Dott. Stefano Coniglio

Tesi di laurea di: Nicotra Andrea
matr. 760054

Anno Accademico 2012-2013

Sommario

Questa tesi di Ricerca Operativa riguarda l'instradamento di comunicazioni su reti di telecomunicazione soggetto a meccanismi automatici di allocazione della banda (e.g. Transfer Control Protocol – TCP). Il problema affrontato è quello dell'operatore di rete che, volendo massimizzare una funzione di utilità del servizio proporzionale alla quantità totale di traffico instradato, deve determinare i cammini per ogni coppia di origine-destinazione. Si suppone che, scelti dall'operatore i cammini, nelle reti IP (Internet Protocol) il meccanismo automatico di allocazione della banda sia ben approssimato dal cosiddetto principio di Max-Min Fairness (MMF).

Viene così proposto un modello di Programmazione Lineare Misto-Intera (PLMI) che mira a massimizzare la sommatoria dei flussi assegnati, includendo il rispetto del principio di Max-Min Fairness sotto forma di vincoli. Questo approccio è diverso da quello classico adottato in letteratura, dove il raggiungimento di una soluzione Max-Min Fair viene considerato come obiettivo dell'ottimizzazione, piuttosto che come vincolo da rispettare.

Successivamente viene analizzato il problema della formazione di sotto-cicli nelle soluzioni. Sono quindi definiti e confrontati diversi insiemi di vincoli per la rimozione dei sotto-cicli, proponendo tre diverse formulazioni estese ed un approccio iterativo di disuguaglianze valide (piani di taglio) nel contesto di un algoritmo di Branch-and-Cut.

Si definisce poi un insieme di vincoli da aggiungere alla formulazione nel tentativo di rafforzarla. Vengono proposti vincoli alternativi per il bilanciamento del flusso e per il raggiungimento della Max-Min Fairness.

Infine, vengono presentati dei metodi euristici basati su un algoritmo esatto di assegnazione equa del flusso, chiamato "waterfilling". L'intento è di ridurre gli elevati tempi di risoluzione causati dalla notevole complessità del modello, fornendo soluzioni di buona qualità utilizzate nel contesto di un algoritmo di Branch-and-Bound.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

In questa tesi viene trattato un problema di instradamento di comunicazioni su reti di telecomunicazione soggetto a vincoli di equità della suddivisione della banda, rilevante dal punto di vista applicativo per le reti internet. Un esempio è quello di un insieme di utenti che, dovendo scaricare dati dalla rete senza la necessità di soddisfare una precisa domanda, si aspettano di farlo il più velocemente possibile, ricevendo la quantità di banda maggiore possibile.

1.1 Instradamento di comunicazioni ed equità

Uno dei principali problemi affrontati nell'ambito dell'ottimizzazione di reti è quello riguardante l'instradamento di comunicazioni, detto *routing*, attraverso reti con capacità limitate. Consideriamo una rete, costituita da un insieme di nodi e di archi con relativa capacità, ed un insieme di comunicazioni caratterizzate da una sorgente s, una destinazione t ed, eventualmente, una domanda da soddisfare. L'obiettivo del problema generale di instradamento consiste nel determinare i percorsi per ogni comunicazione con le relative quantità di flusso transitanti su ciascun arco in modo da massimizzare la banda totale assegnata.

Un aspetto su cui viene posta grande attenzione nello studio di modelli per problemi di instradamento su reti è quello relativo all'allocazione delle risorse secondo un principio di equità, in inglese *"fairness"*. L'idea è quella di allocare la banda alle varie comunicazioni in modo che nessuna venga penalizzata. Questo perché nella risoluzione di problemi di instradamento è possibile che le soluzioni presentino sia comunicazioni a cui vengono assegnate quantità di flusso estremamente contenute sia comunicazioni a cui vengono assegnate quantità molto elevate. Nonostante il concetto di equità possa sembrare di facile intuizione ed applicazione, trovarne una definizione che sia soddisfacente rispetto agli obiettivi classici di instradamento è un

problema tutt'altro che banale. Un possibile approccio alla risoluzione del problema di allocazione equa è quello di assegnare la stessa quantità di flusso a ciascuna comunicazione, garantendo così l'assenza di comunicazioni "dominanti". Tuttavia tale approccio condurrebbe ad un utilizzo poco efficiente della rete, con possibile presenza di comunicazioni a cui viene assegnata una maggiore quantità di banda. L'obiettivo invece è quello di assegnare le risorse in modo che l'allocazione sia efficiente ed equa allo stesso tempo. In quest'ottica, un approccio alternativo, diventato punto di riferimento per l'allocazione equa delle risorse in problemi di instradamento, è quello della *Max-Min Fairness (MMF)*, ampiamente descritto ed analizzato in [1], [2]. Il nostro studio si concentra su questa definizione di equità.

In letteratura inoltre ritroviamo studi effettuati su differenti varianti del problema di instradamento, relative al tipo di instradamento effettuato. Infatti può essere concesso che il flusso di una singola comunicazione venga suddiviso lungo cammini differenti, i quali si ricongiungono in modo da garantire che al nodo destinazione arrivi l'intera quantità di flusso partito dalla sorgente. Questi problemi sono noti come *Splittable Multi Commodity Flow problems*. Nel nostro caso affrontiamo un problema di *Unsplittable Multi Commodity Flow*, in cui il flusso di ogni singola comunicazione non può essere suddiviso su più percorsi, ma deve essere instradato lungo un solo cammino.

1.2 Principio di Max-Min Fairness (MMF)

Il principio di condivisione equa delle risorse (MMF) equivale a suddividere le risorse in modo che non sia possibile aumentare la quantità assegnata a ciascuna domanda senza diminuire quella assegnata a una domanda che sta sfruttando una quantità uguale o minore.

Presentiamo ora un semplice esempio, estraneo al mondo delle reti, per chiarire l'idea di allocazione MMF. Tre persone vogliono condividere 100cl di birra in tre bicchieri di differente capacità, rispettivamente di 25cl, 40cl e 50cl. La soluzione più equa è quella che assegna 25cl alla prima persona e 37,5cl alle altre due, come mostrato in Figura 1.1. In pratica alla persona col bicchiere più piccolo è stato assegnato la maggior quantità di birra possibile, mentre la rimanente è stata equamente divisa tra le altre due persone, in modo da non penalizzare nessuno. Questo semplice esempio illustra i principali aspetti dell'allocazione MMF delle risorse, applicando quello che viene chiamato *waterfilling algorithm*.

Da un punto di vista intuitivo, il procedimento appena descritto corrisponde a massimizzare la quantità di chi riceve il minimo, successivamente a massimizzare

Figura 1.1: Esempio di condivisione *fair* di 100cl di birra tra tre persone. Alla persona col bicchiere meno capiente viene assegnato il massimo, 25cl, ai restanti due viene assegnata la stessa quantità: 37,5cl.

la quantità del secondo che riceve il minimo e così via.

Rimandiamo al Capitolo 2 la descrizione più formale di questo principio, contestualizzandolo poi nell'ambito dell'allocazione del flusso nelle reti di telecomunicazioni.

1.3 Il problema

In letteratura il paradigma MMF è stato utilizzato esclusivamente come obiettivo per problemi di instradamento e allocazione di flusso. Questo nonostante nelle reti IP (Internet Protocol) la banda venga assegnata con meccanismi distribuiti che non possono essere controllati dagli operatori di rete, come ad esempio il protocollo TCP. Sostanzialmente gli operatori di rete hanno interesse ad instradare le comunicazioni in modo da ottimizzare uno degli obiettivi classici dei problemi di instradamento, come la massimizzazione di una funzione di utilità, piuttosto che massimizzare una funzione di equità.

In [3] viene invece proposto un modello per risolvere un problema di instradamento multi-comunicazione in cui il paradigma MMF viene imposto sotto forma di vincoli per l'assegnazione della banda. Si tratta del problema dell'operatore di rete che vuole massimizzare una funzione di prestazione e di utilità della rete, come la somma dei flussi, tenendo conto che l'allocazione dei flussi viene effettuata automaticamente dal protocollo di rete in modo MMF.

Andiamo ora a formalizzare il problema sopra descritto. I dati a disposizione sono i seguenti:

- V, che rappresenta l'insieme dei nodi della rete;

- A, che rappresenta l'insieme degli archi della rete;
- $G = (V, A)$, che rappresenta il grafo orientato definito sugli insiemi V e A;
- K, che rappresenta l'insieme delle comunicazioni "elastiche" caratterizzate dalle corrispondenti coppie di origine-destinazione (s, t), con $(s, t) \in K$.

Definiamo inoltre:

- $c_{ij} \geq 0 \quad \forall (i, j) \in A$, che rappresenta la capacità di ogni arco (i, j);
- $\phi^{st} \quad \forall (s, t) \in K$, che rappresenta la quantità di flusso assegnata a ciascuna comunicazione (s, t).

Viene chiamato *MMF-Constrained Traffic Engineering problem (MMF-CTE)* il problema dove, per ogni coppia $(s, t) \in K$, deve essere determinato un singolo cammino nel grafo G in modo da massimizzare la funzione di utilità legata alla quantità di flusso assegnata a ciascuna comunicazione ϕ^{st}, assumendo che la banda sia allocata secondo il principio di MMF. Come funzione di utilità consideriamo la funzione pesata:

$$\sum_{st \in K} w^{st} \phi^{st} \tag{1.1}$$

dove w^{st} rappresenta il peso non negativo associato alla coppia (s, t), grazie al quale l'operatore di rete può assegnare una differente priorità a ciascuna connessione, in modo da affrontare situazioni come, ad esempio, utenti premium che pagano di più per avere più banda.

Come mostrato in [4], il problema MMF-CTE è NP-difficile e quindi, a meno che $P = NP$, non può esistere un algoritmo di risoluzione polinomiale.

Intuitivamente, possiamo vedere il problema MMF-CTE come un problema a due livelli, dove il primo decisore sceglie i cammini per ogni coppia origine-destinazione, mentre il secondo alloca i flussi. Sapendo in che modo agisce il secondo decisore, il primo può effettuare una scelta dei cammini in modo da ottimizzare la distribuzione dei flussi.

Riportiamo in Figura 1.2 un esempio che mostra le differenze tra i vettori di allocazione del flusso nel caso in cui si massimizzi la somma totale dei flussi assegnati rispettando o meno il principio di MMF. Supponiamo che tutti gli archi della rete rappresentata abbiano capacità infinita, eccetto gli archi di cui sono riportate in figura le capacità. L'obiettivo è quello di instradare i flussi relativi alle tre coppie di origine-destinazione (s_1, t_1), (s_2, t_2) e (s_3, t_3) massimizzando la loro somma. Nel primo caso, dove non viene imposto il principio di MMF, sono instradate 14 unità

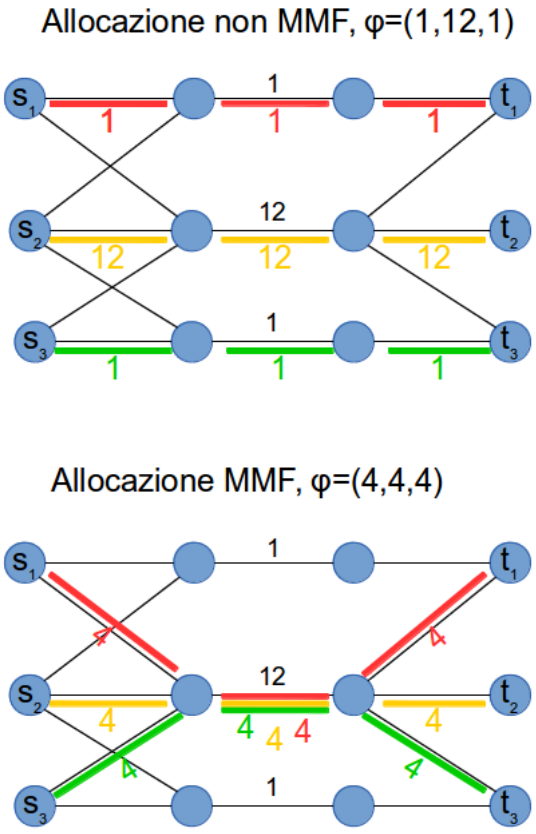

Figura 1.2: Esempio di allocazione del flusso con e senza il principio MMF. La capacità degli archi per cui non è riportato un valore viene supposta infinita.

di flusso, corrispondenti al vettore $\underline{\phi} = (1, 12, 1)$. Nel secondo caso viene imposto il rispetto dei vincoli MMF, portando ad un instradamento totale di 12 unità di flusso, corrispondenti al vettore $\underline{\phi} = (4, 4, 4)$. L'ottimizzazione soggetta al principio di MMF conduce ad un valore della funzione obiettivo minore, ma allo stesso tempo garantisce una distribuzione più equa delle risorse tra le tre coppie di origine-destinazione.

1.4 Obiettivi e organizzazione della tesi

Il primo obiettivo di questa tesi consiste nello stabilire se, come in molti altri problemi di ottimizzazione combinatoria, è più promettente un approccio di tagli applicati al modello di ottimizzazione MMF-CTE oppure un approccio basato su formulazioni estese (compatte) con numero polinomiale di vincoli. Il secondo obiettivo consiste nel determinare se tagli aggiuntivi sono utili a migliorare la formulazione iniziale. Il terzo obiettivo è quello di sviluppare approcci euristici specifici per il problema

affrontato e verificarne l'efficacia in termini di miglioramento delle soluzioni trovate e di tempi di risoluzione.

Nel Capitolo 2 citiamo alcuni lavori relativi a problemi di ottimizzazione di reti che coinvolgono il principio MMF e presentiamo la formulazione PLMI dal quale partiamo. Nel Capitolo 3 descriviamo le formulazioni estese e l'approccio dei piani di taglio proposti per l'eliminazione dei sotto-cicli. Nel Capitolo 4 presentiamo lo studio effettuato relativo a nuovi possibili tagli da aggiungere alla nostra formulazione. Nel Capitolo 5 descriviamo gli approcci euristici implementati per supportare il procedimento di Branch-and-Cut nella risoluzione del problema. Infine nei Capitoli 6 e 7 riportiamo rispettivamente l'analisi dei risultati computazionali e le conclusioni del nostro lavoro.

Capitolo 2

Ottimizzazioni di reti MMF: preliminari

Presentiamo in questo capitolo un'analisi più approfondita sul problema di allocazione del flusso secondo un criterio di MMF.

Nella Sezione 2.1 riassumiamo brevemente i principali lavori relativi all'ottimizzazione di reti MMF presenti in letteratura. Nella Sezione 2.2 presentiamo una definizione formale del principio di MMF. Nella Sezione 2.3 descriviamo l'algoritmo di *waterfilling* per l'allocazione del flusso in modo MMF su un insieme di cammini dati. Infine, nella Sezione 2.4, descriviamo la formulazione di PLMI dalla quale partiamo nel nostro lavoro.

Lo scopo di questo capitolo è quello di fornire al lettore gli elementi necessari per una comprensione completa degli argomenti trattati nei successivi capitoli.

2.1 Ottimizzazione di reti con MMF

Precedenti lavori legati all'ottimizzazione di reti MMF considerano diversi principi di allocazione dei flussi e tipi di instradamento.

Se un percorso è già stato determinato per ogni origine-destinazione, un semplice algoritmo tempo-polinomiale, conosciuto come *Water (o Proportional) filling* [5], è sufficiente ad allocare la banda secondo il principio di MMF. Se i percorsi di instradamento non sono noti a priori, sono stati proposti algoritmi volti a determinare un instradamento tale che l'allocazione di banda MMF sia il quanto più equa possibile per *splittable routing*, vedi [6] [2], e *unsplittable routing*, vedi [7], [8], [6].

Nel caso di un problema di ottimizzazione generale su un dato insieme di origini-destinazioni, possiamo determinare una soluzione MMF risolvendo, per ogni

coppia, una versione leggermente modificata del problema originale (vedi [2], [9]. per il caso in cui il problema originale è convesso, o [10] se non convesso). Una definizione differente di fairness adatta all'implementazione di algoritmi più semplici viene proposta in [11].

In [12] viene proposta una formulazione di programmazione lineare che agisce su routing e allocazione del flusso al fine di bilanciare il flusso totale e la fairness nel caso di flussi splittable. In [13] viene mostrato come, sotto determinate condizioni, le soluzioni MMF possano essere ottenute ottimizzando funzioni obiettivo (non lineari) scelte appositamente.

Il lavoro sviluppato nell'ambito di questa tesi si basa sul modello descritto in [3] e considerato anche in [4], in cui il criterio MMF non viene considerato come un obiettivo ma come un vincolo imposto ad un problema più generale di ottimizzazione.

2.2 Definizione formale di MMF

Forniamo ora una definizione formale di MMF che, come in [1, 8], è basata sul concetto di ordine lessicografico.

Definizione 1 *Un vettore* $\underline{\phi} = (\phi_1, \phi_2, ..., \phi_k) \in \mathcal{R}^k$ *è lessicograficamente maggiore del vettore* $\underline{\psi} = (\psi_1, \psi_2, ..., \psi_k) \in \mathcal{R}^k$, $\underline{\phi} \succ \underline{\psi}$, *se esiste* $l \in \{1, 2, .., k\}$ *tale che* $\phi_l = \psi_m$ *per ogni* $m \in \{1, 2, ..., l-1\}$ *e* $\phi_l > \psi_l$.

Definiamo $\langle \underline{\phi} \rangle = (\langle \phi \rangle_1, \langle \phi \rangle_2, ..., \langle \phi \rangle_k)$ come la versione del vettore $\underline{\phi} = (\phi_1, \phi_2, ..., \phi_k) \in \mathcal{R}^k$ ordinato in un ordine non decrescente.

Definizione 2 *Il vettore di allocazione dei flussi* $\underline{\phi}$ *è MMF se e solo se, per ogni altro vettore* $\underline{\psi} \in \mathcal{R}^k$, $\langle \underline{\phi} \rangle$ *domina lessicograficamente* $\langle \underline{\psi} \rangle$, *ovvero* $\langle \underline{\phi} \rangle \succ \langle \underline{\psi} \rangle$.

In altre parole, il vettore dei flussi $\underline{\phi}$ è MMF se non è possibile aumentare il flusso assegnato ad una qualsiasi coppia di origine-destinazione senza diminuire il flusso assegnato ad un'altra coppia di origine-destinazione che sta ricevendo una quantità minore.

2.3 Flussi MMF per cammini dati

Presentiamo in questa sezione un algoritmo di allocazione di flusso che, dato un cammino per ogni coppia origine-destinazione, permette di determinare un flusso

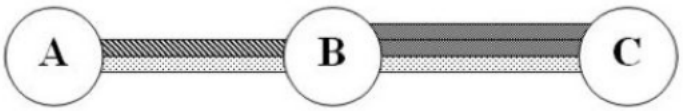

Figura 2.1: Semplice esempio di allocazione MMF. L'arco AB viene saturato dalle comunicazioni A-B e A-C, definendo un flusso pari ad un'unità per entrambe le comunicazioni. L'arco BC contiene l'unità di flusso della comunicazione A-C ed assegna le restanti due unità alla comunicazione BC, diventando così saturo.

MMF. L'algoritmo qui riportato verrà utilizzato come punto di partenza per lo sviluppo delle euristiche presentate nel Capitolo 5.

2.3.1 Esempio di allocazione MMF

In un problema di assegnazione di flusso MMF in reti di telecomunicazione, le risorse disponibili devono essere suddivise tra le differenti coppie origine-destinazione che utilizzano la rete, in modo che nessuna coppia possa aumentare la banda a sua disposizione senza diminuire quella di coppie a cui è stata assegnata una quantità uguale o minore.

Riportiamo in figura 2.1 un esempio di allocazione MMF. Consideriamo una rete costituita da 3 nodi (A, B, C) e i relativi archi AB, con capacità 2, BC con capacità 3. Assumiamo di avere 3 comunicazioni a cui allocare il flusso A-B, B-C, A-C, instradate rispettivamente attraverso gli archi AB, BC, AB-BC. Iniziamo ad allocare il flusso incrementando di 1 unità ciascuna le quantità assegnate a ciascuna comunicazione. Risulterà che l'arco AB, di capacità 2, sarà subito saturato, con conseguente allocazione di flusso pari a 1 per ciascuna delle due comunicazioni che lo utilizzano. L'arco BC, invece, avendo capacità 3, ha a disposizione un'unità libera che può essere assegnata alla comunicazione B-C. Il vettore finale del flusso assegnata alle comunicazioni risulta essere $\underline{\phi} = (1, 2, 1)$.

Nell'esempio appena mostrato, abbiamo applicato l'algoritmo di assegnazione del flusso progressivo, chiamato anche algoritmo di waterfilling, presentato in [5]. Descriviamolo ora in maniera più formale.

2.3.2 Algoritmo waterfilling

Dato il grafo $G = (V, A)$, le capacita c_{ij} per ogni arco $(i, j) \in A$ ed un cammino per ogni coppia di origine-destinazione (s, t), l'algoritmo waterfilling determina il flusso ϕ^{st} per ogni coppia di origine-destinazione $(s, t) \in K$. Una volta inizializzato

il vettore dei flussi $\underline{\phi}$ con una quantità pari a zero per ogni coppia (s,t), ad ogni iterazione viene selezionato l'arco la cui capacità divisa per il numero di coppie origine-destinazione che lo utilizzano, definita da $|\{(s,t) \in K : (i,j) \in P_{st}\}|$, risulta essere più piccola, in modo da determinare un valore δ per l'incremento dei flussi. Si aumentano uniformemente, di una quantità pari a δ, le allocazioni di tutte le coppie (s,t), finché uno o più archi non vengono saturati. Si rimuovono le coppie (s,t) a cui non è più possibile incrementare il flusso, si aggiornano le capacità residue degli archi e si ripete il procedimento fino a non poter più incrementare nessuna allocazione.

SISTEMARE NOTAZIONE!!!

Data: Archi con relative capacità, percorsi definiti per ogni coppia (s,t)
Result: Vettore di allocazione MMF f^0
Inizializzazione:;
$\phi^{st} = 0 \qquad \forall (s,t) \in K$;
$q = 0$ (contatore delle iterazioni);
while *Esistono connessioni con archi non ancora saturati* **do**
  $q := q + 1$;
  $\delta := \min_{(i,j)\in A} \frac{c_{ij}}{|\{(s,t)\in K:(i,j)\in P_{st}\}|}$;
  $c_{ij} := c_{ij} - \delta\,|\{(s,t) \in K : (i,j) \in P_{st}\}|$;
  $\phi_{st} := \phi_{st} + \delta$;
  Rimuovere archi saturati ($c_{ij} = 0$);
  Rimuovere connessioni che utilizzano archi saturati;
end

Algorithm 1: Algoritmo di Waterfilling per l'assegnazione MMF del flusso.

Osservando l'allocazione finale, si nota che le coppie (s,t) che sono state rimosse nella prima iterazione hanno ricevuto il valore massimo di flusso possibile, allo stesso modo quelle rimosse nella seconda iterazione e a seguire. Il valore del contatore q ci restituisce l'informazione riguardante il numero di volte che viene eseguita l'operazione descritta e il numero di valori distinti che troviamo nel vettore finale $\underline{\phi}$.

Il vettore di allocazione finale $\underline{\phi} = (\phi_1, ..., \phi_k)$ presenta una proprietà interessante, quella di avere un arco saturato per ogni coppia origine-destinazione.

Proprietà *Per ogni coppia (s,t) esiste un arco saturo b, chiamato "bottleneck", sul cammino P_{st} tale che ϕ_{st} è almeno tanto grande quanto il flusso assegnato ad ogni altra coppia che utilizza l'arco b.*

Osservando la Figura 2.1 possiamo riscontrare questa proprietà per ogni coppia origine-destinazione presentata nell'esempio.

2.4 Formulazione PLMI parziale

Riportiamo ora il modello di programmazione lineare mista-intera (PLMI) proposto in [3], con variabili associate agli archi, per risolvere il problema con i risolutori MILP allo stato dell'arte.

Definiamo le seguenti variabili:

- x_{ij}^{st}, variabile binaria che indica se l'arco (i, j) viene utilizzato per la comunicazione (s, t);
- f_{ij}^{st}, quantità di flusso allocata per la comunicazione (s, t) sull'arco (i, j);
- u_{ij}, flusso allocato per la comunicazione che riceve la quantità maggiore sull'arco (i, j);
- y_{ij}^{st}, variabile binaria che indica se l'arco (i, j) è un arco bottleneck per la comunicazione (s, t).

La formulazione PLMI è definita come segue:

Funzione obiettivo

$$\Psi = \max \sum_{st \in K} w^{st} \phi^{st}. \tag{2.1}$$

Viene massimizzata la somma dei flussi pesati di tutte le coppie origine-destinazione. Il peso w^{st} per ogni coppia (s, t) consente all'operatore di rete di assegnare differenti priorità alle comunicazioni.

Vincoli

Vincoli di flusso "multi-commodity unsplittable":

$$\sum_{ij \in A} f_{ij}^{st} - \sum_{ji \in A} f_{ji}^{st} = \begin{cases} \phi^{st} & if\ i = s \\ -\phi^{st} & if\ i = t \\ 0 & else \end{cases} \quad \forall i \in V, (s, t) \in K, \tag{2.2}$$

$$\sum_{st \in K} f_{ij}^{st} \leq c_{ij} \qquad \forall (i,j) \in A, \tag{2.3}$$

$$f_{ij}^{st} \leq c_{ij} x_{ij}^{st} \qquad \forall (i,j) \in A, (s,t) \in K, \tag{2.4}$$

$$\sum_{ij \in A} x_{ij}^{st} \leq 1 \qquad \forall i \in V, (s,t) \in K, \tag{2.5}$$

$$x_{ij}^{st} \in \{0,1\} \qquad \forall (i,j) \in A, (s,t) \in K, \tag{2.6}$$

$$f_{ij}^{st} \geq 0 \qquad \forall (i,j) \in A, (s,t) \in K, \tag{2.7}$$

$$\phi^{st} \geq 0 \qquad \forall (s,t) \in K. \tag{2.8}$$

Vincoli MMF:

$$\sum_{ij \in A} y_{ij}^{st} \geq 1 \qquad \forall (s,t) \in K, \tag{2.9}$$

$$\sum_{od \in K} f_{ij}^{od} \geq c_{ij} y_{ij}^{st} \qquad \forall (i,j) \in A, (s,t) \in K, \tag{2.10}$$

$$u_{ij} \geq f_{ij}^{st} \qquad \forall (i,j) \in A, (s,t) \in K, \tag{2.11}$$

$$f_{ij}^{st} \geq u_{ij} - c_{ij}(1 - y_{ij}^{st}) \qquad \forall (i,j) \in A, (s,t) \in K, \tag{2.12}$$

$$y_{ij}^{st} \in \{0,1\} \qquad \forall (i,j) \in A, (s,t) \in K, \tag{2.13}$$

$$u_{ij} \geq 0 \qquad \forall (i,j) \in A. \tag{2.14}$$

Disequazioni valide:

$$y_{ij}^{st} \leq x_{ij}^{st} \qquad \forall (i,j) \in A, (s,t) \in K, \tag{2.15}$$

$$\phi^{st} \geq \frac{\min_{ij \in A}\{c_{ij}\}}{|K|} \qquad \forall (s,t) \in K. \tag{2.16}$$

Vincoli per l'eliminazione dei sotto-cicli:
insieme di vincoli volti ad evitare la formazione di sotto-cicli esterni al cammino tra una coppia origine-destinazione. Avendo affrontato in maniera più approfondita questo problema, rimandiamo la descrizione di questi vincoli nel Capitolo 3.

Analizziamo ora i vincoli del problema che abbiamo suddiviso, per comodità di lettura, in tre gruppi:

- **Vincoli di flusso:** i vincoli (2.2) garantiscono il bilanciamento del flusso in ogni nodo, i vincoli (2.3) impongono che la somma dei flussi di tutte le

comunicazioni su un arco non ecceda la capacità dell'arco, le disequazioni (2.4) vincolano il flusso di ogni comunicazione a transitare su archi selezionati per quella comunicazione e la disequazione (2.5) assicura che per ogni coppia (s, t) venga definito un solo cammino. Coi vincoli (2.6), (2.7) e (2.8) sono definiti i domini delle variabili x, f e ϕ.

- **Vincoli MMF:** la disequazione (2.9) assicura che vi sia almeno un arco bottleneck per ciascuna comunicazione, i vincoli (2.10) impongono la saturazione di archi bottleneck, i vincoli (2.11) assegnano alla variabile u_{ij} il valore del flusso massimo che transita sull'arco (i, j), i vincoli (2.12) impongono che il flusso su un arco bottleneck (i, j) per una coppia (s, t) sia grande almeno quanto il flusso assegnato alle altre comunicazioni nell'arco (i, j). I vincoli (2.13) e (2.14) definiscono le variabili y e u.

- **Disequazioni valide:** questi vincoli aiutano la convergenza del problema. In particolare (2.15) impone che un arco (i, j) possa essere bottleneck per la coppia (s, t) solo se è stato selezionato nel cammino per quella coppia, mentre la disequazione (2.16) impone un bound inferiore alla quantità di flusso di ogni comunicazione.

Capitolo 3

Formulazioni estese e tagli per l'eliminazione di sotto-cicli

In questo capitolo analizziamo il problema dei sotto-cicli accennato nel capitolo precedente e proponiamo diversi approcci ispirandoci a quanto fatto per il problema del commesso viaggiatore (Travelling Salesman Problem – TSP).

Nella Sezione 3.1 descriviamo il problema della formazione di sotto-cicli nella formulazione (2.1)-(2.16) per il problema MMF-CTE e ricordiamo una formulazione di PLI classica per il TSP. Nella Sezione 3.2 presentiamo tre differenti formulazioni estese alternative per evitare la formazione di sotto-cicli. Descriviamo poi, nella Sezione 3.3, un approccio differente, basato sulla generazione iterativa di tagli. Infine, nella Sezione 3.4 riportiamo le scelte effettuate anticipando i risultati ottenuti dal confronto dei diversi approcci descritti.

La scopo di questo capitolo è quello di determinare la formulazione migliore per il problema affrontato in termini di tempi di esecuzione e qualità delle soluzioni trovate.

3.1 Necessità dell'eliminazione di sotto-cicli

Nel problema di instradamento trattato in questa tesi, il fatto che i cammini determinati per le comunicazioni debbano essere privi di cicli e che archi esterni a dato cammino non debbano essere utilizzati sono concetti abbastanza intuitivi dal punto di vista teorico. Però, risolvendo la formulazione parziale (2.1)-(2.16), possono presentarsi soluzioni ammissibili che contengono dei sotto-cicli.

3.1.1 Presenza di sotto-cicli nelle soluzioni del problema MMF-CTE

La ricerca di una soluzione equa per tutte le coppie origine-destinazione tramite il modello parziale (2.1)-(2.16) ammette delle soluzioni che contengono dei sotto-cicli.

Riportiamo in figura 3.1 un esempio grafico della problematica appena descritta. Nonostante la formulazione parziale garantisca che il cammino definito tra una coppia di nodi (s,t) sia privo di cicli, è possibile che la soluzione presenti un ciclo tra due o più nodi esterni al suddetto cammino. Questo accade poiché la presenza di un ciclo esterno al cammino tra s e t, con una quantità di flusso transitante sugli archi del ciclo, Δ, differente dalla quantità ϕ^{st} assegnata alla coppia (s,t), non viola nessun vincolo della formulazione (2.1)-(2.16) e permette di soddisfare i vincoli di presenza di un arco bottleneck tra gli archi del cammino tra quella coppia origine-destinazione (s,t), consentendo alla quantità ϕ^{st} di assumere un valore grande a piacere, violando quindi il principio di MMF.

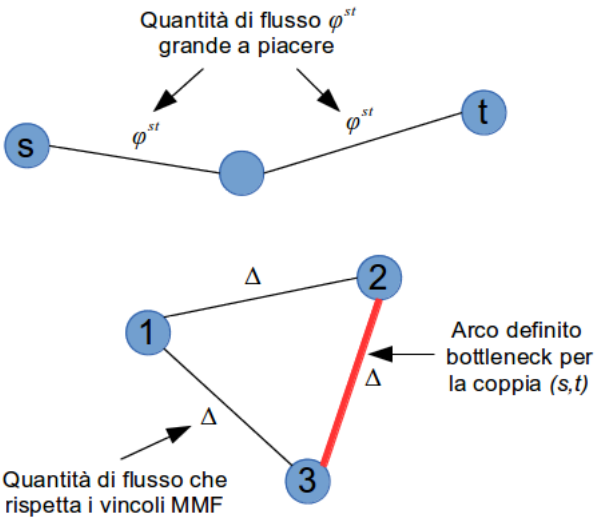

Figura 3.1: Esempio di formazione di sotto-cicli esterni al percorso tra s e t. Sugli archi che congiungono s e t viene fatta transitare la quantità di flusso ϕ^{st} che assume un valore grande a piacere, mentre sugli archi che compongono il ciclo viene fatta transitare una quantità Δ e viene individuato un arco bottleneck, aggirando così gli effetti dei vincoli MMF.

Per evitare ciò, sono stati studiati ed estesi vincoli per la rimozione di sotto-cicli provenienti da formulazioni del problema del commesso viaggiatore (TSP dall'inglese *Travelling Salesman Problem*), opportunamente adattati al problema di assegnazione di flusso a comunicazioni multiple.

3.1.2 Il problema del commesso viaggiatore

Il problema del commesso viaggiatore è uno dei più studiati problemi di ottimizzazione su grafi. Dato un grafo pesato $G = (V, A)$, dove V è l'insieme dei nodi (le città) e A è l'insieme degli archi (le strade), definendo con c_{ij} il costo associato ad ogni arco $(i, j) \in A$, l'obiettivo è quello di determinare il cammino hamiltoniano di costo minimo, vale a dire il ciclo che visiti tutti i nodi del grafo una ed una sola volta, tornando al nodo di partenza.

Un possibile modello di PLI del problema considera una variabile decisionale associata a ciascun arco:

$$x_{ij} = \begin{cases} 1, & \text{se l'arco } (i,j) \text{ è nel ciclo} \\ 0, & \text{altrimenti} \end{cases} \qquad \forall (i,j) \in A, \tag{3.1}$$

ed è così definito:

$$\min \sum_{ij \in A} c_{ij} x_{ij}, \tag{3.2}$$

s.t.

$$\sum_{in \in A} x_{in} = 1 \qquad \forall\ n \in V, \tag{3.3}$$

$$\sum_{nj \in A} x_{nj} = 1 \qquad \forall\ n \in V, \tag{3.4}$$

$$\sum_{(i,j) \in S,\ i \neq j} x_{ij} \leq |M| - 1 \qquad \forall\ M \subset N : \{1\} \notin M, |M| \geq 2. \tag{3.5}$$

La funzione obiettivo minimizza la somma dei costi degli archi selezionati. Ogni nodo deve avere uno ed un solo arco entrante (3.3) e uno ed un solo arco uscente (3.4). I vincoli (3.5) evitano la formazione di sotto-cicli (cicli che non visitano tutti i nodi) nella soluzione e sono detti vincoli di eliminazione dei sotto-cicli.

Prendendo spunto dalle disequazioni (3.5) potremmo sviluppare un insieme di vincoli adatto al nostro problema, evitando così la formazione di sotto-cicli esterni ai cammini definiti per ogni coppia origine-destinazione. Questi vincoli sono però presenti in numero esponenziale, più precisamente $2^{n-1} + n - 1$, con $n = |V|$, e non è quindi pensabile una loro inclusione anche per istanze di medie dimensioni. Pertanto, come riassunto in [14], sono stati proposti differenti formulazioni estese per il TSP che eliminano i sotto-cicli con vincoli alternativi a quelli definiti in (3.5), mantenendo nella formulazione i vincoli (3.3) e (3.4). Queste formulazioni presentano un numero

polinomiale di vincoli e possono quindi essere impiegate per risolvere istanze di maggiori dimensioni.

3.2 Formulazioni estese per il problema MMF-CTE

Prendendo spunto da tre formulazioni estese, abbiamo proposto e confrontato diversi vincoli di eliminazione dei sotto-cicli per il nostro problema MMF-CTE.

In questa sezione presentiamo i vincoli che sono stati considerati ed aggiunti alla formulazione parziale (2.1)-(2.16), distinguendo tre differenti formulazioni estese. Ricordiamo che tutte queste formulazioni sono sufficienti a garantire l'assenza di cicli nelle soluzioni del nostro problema, ma, almeno per il TSP, non sono equivalentemente stringenti.

Per tutte le formulazioni analizzate sono stati mantenuti i vincoli (3.3) e (3.4), riadattati al problema di instradamento di comunicazioni multiple come segue:

$$\sum_{ij \in A} x_{ij}^{st} - \sum_{ji \in A} x_{ji}^{st} = \begin{cases} 1 & \text{se } i = s \\ -1 & \text{se } i = t \\ 0 & \text{altrimenti} \end{cases} \qquad \forall i \in V, (s,t) \in K. \tag{3.6}$$

3.2.1 Formulazione sequenziale

Questa formulazione, per evitare la formazione di sotto-cicli, impone vincoli sulla sequenza in cui vengono visitati i nodi. Definiamo con V_s l'insieme $V - \{s\}$ e con u_i^{st} la variabile che indica la sequenza in cui il nodo i è visitato per la coppia (s,t). Abbiamo inserito i seguenti vincoli:

$$u_j^{st} - u_i^{st} \geq 1 - |V|\,(1 - x_{ij}^{st}) \quad \forall (s,t) \in K, (i,j) \in A : i, j \neq s, \tag{3.7}$$

dove la variabile u_i^{st} gioca un ruolo simile a quello dei potenziali ai nodi, definendo vincoli che eliminano percorsi che non iniziano al nodo s e finiscono al nodo t. Se l'arco (i,j) viene selezionato per la coppia (s,t), $x_{ij}^{st} = 1$, il vincolo impone che la differenza tra i valori u_j^{st} e u_i^{st} sia maggiore o uguale a 1, rispettando così la sequenza di visita.

3.2.2 Formulazione con flusso ausiliario singolo

In questa formulazione il nodo sorgente invia $|V| - 1$ unità di flusso ausiliario lungo la rete. Ogni nodo visitato trattiene un'unità di flusso, finché l'ultima non torna alla

sorgente. In Figura 3.2 è riportato un esempio grafico della distribuzione del flusso in una rete di 5 nodi.

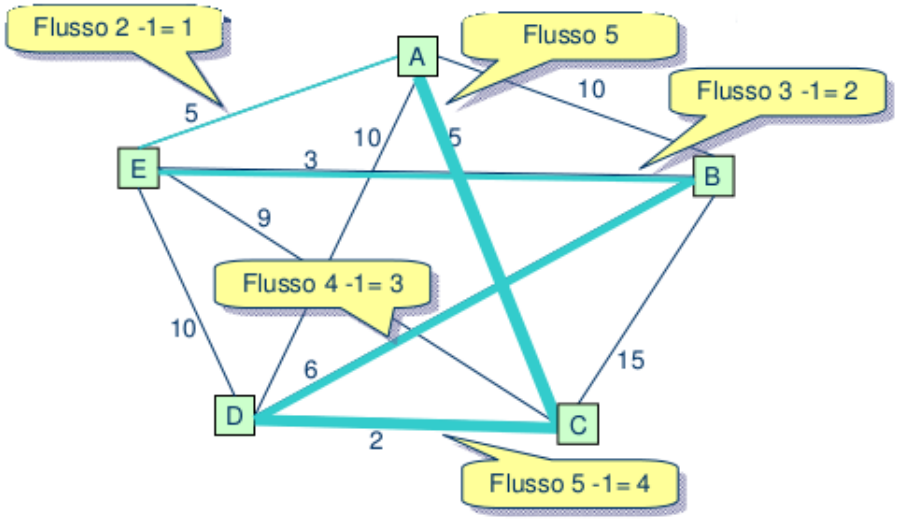

Figura 3.2: Esempio di distribuzione del flusso ausiliario lungo un percorso con 5 nodi. Ad ogni nodo visitato viene decrementata di uno la quantità di flusso ausiliario, in modo da tornare al nodo di partenza senza determinare sotto-cicli.

Per ottenere questo risultato, abbiamo inserito i seguenti vincoli, introducendo la variabile p_{ij}^{st} che indica la quantità di flusso ausiliario presente sull'arco (i, j) relativa alla coppia (s, t):

$$p_{ij}^{st} \leq (|V| - 1)x_{ij}^{st} \quad \forall (i, j) \in A, (s, t) \in K, \tag{3.8}$$

$$\sum_{sj \in A} p_{sj} = \sum_{v \in V} z_v^{st} - 1 \quad \forall (s, t) \in K, \tag{3.9}$$

$$\sum_{ij \in A} p_{ij}^{st} - \sum_{jk \in A} p_{jk}^{st} = z_j^{st} \quad \forall (s, t) \in K, \tag{3.10}$$

$$\sum_{ih \in A} x_{ih}^{st} = z_h^{st} \quad \forall h \in V_s, (s, t) \in K, \tag{3.11}$$

$$z_h^{st} = 0 \quad \forall (s, t) \in K, h \in V : h = s. \tag{3.12}$$

I vincoli (3.8) garantiscono che le unità di flusso transitino solo negli archi selezionati. I vincoli (3.9) e (3.10) impongono che vi siano $|V| - 1$ unità di flusso ausiliare uscenti dal nodo sorgente e che ad ogni nodo ne venga eliminata un'unità. I

vincoli (3.11) definiscono la variabile z_h^{st}, che rappresenta il numero di archi entranti nel nodo h attivati per ogni coppia (s,t).

3.2.3 Formulazione multi-flusso con flussi ausiliari unitari

Questa formulazione è basata anch'essa sull'utilizzo di quantità di flusso ausiliario. Differentemente dalla formulazione con flusso singolo, in questo caso si invia dall'origine s ad ogni nodo intermedio del cammino, definito tra s e la destinazione t corrispondente, un'unità di flusso ausiliario. Vengono quindi utilizzati $|V|-1$ flussi unitari.

Introduciamo la variabile v_{ijh}^{st} che indica la quantità di flusso ausiliario della comunicazione h che transita sull'arco (i,j). I vincoli sono i seguenti:

$$v_{ijh}^{st} \leq x_{ij}^{st} \quad \forall h \in V_s, (s,t) \in K, (i,j) \in A, \tag{3.13}$$

$$\sum_{ih \in A} x_{ih}^{st} = z_h^{st} \quad \forall h \in V_s, (s,t) \in K, \tag{3.14}$$

$$\sum_{ij \in A} v_{ijh}^{st} - \sum_{ji \in A} v_{jih}^{st} = \begin{cases} z_h^{st} & se \quad i = s \\ -z_h^{st} & se \quad i = t \\ 0 & altrimenti \end{cases} \quad \forall h \in V_s, j \in V, (s,t) \in K, \tag{3.15}$$

$$v_{ijh}^{st}, z_h^{st} \geq 0 \quad \forall (i,j) \in A, (s,t) \in K, h \in V_s. \tag{3.16}$$

Il vincolo (3.13) permette il transito di flusso esclusivamente su archi selezionati. Il vincolo (3.14) definisce z_h^{st} che rappresenta il numero di archi entranti nel nodo h attivati per ogni coppia (s,t). Il vincolo (3.15) controlla flusso in ingresso ed in uscita dal nodo per i nodi s e t e garantisce il bilanciamento del flusso negli altri nodi.

3.3 Piani di taglio per l'eliminazione dei sotto-cicli

In questa sezione presentiamo un approccio differente utilizzato per l'eliminazione dei sotto-cicli.

Nelle formulazioni precedenti, i vincoli espressi garantivano che nessun sotto-ciclo si formasse nella soluzione. Questo al costo di un numero elevato (benché polinomiale rispetto alla dimensione dell'istanza) di vincoli inseriti. Vista la difficoltà

di risoluzione delle formulazioni estese, si può adottare un approccio di tipo piani di taglio che consiste nel separare dalla formulazione i vincoli per la rimozione dei sotto-cicli ed aggiungerli in caso di necessità durante il procedimento di *Branch-and-Cut*.

L'idea è quella di risolvere un problema semplificato, dove sono mantenuti esclusivamente i vincoli di bilanciamento definiti in (3.6), in cui è consentita la formazione di cicli. Durante il processo di risoluzione, quando viene trovata una soluzione ammissibile intera in un nodo di branching, questa viene esplorata alla ricerca di eventuali cicli e, per ognuno di essi, viene inserito il relativo vincolo derivante da (3.5), che ne impedisce la formazione. Si risolve così il problema aggiornato e si ripete finché non si raggiunge una soluzione priva di sotto-cicli.

Forniamo ora un semplice esempio per chiarire la procedura appena descritta. Ipotizziamo di trovare nella soluzione un ciclo tra 3 nodi, i, j, h, esterni al percorso tra s e t. L'algoritmo implementato, una volta rilevata la presenza del ciclo, andrà ad aggiungere il seguente vincolo al problema:

$$x_{ij}^{st} + x_{jh}^{st} + x_{hi}^{st} \leq 2, \tag{3.17}$$

dove si impone che la lunghezza del ciclo (numero di archi che lo compongono) deve essere minore o uguale alla lunghezza originale decrementata di uno. In questo modo è garantito che il ciclo tra i nodi i, j e h non si presenterà nelle future soluzioni.

3.4 Conclusioni

In questo capitolo, ispirandoci a quanto fatto in letteratura per il TSP, abbiamo proposto tre formulazioni estese per il nostro problema MMF-CTE. Ricordiamo che sono tutte formulazioni corrette del problema risolto come PLMI, che escludono a priori la formazione di sotto-cicli. I risultati che presenteremo nel Capitolo 6, indicano che la formulazione multi-flusso risulta generalmente migliore delle altre due.

In seguito è stato proposto un approccio alternativo di generazione iterativa di piani di taglio che eliminano i sotto-cicli durante il processo di Branch-and-Cut.

Capitolo 4

Rafforzamento della formulazione estesa multi-flusso

In questo capitolo vengono presentati due nuovi tipi di vincoli introdotti nella formulazione multi-flusso in alternativa a due insiemi di vincoli presenti, nel tentativo di renderla più stringente, migliorando i limiti superiori ed inferiori della soluzione ottima.

In particolare nella Sezione 4.1 introduciamo nuovi vincoli per il bilanciamento del flusso, mentre nella Sezione 4.2 presentiamo nuove disequazioni per la definizione di una soluzione MMF, proponendo sia una formulazione alternativa a quella del modello originale, sia un nuovo insieme di vincoli che rafforzano la formulazione multi-flusso.

4.1 Vincoli di flusso

In questa sezione presentiamo i nuovi vincoli di flusso introdotti.

Nella formulazione di multi-flusso, il vincolo

$$\sum_{ij \in A} f_{ij}^{st} - \sum_{ji \in A} f_{ji}^{st} = \begin{cases} \phi^{st} & if\ i = s \\ -\phi^{st} & if\ i = t \\ 0 & else \end{cases} \quad \forall i \in V, (s,t) \in K, \tag{4.1}$$

garantisce il bilanciamento tra flusso in entrata e flusso in uscita in ogni nodo per ogni coppia di origine-destinazione attraverso la variabile di flusso associata ad ogni arco f_{ij}^{st}. Una possibile formulazione alternativa consiste nell'utilizzare la variabile x_{ij}^{st}, associata ad ogni arco (i,j) per ogni coppia (s,t), per bilanciare gli archi in entrata ed in uscita selezionati ad ogni nodo, considerando le seguenti equazioni

valide:

$$\sum_{ij \in A} x_{ij}^{st} - \sum_{ji \in A} x_{ji}^{st} = \begin{cases} 1 & if\ i = s \\ -1 & if\ i = t \\ 0 & else \end{cases} \qquad \forall i \in V, (s,t) \in K. \tag{4.2}$$

Questi vincoli garantiscono che in ogni nodo intermedio del percorso siano attivati lo stesso numero di archi entranti ed uscenti, limitati ad 1 dal vincolo

$$\sum_{ij \in A} x_{ij}^{st} \leq 1 \qquad \forall i \in V, (s,t) \in K. \tag{4.3}$$

Non viene però imposto nessun limite sulla quantità di flusso che transita su ogni arco. É necessario perciò introdurre un nuovo vincolo di capacità:

$$\sum_{(s,t) \in K} \phi^{st} x_{ij}^{st} \leq c_{ij} \qquad \forall (i,j) \in A, (s,t) \in K, \tag{4.4}$$

che sostituisca l'equazione

$$\sum_{(s,t) \in K} f_{ij}^{st} \leq c_{ij} \qquad \forall (i,j) \in A, \tag{4.5}$$

garantendo così che la somma dei flussi transitanti su ogni arco non ecceda la capacità disponibile dell'arco. Il vincolo appena introdotto è però bilineare, pertanto è necessario linearizzarlo per mantenere la formulazione lineare. Utilizziamo così la variabile di flusso sugli archi f_{ij}^{st} e i corrispondenti vincoli bilineari:

$$f_{ij}^{st} = \phi^{st} x_{ij}^{st} \qquad \forall (i,j) \in A, (s,t) \in K, \tag{4.6}$$

che sono poi linearizzati, attraverso l'inviluppo di McCormick[1] come segue:

[1]Date due variabili x, y dove $x \in [x^L, x^U], y \in [y^L, y^U]$, l'inviluppo di McCormick della superficie bilineare $z = xy$ è definito dalle seguenti disequazioni:

$$z \geq x^L y + x y^L - x^L y^L, \tag{4.7}$$

$$z \geq x^U y + x y^U - x^U y^U, \tag{4.8}$$

$$z \leq x^L y + x y^U - x^L y^U, \tag{4.9}$$

$$z \leq x^U y + x y^L - x^U y^L. \tag{4.10}$$

$$f_{ij}^{st} \geq 0 \qquad \forall (i,j) \in A, (s,t) \in K, \tag{4.11}$$

$$f_{ij}^{st} \geq \phi^{st} - \bar{\phi}^{st}(1 - x_{ij}^{st}) \qquad \forall (i,j) \in A, (s,t) \in K, \tag{4.12}$$

$$f_{ij}^{st} \leq \phi^{st} \qquad \forall (i,j) \in A, (s,t) \in K, \tag{4.13}$$

$$f_{ij}^{st} \leq \bar{\phi}^{st} x_{ij}^{st} \qquad \forall (i,j) \in A, (s,t) \in K, \tag{4.14}$$

dove $\bar{\phi}^{st}$ è un limite superiore di ϕ^{st}.

Per calcolare $\bar{\phi}^{st}$ risolviamo per ogni coppia origine-destinazione un problema di flusso massimo. Ottenuto il valore per ogni comunicazione, lo utilizziamo nei vincoli della linearizzazione descritti precedentemente. Possiamo inoltre rafforzare il vincolo (4.14) nel seguente modo:

$$f_{ij}^{st} \leq \min\{c_{ij}, \bar{\phi}^{st}\} x_{ij}^{st}. \tag{4.15}$$

I vincoli presentati in questa sezione costituiscono un'alternativa ai vincoli di flusso (2.2) (2.3), ma possono essere utilizzati in aggiunta a questi con l'idea di rafforzare la formulazione multi-flusso. In particolare con il vincolo (4.15) si rafforza il Vincolo (2.4) e con (4.12) viene definita una nuova diseguaglianza valida. Nel Capitolo 6 analizzeremo entrambe le possibilità, presentando i risultati computazionali.

4.2 Vincoli MMF

In questa sezione introduciamo alcune disequazioni valide per il nostro problema focalizzando l'attenzione sull'insieme di vincoli che garantiscono una soluzione di Max-Min Fairness.

4.2.1 MMF bound

Grazie alla correttezza dell'algoritmo di waterfilling (spiegato nella Sezione 2.3.2), che aumenta proporzionalmente il valore di ogni flusso fino a saturare un arco, in ogni arco saturato (i,j) utilizzato da k' coppie di origine-destinazione, il flusso per quelle coppie per cui l'arco (i,j) rappresenta un bottleneck è limitato superiormente da

$$\frac{c_{ij} - \sum_{(o,d) \in K} \phi^{od}(1 - y_{ij}^{od})}{\sum_{(o,d) \in K} y_{ij}^{od}}, \tag{4.16}$$

dove $\sum_{(o,d)\in K} \phi^{od}(1 - y_{ij}^{od})$ corrisponde alla somma delle quantità di flusso assegnate alle coppie origine-destinazione (o, d) per cui l'arco (i, j) non è bottleneck, mentre $\sum_{(o,d)\in K} y_{ij}^{od}$ corrisponde al numero di coppie origine-destinazione (o, d) per cui l'arco (i, j) è bottleneck.

Utilizzando (4.16) vogliamo esprimere con delle disuguaglianze lineari che

$$y_{ij}^{st} = 1 \Rightarrow \; f_{ij}^{st} = \frac{c_{ij} - \sum_{(o,d)\in K} f_{ij}^{od}(1 - y_{ij}^{od})}{\sum_{(o,d)\in K} y_{ij}^{od}} \qquad \forall (s,t) \in K, (i,j) \in A. \tag{4.17}$$

Questo implica la seguente diseguaglianza non-lineare, valida per la condizione di MMF:

$$f_{ij}^{st} \leq \frac{c_{ij} - \sum_{(o,d)\in K} f_{ij}^{od}(1 - y_{ij}^{od})}{\sum_{(o,d)\in K} y_{ij}^{od}} \qquad \forall (s,t) \in K, (i,j) \in A. \tag{4.18}$$

La quantità di flusso f_{ij}^{st} viene limitata superiormente dalla capacità dell'arco (i, j) decrementata della somma dei flussi delle origini-destinazioni per cui l'arco non è bottleneck e divisa per il numero delle origini-destinazioni per cui l'arco è bottleneck.

Riscriviamo la disequazione (4.18) nella seguente forma:

$$f_{ij}^{st} \sum_{(o,d)\in K} y_{ij}^{od} \leq c_{ij} - \sum_{(o,d)\in K} f_{ij}^{od} + \sum_{(o,d)\in K} f_{ij}^{od} y_{ij}^{od} \qquad \forall (s,t) \in K, (i,j) \in A. \tag{4.19}$$

Per linearizzare 4.19 si introduce la variabile continua $f_{ij}^{stod} \geq 0$, definita da

$$f_{ij}^{stod} = y_{ij}^{st} f_{ij}^{od}, \tag{4.20}$$

e i vincoli lineari a variabili misto-intere

$$f_{ij}^{stod} \geq 0 \quad \forall (s,t) \in K, (o,d) \in K, (i,j) \in A, \tag{4.21}$$

$$f_{ij}^{stod} \geq f_{ij}^{o,d} - c_{ij}(1 - y_{ij}^{st}) \quad \forall (s,t) \in K, (o,d) \in K, (i,j) \in A, \tag{4.22}$$

$$f_{ij}^{stod} \leq f_{ij}^{od} \quad \forall (o,d) \in K, (i,j) \in A, \tag{4.23}$$

$$f_{ij}^{stod} \leq c_{ij} y_{ij}^{st} \quad \forall (s,t) \in K, (o,d) \in K, (i,j) \in A. \tag{4.24}$$

Un'altra condizione valida per il nostro problema è la seguente:

$$f_{ij}^{st} \geq \frac{c_{ij} - \sum_{(o,d)\in K} f_{ij}^{od}(1 - y_{ij}^{od})}{\sum_{(o,d)\in K} y_{ij}^{od}} y_{ij}^{st} - c_{ij}(1 - y_{ij}^{st}) \quad \forall (s,t) \in K, (i,j) \in A, \tag{4.25}$$

che rappresenta un limite inferiore per la variabile f_{ij}^{st}. Se l'arco (i, j) è bottleneck per la coppia (s, t) ($y_{ij}^{st} = 1$), allora il limite inferiore di f_{ij}^{st} corrisponde esattamente

a

$$f_{ij}^{st} \geq \frac{c_{ij} - \sum_{(o,d)\in K} f_{ij}^{od}(1 - y_{ij}^{od})}{\sum_{(o,d)\in K} y_{ij}^{od}} \qquad \forall (s,t) \in K, (i,j) \in A. \tag{4.26}$$

Se l'arco (i,j) non è bottleneck per la coppia (s,t) ($y_{ij}^{st} = 0$), la variabile f_{ij}^{st} viene limitata inferiormente da una quantità negativa, valore che non ha influenza essendo $f_{ij}^{st} > 0$.

La diseguaglianza (4.25) risulta anch'essa essere non-lineare, di grado superiore alla (4.18). Riscriviamola come segue:

$$f_{ij}^{st} \sum_{(o,d)\in K} y_{ij}^{od} \geq y_{ij}^{st} c_{ij} - y_{ij}^{st} \sum_{(o,d)\in K} f_{ij}^{od} + y_{ij}^{st} \sum_{(o,d)\in K} f_{ij}^{od} y_{ij}^{od} - k c_{ij}(1 - y_{ij}^{st}) \qquad \forall (s,t) \in K, (i,j) \in A. \tag{4.27}$$

La linearizzazione viene effettuata utilizzando la variabile f_{ij}^{stod} ed i relativi vincoli derivanti dalla linearizzazione, ed aggiungendo successivamente la variabile $f_{ij}^{stodbe} \geq 0$ definita come:

$$f_{ij}^{stodbe} = f_{ij}^{stod} y_{ij}^{be}, \tag{4.28}$$

insieme ai seguenti vincoli:

$$f_{ij}^{stodbe} \geq 0 \quad \forall (s,t) \in K, (o,d) \in K, (b,e) \in K, (i,j) \in A, \tag{4.29}$$

$$f_{ij}^{stodbe} \geq f_{ij}^{stod} - c_{ij}(1 - y_{ij}^{be}) \quad \forall (s,t) \in K, (o,d) \in K, (b,e) \in K, (i,j) \in A, \tag{4.30}$$

$$f_{ij}^{stodbe} \leq f_{ij}^{stod} \quad \forall (o,d) \in K, (i,j) \in A, \tag{4.31}$$

$$f_{ij}^{stodbe} \leq c_{ij} y_{ij}^{be} \quad \forall (s,t) \in K, (o,d) \in K, (b,e) \in K, (i,j) \in A. \tag{4.32}$$

Le disuguaglianze valide introdotte fino a questo momento rappresentano rispettivamente un limite superiore (4.18) ed un limite inferiore (4.25) per la variabile f_{ij}^{st}, rispettando il principio di MMF per l'allocazione del flusso. Possono perciò essere utilizzate in aggiunta ai vincoli di MMF originali oppure possono essere utilizzate in una formulazione alternativa mantenendo il vincolo (2.9), per garantire che venga assegnato almeno un arco bottleneck ad ogni comunicazione.

4.2.2 Vincoli sulle capacità residue

In una soluzione MMF, ogni arco bottleneck viene saturato dai flussi delle comunicazioni che lo utilizzano. Vale perciò la condizione:

$$\sum_{(s,t)\in K} f_{ij}^{st} = c_{ij} \qquad \forall (i,j) \in A \quad t.c. \sum_{(s,t)\in K} y_{ij}^{st} > 0, \tag{4.33}$$

che si verifica in tutti gli archi che sono bottleneck per una qualsiasi comunicazione (s,t).

Analogamente, negli archi che non sono bottleneck per nessuna coppia origine-destinazione, viene verificata la seguente condizione:

$$\sum_{(s,t)\in K} f_{ij}^{st} < c_{ij} \qquad \forall (i,j) \in A \quad t.c. \sum_{(s,t)\in K} y_{ij}^{st} = 0. \tag{4.34}$$

Introduciamo la variabile $s_{ij} \in [0, c_{ij}]$, che rappresenta lo scarto o "slack" di un arco, vale a dire la quantità di capacità dell'arco (i,j) che rimane inutilizzata. Possiamo quindi riscrivere le equazioni (4.33) e (4.34) riassumendole nella seguente forma:

$$\sum_{(s,t)\in K} f_{ij}^{st} + s_{ij} = c_{ij} \qquad \forall (i,j) \in A \tag{4.35}$$

in cui il valore assunto dalla variabile di slack condiziona il valore delle variabili di bottleneck degli archi coinvolti. In particolare notiamo che:

$$s_{ij} > 0 \Rightarrow y_{ij}^{st} = 0 \qquad \forall (s,t) \in K. \tag{4.36}$$

Queste considerazioni ci portano a scrivere la seguente disequazione:

$$s_{ij} \sum_{(s,t)\in K} y_{ij}^{st} = 0 \qquad \forall (i,j) \in A, \tag{4.37}$$

che risulta essere sempre valida in quanto la sommatoria delle y è garantita essere uguale a zero nei casi in cui lo slack sia maggiore di zero. Il vincolo (4.37) può anche essere espanso nella seguente formula:

$$s_{ij} y_{ij}^{st} = 0 \qquad \forall (i,j) \in A, (s,t) \in K. \tag{4.38}$$

Essendo anch'esso un vincolo non lineare, per introdurlo nel nostro modello di PLMI procediamo alla linearizzazione. Riscriviamo (4.35) esplicitando il valore di slack:

$$s_{ij} = c_{ij} - \sum_{(s,t)\in K} f_{ij}^{st} \qquad \forall (i,j) \in A. \tag{4.39}$$

Sostituiamo quindi in (4.37):

$$(c_{ij} - \sum_{(s,t)\in K} f_{ij}^{st}) y_{ij}^{st} = 0 \qquad \forall (i,j) \in A. \tag{4.40}$$

Espandiamo il prodotto:

$$y_{ij}^{st} c_{ij} - y_{ij}^{st} \sum_{(s,t)\in K} f_{ij}^{st} = 0 \qquad \forall (i,j) \in A. \tag{4.41}$$

Eseguiamo la linearizzazione utilizzando la variabile f_{ij}^{stod} definita in (4.20) e i corrispettivi Vincoli (4.21)-(4.24).

I nuovi vincoli con la variabile di slack che abbiamo introdotto in questa sezione non costituiscono un'alternativa ai Vincoli MMF originari (2.9)-(2.14). Andremo perciò ad utilizzarli in aggiunta a questi o ai vincoli MMF alternativi introdotti nella sottosezione 4.2.1.

Capitolo 5

Euristiche

In questo capitolo descriviamo i metodi euristici che sono stati sviluppati per il problema MMF-CTE.

Avendo a disposizione l'algoritmo "waterfilling" che, dati i cammini per ogni coppia di origine-destinazione, determina il valore di flusso f_{ij}^{st} su ogni arco (i, j) per ogni coppia (s, t) in modo che la soluzione sia MMF, ci siamo concentrati sullo sviluppo di metodi per la scelta di un cammino per ogni coppia di origine-destinazione.

Dopo aver definito nella Sezione 5.1 i vantaggi apportati dall'aggiunta di un algoritmo euristico, presentiamo nella Sezione 5.2 i due metodi proposti per il calcolo dei cammini. Un primo approccio utilizza il famoso algoritmo di Dijkstra opportunamente riadattato al nostro problema. Una seconda tecnica si basa sull'arrotondamento del valore delle variabili x_{ij}^{st} che definiscono i cammini nella soluzione ottima del rilassamento continuo.

5.1 Algoritmi euristici

Un algoritmo euristico è un qualunque metodo in grado di fornire una soluzione ammissibile del problema considerato. Generalmente un algoritmo euristico richiede un tempo di calcolo polinomiale nella dimensione dell'istanza che deve risolvere, ma per problemi complessi si può rilassare questo vincolo, richiedendo che l'algoritmo sia sufficientemente "veloce".

L'algoritmo euristico "ideale" dovrebbe essere in grado di fornire sempre la soluzione ottima di un problema. Questo è quello che accade per alcuni problemi che hanno una particolare struttura (ad esempio l'algoritmo greedy per il problema della determinazione di un albero di supporto di costo minimo). Per problemi più complessi si richiede che l'algoritmo euristico sia in grado di determinare sempre una

"buona" soluzione ammissibile, dove buona significa una soluzione il cui valore sia abbastanza vicino al valore di una soluzione ottima.

Esistono però una serie di problemi per i quali la determinazione di una soluzione ammissibile è un problema NP-difficile (ad esempio il problema del TSP). In questi casi non è neanche detto che l'algoritmo euristico sia in grado di determinare una soluzione ammissibile.

In generale, data un'istanza I di un problema di massimizzazione, quale risulta essere il problema MMF-CTE, un algoritmo euristico A fornisce una soluzione di valore $z^A(I)$ tale che

$$z^A(I) \leq z^*(I), \tag{5.1}$$

dove $z^*(I)$ rappresenta il valore ottimo dell'istanza. In pratica l'algoritmo fornisce un limite inferiore sul valore di una soluzione ottima.

5.2 Scelta dei cammini

L'algoritmo waterfilling descritto nella Sezione 2.3 necessita di due elementi di input oltre al grafo $G = (V, A)$:

- le capacità c_{ij} per ogni arco $(i, j) \in A$, dato già in nostro possesso;
- un cammino definito per ogni coppia origine-destinazione, che deve invece essere determinato.

In questa sezione presentiamo due metodi che permettono di definire un'insieme ragionevole di cammini da impiegare con l'algoritmo di waterfilling. Un primo metodo può essere utilizzato a prescindere dal tipo di formulazione utilizzata, in particolare con qualsiasi insieme di vincoli per l'eliminazione di sotto-cicli, poiché costruisce da zero i cammini. Il secondo metodo, basato su tecniche di arrotondamento di soluzioni del rilassamento continuo, non può essere utilizzato in combinazione al metodo di generazione iterativa di piani di taglio per la rimozione dei sotto-cicli, descritto nella sezione 3.3.

5.2.1 Cammini poco sovrapposti

La quantità di flusso f_{ij}^{st} che viene assegnata a ciascuna coppia di origine destinazione (s, t) nell'arco (i, j) è vincolata dalla capacità del suddetto arco e dal numero di coppie che vi transitano. Maggiore il numero di coppie di origine-destinazione che vengono fatte transitare sull'arco (i, j), minore sarà la quantità della capacità c_{ij} messa a disposizione di ciascuna coppia e, di conseguenza, minore sarà la quantità

di flusso f_{ij}^{st} assegnata ad ogni coppia (s,t). Riteniamo perciò sensato determinare un cammino per ogni coppia origine-destinazione cercando di sovrapporli il meno possibile, in modo da utilizzare al meglio gli archi disponibili, evitando di avere archi troppo congestionati ed archi inutilizzati. Per fare ciò abbiamo adattato opportunamente al nostro problema l'algoritmo di Dijkstra.

L'algoritmo di Dijkstra è un algoritmo utilizzato per determinare i cammini minimi in un grafo con pesi sugli archi non negativi. Dato un grafo $G = (V, A)$ ed un nodo origine $s \in V$, l'algoritmo determina il cammino di costo minimo tra s ed ogni altro nodo $t \in V$ (con $t \neq s$). I principi di funzionamento dell'algoritmo sono i seguenti:

- ad ogni nodo è associata un'etichetta che rappresenta il costo del cammino migliore trovato per raggiungerlo dal nodo origine;
- l'algoritmo etichetta progressivamente i nodi partendo dall'origine. Il nodo successivo è quello raggiungibile a costo più basso a partire da un nodo già etichettato;
- l'algoritmo termina quando a tutti i nodi è stata associata un'etichetta.

Nel nostro caso, dato che desideriamo cercare il cammino con minore congestione, abbiamo utilizzato come peso di un arco il numero di coppie origine-destinazione che transitano su di esso. Questo valore deve essere aggiornato ogni volta che viene definito un cammino per una coppia di origine-destinazione.

Descriviamo il funzionamento dell'algoritmo per il calcolo dei cammini disgiunti:

- **Passo 1:** ogni arco (i, j) viene inizializzato con un peso $p_{ij} = 0$, che rappresenta il numero di comunicazioni che stanno transitando su di esso.
- **Passo 2:** si considera la prima coppia (s, t) per cui bisogna determinare il cammino. Viene eseguito l'algoritmo di Dijkstra ricercando il percorso di costo minimo tra s e t.
- **Passo 3:** vengono aggiornati i pesi degli archi, incrementando di un'unità il valore degli archi appartenenti al cammino appena definito.
- **Passo 4:** si ripetono i Passi 2 e 3 fino a che non viene definito un cammino per ogni coppia di origine-destinazione.

Alla fine dell'esecuzione dell'algoritmo avremo un insieme di cammini abbastanza disgiunti tra loro. Ovviamente maggiore il numero di origini-destinazioni da instra-

dare, più sarà facile che queste condividano archi, con conseguente aumento della congestione.

Visto che i cammini trovati dipendono dall'ordine in cui vengono considerate le coppie di origine-destinazione, l'algoritmo viene eseguito un numero predefinito di volte, ciascuna corrispondente ad un ordine casuale. In questo modo si ottengono insiemi di cammini differenti e viene selezionato quello con il miglior valore della funzione obiettivo.

5.2.2 Tecniche di arrotondamento

Introduciamo ora un secondo metodo implementato per determinare un cammino per ogni coppia origine-destinazione.

Una tipologia di euristiche ampiamente diffusa in problemi di ottimizzazione lineare intera è quella che si basa su tecniche di arrotondamento, le cosiddette *Rounding heuristics*. L'idea è quella di utilizzare una soluzione ottima del rilassamento continuo (problema di PL ottenuto omettendo i vincoli di interezza sulle variabili intere), proveniente dal nodo corrente dell'albero di branching, per applicare tecniche di arrotondamento sui valori frazionari delle variabili.

Data una soluzione del rilassamento continuo corrente con x_{ij}^{st} frazionari, il processo di arrotondamento ci consente di ottenere dei valori interi per le variabili x_{ij}^{st}, definendo quindi un cammino per ogni coppia origine-destinazione. Descriviamo ora i diversi passaggi dell'algoritmo.

- **Passo 1:** quando in un nodo di branching viene trovata una soluzione ottima del rilassamento continuo, tramite una procedura di *callback* si recuperano i valori relativi alla variabile x_{ij}^{st}.
- **Passo 2:** per ogni coppia (s, t) si procede poi nel seguente modo:
  - viene generato un grafo pesato utilizzando i valori frazionari dalla variabile x_{ij}^{st}. I pesi di ogni arco (i, j) sono determinati campionando casualmente un valore dalla distribuzione uniforme definita come $U(0, 1 - x_{ij}^{st})$, ovvero il valore frazionario di x_{ij}^{st} viene utilizzato per determinare il limite superiore dell'intervallo di campionamento.
  - Nel grafo pesato si ricerca un cammino minimo tra s e t tramite l'algoritmo di Dijkstra.
- **Passo 3:** definiti i cammini per tutte le coppie origine-destinazione dell'insieme K, si determina il vettore di allocazione tramite l'algoritmo di waterfilling,

definendo così per ogni coppia (s, t) e per ogni arco (i, j) il valore della variabile f_{ij}^{st}.

Partendo da una soluzione frazionaria proveniente dal rilassamento continuo, siamo così in grado di determinare un cammino per ogni coppia origine-destinazione, definendo una soluzione ammissibile per il problema MMF-CTE. La complessità computazionale dell'algoritmo è molto bassa e possiamo quindi effettuare più esecuzioni partendo dalla stessa soluzione ottima del rilassamento continuo, in modo da selezionare la miglior soluzione trovata in termini di funzione obiettivo. Inoltre l'euristica può essere eseguita in più nodi dell'albero di branching, utilizzando come base per l'arrotondamento soluzioni del rilassamento continuo che tendono a migliorare nel tempo.

Capitolo 6

Risultati computazionali

In questo Capitolo presentiamo i risultati computazionali in cui confrontiamo le varianti di formulazioni estese e i metodi proposti nei Capitoli 3, 4 e 5.

Nelle Sezioni 6.1 e 6.2 descriviamo rispettivamente le istanze e gli strumenti informatici utilizzati per realizzare le prove computazionali. Nella Sezione 6.3 confrontiamo i risultati ottenuti applicando le diverse tecniche per l'eliminazione dei sotto-cicli presentati nel Capitolo 3. Nella Sezione 6.4 analizziamo i vincoli alternativi proposti nel Capitolo 4. Infine, nella Sezione 6.5 presentiamo i risultati ottenuti applicando gli algoritmi euristici proposti nel Capitolo 5.

6.1 Istanze

Le prove computazionali sono state realizzate utilizzando alcune topologie di reti estratte dalla *SND Library* [15]. Questa libreria raccoglie un insieme di istanze relative a problemi di reti fisse, con lo scopo di fornire uno standard per la validazione di modelli e algoritmi di ottimizzazione esatti ed euristici.

In particolare abbiamo considerato quattro reti (`polska, abilene, atlanta, geant`) per generare le nostre istanze, in modo da avere topologie di differente complessità in termini di numero di nodi e archi. Ogni rete è stata completata generando un insieme di capacità degli archi, scegliendo casualmente per ogni arco una dimensione tra 2 Gbps, 2,4 Gbps, 5 Gbps e 8 Gbps, e 5 insiemi di coppie origine-destinazione K, di cardinalità crescente. Riportiamo in Tabella 6.1 l'elenco delle istanze generate con i dati relativi alla topologia della rete e al numero di coppie origine-destinazione, e nelle Figure 6.1, 6.2, 6.3 e 6.4 le topologie delle reti utilizzate.

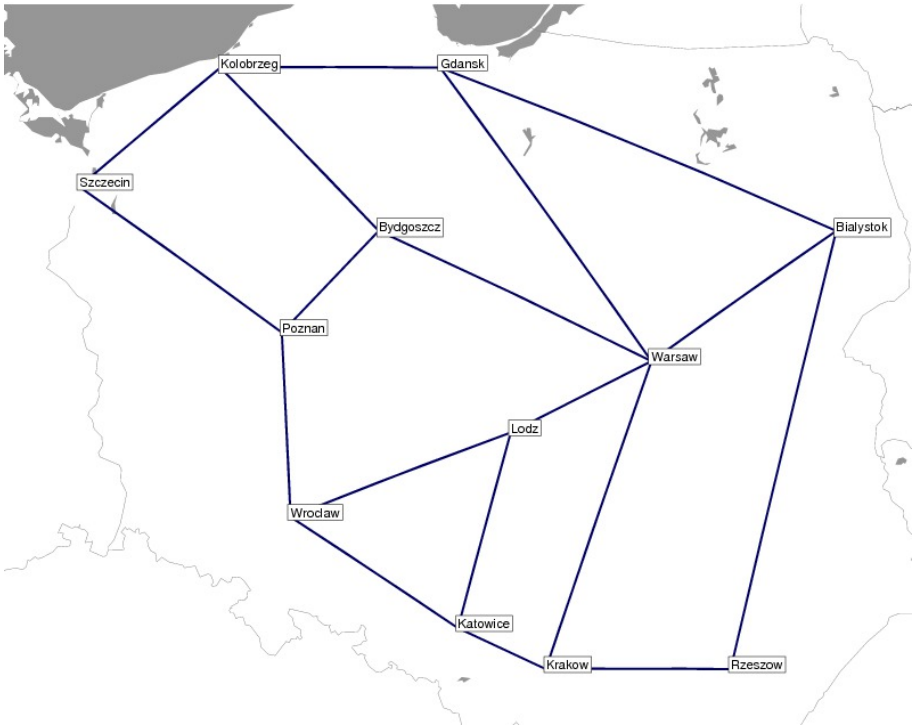

Figura 6.1: Rappresentazione della rete `polska`

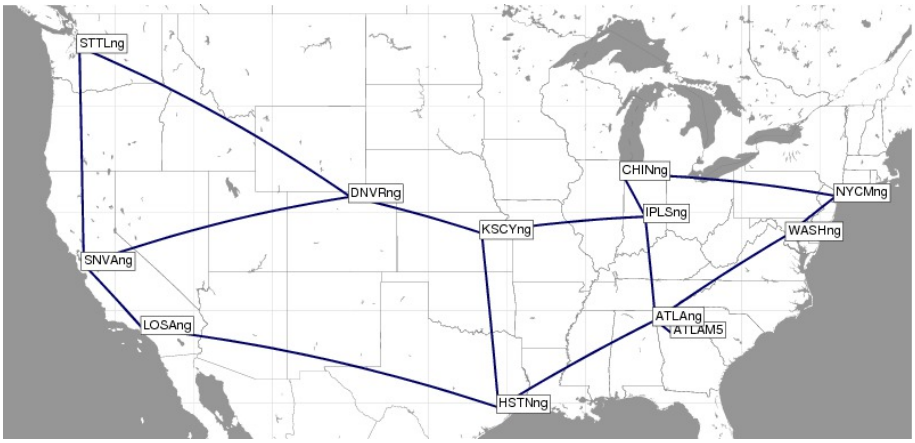

Figura 6.2: Rappresentazione della rete `abilene`

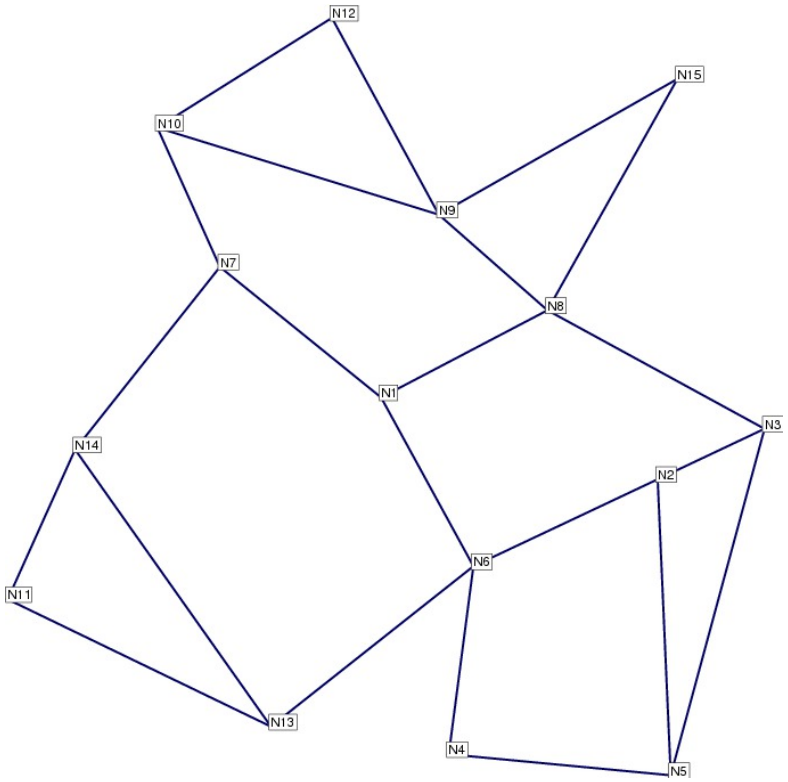

Figura 6.3: Rappresentazione della rete `atlanta`

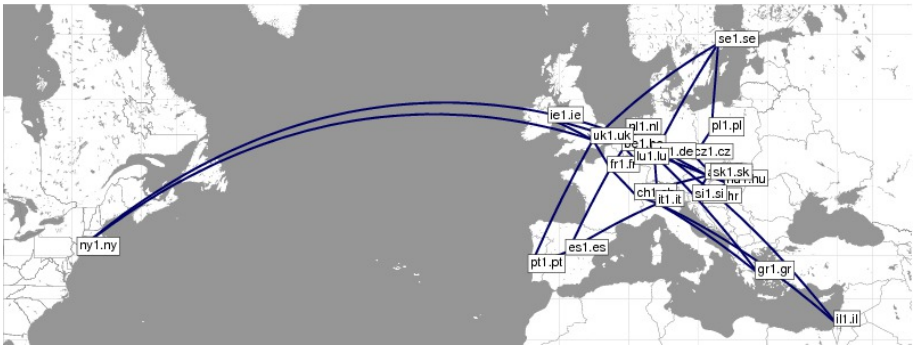

Figura 6.4: Rappresentazione della rete `geant`

Tabella 6.1: Dati relativi alle istanze generate

rete	$\|V\|$	$\|A\|$	$\|K\|$
`abilene`	12	30	12
`abilene`	12	30	20
`abilene`	12	30	30
`abilene`	12	30	42
`abilene`	12	30	56
`polska`	12	36	10
`polska`	12	36	21
`polska`	12	36	28
`polska`	12	36	36
`polska`	12	36	45
`atlanta`	15	44	12
`atlanta`	15	44	20
`atlanta`	15	44	30
`atlanta`	15	44	42
`atlanta`	15	44	56
`geant`	22	72	12
`geant`	22	72	20
`geant`	22	72	30
`geant`	22	72	42
`geant`	22	72	56

6.2 Implementazione

Le formulazioni e i metodi proposti sono stati risolti tramite il risolutore di PLMI *Gurobi 5.5* interfacciato con il linguaggio *python*. Il linguaggio *python*, oltre ad essere pienamente integrato con Gurobi, presenta interessanti strutture dati, come i *Dizionari*, che permettono di gestire in modo semplice e funzionale insiemi di dati, rivelandosi ideale sia per la definizione delle formulazioni che degli algoritmi.

Le varianti di formulazioni estese per la rimozione dei sotto-cicli, le varianti con vincoli alternativi di bilanciamento del flusso e vincoli MMF e le varianti con tagli aggiuntivi, sono state definite tramite l'interfaccia Python fornita da Gurobi. L'approccio per l'eliminazione di sotto-cicli con generazione di tagli è stato inserito nel contesto dell'algoritmo di Branch-and-Cut di Gurobi tramite una funzione di *Callback*, che viene chiamata periodicamente dal risolutore durante il processo di ottimizzazione. La funzione di Callback, attraverso il parametro *where*, permette di controllare da che punto della risoluzione debba essere avviata. Quando viene ritrovata una soluzione intera (*where* = *MIPSOL*), vengono avviati gli algoritmi per la ricerca dei sotto-cicli nella soluzione. Le disequazioni necessarie sono aggiunte alla

formulazione grazie all'utilizzo dei *Lazy Constraints*. Per le euristiche di ricerca dei cammini poco sovrapposti e di arrotondamento, è stata implementata una funzione di Callback che viene attivata quando il risolutore sta esplorando un nodo dell'albero di branching (*where* = *MIPNODE*). Le soluzioni euristiche trovate vengono fornite al risolutore che le completa determinando i valori delle variabili mancanti.

Le prove computazionali sono state effettuate su una macchina equipaggiata con un processore Intel Xeon E5645 (6 core, 12 thread) e 16 GB di RAM, imponendo un tempo massimo di 3600 secondi per ogni esecuzione ed assegnando un singolo thread tramite il parametro `Threads` di Gurobi, impostato a 1.

Per ulteriori dettagli relativi al codice rimandiamo all'Appendice A.

6.3 Valutazione vincoli di eliminazione dei sotto-cicli

In questa sezione presentiamo i risultati computazionali relativi ai modelli proposti nel Capitolo 3. L'obiettivo è quello di verificare qual è la formulazione estesa migliore per il problema MMF-CTE tra le tre proposte e se l'approccio di generazione iterativa di piani di taglio è efficace come nel TSP. La funzione obiettivo (2.1) viene massimizzata con i pesi $w^{st} = 1$ per ogni coppia $(s, t) \in K$.

6.3.1 Confronto vincoli di eliminazione dei sotto-cicli

Analizziamo i risultati ottenuti applicando alla formulazione (2.2)-(2.16) i vincoli estesi per l'eliminazione di sotto-cicli descritti nella Sezione 3.2 del Capitolo 3.

A questo proposito, distinguiamo tre differenti formulazioni estese:

- *sequential*, che comprende i vincoli (3.6), (3.7) della formulazione sequenziale;
- *single-flow*, che comprende i vincoli (3.6), (3.8)-(3.12) della formulazione mono-flusso;
- *multi-flow*, che comprende i Vincoli (3.13)-(3.16) della formulazione multi-flusso.

In Tabella 6.2 sono riportati i risultati ottenuti, distinguendo la formulazione e le istanze utilizzate. Le colonne Ψ, *time*, *gap %* e Ψ_{up} indicano rispettivamente la miglior soluzione ammissibile trovata, il tempo di esecuzione in secondi, il gap percentuale calcolato tra la miglior soluzione ottenuta nell'intervallo di tempo e il miglior limite superiore calcolato da Gurobi e il miglior limite superiore.

Tabella 6.2: Risultati relativi alle tre formulazioni estese per l'eliminazione dei sotto-cicli.

		sequential				single-flow				multi-flow			
rete	$\|K\|$	Ψ	time	gap%	Ψ_{up}	Ψ	time	gap%	Ψ_{up}	Ψ	time	gap%	Ψ_{up}
pol	10	30600,0	2,9	0,0	30600,0	30600,0	5,0	0,0	30600,0	30600,0	11,5	0,0	30600,0
pol	21	46057,1	3600,0	1,6	46772,1	46057,1	3600,0	1,0	46502,8	46057,1	3600,0	1,1	46553,0
pol	28	-	-	-	53703,0	-	-	-	53703,0	52563,6	3600,0	2,0	53632,0
pol	36	-	-	-	63650,0	61457,1	3600,0	3,4	63540,0	61457,1	3600,0	3,5	63591,7
pol	45	-	-	-	71760,0	-	-	-	71672,0	-	-	-	71672,0
abi	12	26800,0	1,0	0,0	26800,0	26800,0	1,0	0,0	26800,0	26800,0	0,7	0,0	26800,0
abi	20	31250,0	3600,0	11,4	34800,0	34800,0	14,5	0,0	34800,0	34800,0	9,5	0,0	34800,0
abi	30	49600,0	3600,0	1,2	50215,5	49655,6	827,4	0,0	49660,4	49655,6	94,9	0,0	49659,5
abi	42	56218,2	3600,0	0,4	56459,8	56218,2	949,0	0,0	56223,7	56218,2	776,3	0,0	56223,8
abi	56	-	-	-	75350,0	70093,2	3600,0	2,9	72139,8	70931,3	3600,0	0,8	71529,1
atl	12	38200,0	4,5	0,0	38200,0	38200,0	5,8	0,0	38200,0	38200,0	10,0	0,0	38200,0
atl	20	54600,0	3600,0	2,4	55927,3	54600,0	3600,0	2,0	55666,9	54600,0	3600,1	1,1	55200,0
atl	30	-	-	-	70734,0	69600,0	3600,0	1,3	70528,5	69600,0	3600,0	0,7	70100,0
atl	42	-	-	-	87012,0	-	-	-	87012,0	-	-	-	87012,0
atl	56	-	-	-	89160,0	-	-	-	89124,7	-	-	-	90101,5
gea	12	35600,0	810,0	0,0	35600,0	35600,0	169,0	0,0	35600,0	35600,0	544,1	0,0	35600,0
gea	20	-	-	-	50200,0	-	-	-	51358,3	48800,0	3600,0	8,6	52994,5
gea	30	-	-	-	74200,0	-	-	-	74200,0	-	-	-	74200,0
gea	42	-	-	-	99200,0	-	-	-	99200,0	-	-	-	99200,0
gea	56	-	-	-	123600,0	-	-	-	123600,0	-	-	-	123600,0
media				1,9				0,3				0,2	

I risultati indicano che la formulazione *sequential* risulta avere più difficoltà nel trovare una soluzione per tutte le istanze, lasciandone più di metà irrisolte. Confrontando invece le altre due formulazioni, si vede che, per le istanze risolte da entrambi, i due valori di Ψ sono quasi sempre uguali, ma la formulazione *multi-flow* riporta tendenzialmente un limite superiore migliore, presentando, per le istanze risolte da entrambi, un gap medio di $0,6\%$ contro lo $0,9\%$ della formulazione *single-flow*. Inoltre la formulazione *multi-flow* presenta due istanze risolte in più rispetto a quella *single-flow*, risultando essere la formulazione più adatta per il problema MMF-CTE.

6.3.2 Confronto con separazione dei vincoli di eliminazione dei sotto-cicli

Presentiamo ora il confronto tra la formulazione *multi-flow* e la formulazione implementata separando i Vincoli (3.5), descritta nella Sezione 3.3, che indichiamo con il nome *subtour*.

I risultati riportati in Tabella 6.3 indicano che la strategia di separazione dei vincoli di eliminazione dei sotto-cicli risulta poco efficace nella risoluzione del nostro problema. Molto significativo è il caso delle istanze relative alla rete `geant`, che sono rimaste tutte irrisolte, insieme alla quasi totalità delle istanze con $|K| \geq 20$, fatta eccezione per le istanze della rete `abilene`.

Questo fatto è particolarmente interessante poiché rivela un comportamento opposto a quanto accade con il problema del TSP. Infatti l'approccio di generazione iterativa di piani di taglio risulta molto efficiente per il TSP e per vari problemi di ottimizzazione combinatoria. Nel nostro caso, invece, la grossa difficoltà incontrata con le istanze di maggiori dimensioni ci porta ad escludere questa alternativa. Probabilmente questo approccio fallisce poiché la formulazione considerata, ammettendo soluzioni contenenti sotto-cicli, non aiuta l'euristica risolutiva di Gurobi a trovare una buona soluzione ammissibile per il problema MMF-CTE. Infatti, l'utilizzo dei *"Lazy constraints"* per la rimozione dei sotto-cicli fa si che molte delle soluzioni primali trovate da Gurobi diventino inammissibili.

6.3.3 Considerazioni finali

Il primo obiettivo di questa tesi era verificare se fosse più efficace un approccio basato sulla generazione iterativa di tagli per l'eliminazione di sotto-cicli oppure una delle tre formulazioni estese descritte nel Capitolo 3.

Dai risultati riportati in questa sezione è emerso che la formulazione estesa *multi-flow* risulta essere la più efficace per eliminare i sotto-cicli, dimostrandosi

Tabella 6.3: Confronto tra le formulazioni *multi-flow* e *subtour*. La formulazione con generazione iterativa di piani di taglio si è rivelata meno efficace di quella multi-flow.

		multi-flow				subtour			
rete	$\|K\|$	Ψ	time	gap%	Ψ_{up}	Ψ	time	gap%	Ψ_{up}
pol	10	30600,0	11,5	0,0	30600,0	30600,0	14,2	0,0	30600,0
pol	21	46057,1	3600,0	1,1	46553,0	46057,1	3600,0	1,1	46561,4
pol	28	52563,6	3600,0	2,0	53632,0	-	-	-	53703,0
pol	36	61457,1	3600,0	3,5	63591,7	-	-	-	63594,7
pol	45	-	-	-	71672,0	-	-	-	71760,0
abi	12	26800,0	0,7	0,0	26800,0	26800,0	1,0	0,0	26800,0
abi	20	34800,0	9,5	0,0	34800,0	34800,0	107,5	0,0	34800,0
abi	30	49655,6	94,9	0,0	49659,5	49655,6	3103,2	0,0	49660,5
abi	42	56218,2	776,3	0,0	56223,8	56218,2	1529,8	0,0	56223,8
abi	56	70931,3	3600,0	0,8	71529,1	-	-	-	75097,0
atl	12	38200,0	10,0	0,0	38200,0	38200,0	2,1	0,0	38200,0
atl	20	54600,0	3600,1	1,1	55200,0	54600,0	3600,0	1,6	55449,1
atl	30	69600,0	3600,0	0,7	70100,0	-	-	-	70792,8
atl	42	-	-	-	87012,0	-	-	-	87012,0
atl	56	-	-	-	90101,5	-	-	-	89160,0
gea	12	35600,0	544,1	0,0	35600,0	-	-	-	35600,0
gea	20	48800,0	3600,0	8,6	52994,5	-	-	-	52200,0
gea	30	-	-	-	74200,0	-	-	-	74200,0
gea	42	-	-	-	99200,0	-	-	-	99200,0
gea	56	-	-	-	123600,0	-	-	-	123600,0

migliore delle altre formulazioni estese e, sopratutto, dell'approccio di generazione iterativa di piani di talgio, differentemente da quanto accade per il TSP.

In base a queste considerazioni, i vincoli per l'eliminazione di sotto-cicli di tipo *multi-flow* saranno utilizzati in tutte le formulazioni che verranno considerate.

6.4 Valutazione vincoli alternativi

In questa Sezione analizziamo le formulazioni alternative proposte nel Capitolo 4. Presentiamo quindi i risultati relativi all'inserimento dei vincoli alternativi di bilanciamento del flusso, di MMF e di capacità residue.

L'obiettivo è quello di verificare se la formulazione estesa multi-flusso può essere ulteriormente migliorata con l'inserimento di questi tipi di disuguaglianze valide.

6.4.1 Vincoli di flusso alternativi

Consideriamo i vincoli alternativi di bilanciamento del flusso proposti nella Sezione 4.1. Definiamo così la formulazione *flow-bis*, che corrisponde alla formulazione *multi-*

Tabella 6.4: Confronto tra le formulazioni *multi-flow* e *flow-bis* La formulazione *multi-flow* si rivela migliore sia in termini di istanze risolte sia in termini di gap medio, presentando, per le istanze risolte da entrambi, un valore medio dello 0.3% contro un 3,2% della formulazione *flow-bis*.

		multi-flow				flow-bis				
rete	$\|K\|$	Ψ	time	gap%	Ψ_{up}	Ψ	time	gap%	Ψ_{up}	Ψ'_{up}
pol	10	30600,0	11,5	0,0	30600,0	30600,0	12,5	0,0	30600,0	0,0%
pol	21	46057,1	3600,0	1,1	46553,0	46057,1	3600,0	7,4	49470,2	6,3%
pol	28	52563,6	3600,0	2,0	53632,0	-	-	-	63672,3	18,7%
pol	36	61457,1	3600,0	3,5	63591,7	-	-	-	74969,9	17,9%
pol	45	-	-	-	71672,0	-	-	-	91241,3	27,3%
abi	12	26800,0	0,7	0,0	26800,0	26800,0	11,5	0,0	26801,1	0,0%
abi	20	34800,0	9,5	0,0	34800,0	34800,0	327,7	0,0	34800,0	0,0%
abi	30	49655,6	94,9	0,0	49659,5	49655,6	3600,0	2,2	50768,7	2,2%
abi	42	56218,2	776,3	0,0	56223,8	55733,3	3600,0	7,0	59626,8	6,1%
abi	56	70931,3	3600,0	0,8	71529,1	-	-	-	75392,9	5,4%
atl	12	38200,0	10,0	0,0	38200,0	38200,0	120,8	0,0	38200,0	0,0%
atl	20	54600,0	3600,1	1,1	55200,0	54266,7	3600,0	9,0	59170,8	7,2%
atl	30	69600,0	3600,0	0,7	70100,0	-	-	-	82615,9	17,9%
atl	42	-	-	-	87012,0	-	-	-	109679,6	26,1%
atl	56	-	-	-	90101,5	-	-	-	102467,8	13,7%
gea	12	35600,0	544,1	0,0	35600,0	35600,0	1647,8	0,0	35600,0	0,0%
gea	20	48800,0	3600,0	8,6	52994,5	-	-	-	51607,7	-2,6%
gea	30	-	-	-	74200,0	-	-	-	83015,6	11,9%
gea	42	-	-	-	99200,0	-	-	-	124545,4	25,5%
gea	56	-	-	-	123600,0	-	-	-	158217,0	28,0%
media				0,3				3,2		11%

flow a cui sostituiamo i vincoli (2.2) aggiungendo i vincoli (4.2), (4.12), (4.13), (4.15).

Il valore di $\bar{\phi}$ (limite superiore sul flusso) viene calcolato come segue. Per ogni coppia (s,t) risolviamo il relativo problema di flusso massimo sulla rete in uso e memorizziamo il risultato ottenuto in $\bar{\phi}^{st}$. Visto che il tempo di esecuzione di ogni problema di flusso massimo è in media inferiore a 0.1 s, non influisce sui tempi totali di risoluzione.

I risultati del confronto tra la formulazione *flow-bis* e la formulazione *multi-flow* sono riportati in Tabella 6.4. La formulazione con i nuovi vincoli di bilanciamento del flusso presenta un minor numero di istanze risolte, 5 in meno della formulazione *multi-flow*. Inoltre, osservando i valori del miglior limite superiore calcolato, la formulazione *flow-bis* presenta valori più elevati in media dell'11%, come riportato nella colonna Ψ'_{up}, dimostrando una maggiore difficoltà di convergenza.

I risultati ottenuti con la formulazione *flow-bis* indicano che la nuova famiglia di vincoli risulta più pesante di quelli standard della formulazione *multi-flow*. Questo

è probabilmente dovuto all'elevato numero di vincoli derivanti dalla linearizzazione. Volendo però trovare nuovi piani di taglio validi per migliorare la convergenza della formulazione, aggiungiamo alternativamente i nuovi vincoli alla formulazione *multi-flow* per verificare se abbiano un effetto stringente in termini di limite superiore. Distinguiamo in particolare due formulazioni:

- *flow A*, formulazione *multi-flow* con in aggiunta i Vincoli (4.2);
- *flow B*, formulazione *multi-flow* con in aggiunta i Vincoli (4.12), (4.13), (4.15).

Prima di effettuare un confronto dei risultati computazionali ottenuti a time-limit, valutiamo l'eventuale grado di rafforzamento delle disuguaglianze aggiuntive confrontando i valori relativi alle soluzioni del rilassamento continuo eseguito al nodo radice. I valori della funzione obiettivo del rilassamento continuo per ogni istanza relativi ai modelli *multi-flow*, *flow A* e *flow B* sono riportati in Tabella 6.5. I valori trovati dal rilassamento della formulazione *flow B* si dimostrano interessanti. In particolare, se si osservano tutte le istanze della rete `atlanta`, le prime due della rete `geant` e la prima della rete `polska`, si può notare come i rispettivi valori siano minori di quelli della formulazione *multi-flow*. Analizziamo quindi in maniera più approfondita i vincoli aggiuntivi presenti nella formulazione *flow B* per valutare una possibile inclusione nella formulazione *multi-flow*.

Per avere il minor onere computazionale aggiuntivo, analizziamo singolarmente il contributo dei Vincoli (4.12), (4.13), (4.15) alla formulazione *multi-flow*. In Tabella 6.6 sono riportati i rispettivi valori delle soluzioni del rilassamento continuo al nodo radice. Le due colonne relative alla formulazione *multi-flow* alla quale sono stati aggiunti rispettivamente i Vincoli (4.12) e (4.15) presentano risultati positivi in corrispondenza delle istanze citate precedentemente, riportando valori delle soluzioni del rilassamento continuo minori di quelle relative alla formulazione *multi-flow*, restringendo quindi la formulazione. Consideriamo queste due varianti per effettuare le prove computazionali complete.

La Tabella 6.7 riporta i risultati computazionali relativi alle due formulazioni individuate, includendo anche la formulazione *multi-flow* per confronto. Le due formulazioni proposte risolvono rispettivamente, entro il limite di tempo, 1 e 2 istanze in meno rispetto alla formulazione *multi-flow*, senza presentare miglioramenti in termini di migliore soluzione trovata. Si nota inoltre che i valori del limite superiore non confermano la forza dei vincoli aggiuntivi dimostrata a livello di rilassamento continuo su un buon numero di istanze.

Pare quindi non facile migliorare la formulazione *multi-flow* aggiungendo tagli validi più stringenti. Probabilmente l'onere computazionale che ne consegue influen-

Tabella 6.5: Valori delle soluzioni del rilassamento continuo relativo ai modelli *multi-flow*, *flow A* e *flow B*. La formulazione *flow B* presenta risultati minori per quasi la metà delle istanze, dimostrando una potenziale forza stringente. Un valore positivo del gap% indica un miglioramento del valore del rilassamento continuo.

		multi-flow	flow A		flow B	
rete	$\|K\|$	Ψ relax	Ψ relax	gap%	Ψ relax	gap%
polska	10	34200,0	34200,0	0,0	31206,2	9,6
polska	21	46820,0	46820,0	0,0	46820,0	0,0
polska	28	53703,0	53703,0	0,0	53703,0	0,0
polska	36	63650,0	63650,0	0,0	63650,0	0,0
polska	45	71760,0	71760,0	0,0	71760,0	0,0
abilene	12	27760,0	27760,0	0,0	27760,0	0,0
abilene	20	35760,0	35760,0	0,0	35760,0	0,0
abilene	30	51600,0	51600,0	0,0	51600,0	0,0
abilene	42	59742,0	59742,0	0,0	59742,0	0,0
abilene	56	75420,0	75420,0	0,0	75420,0	0,0
atlanta	12	39809,1	39719,4	0,2	39400,0	1,0
atlanta	20	57200,0	57200,0	0,0	56480,0	1,3
atlanta	30	73072,0	73072,0	0,0	72048,1	1,4
atlanta	42	89224,0	89224,0	0,0	88080,3	1,3
atlanta	56	91210,0	91210,0	0,0	90241,0	1,1
geant	12	46200,0	46200,0	0,0	38371,7	20,4
geant	20	56200,0	56200,0	0,0	52971,7	6,1
geant	30	74200,0	74200,0	0,0	74200,0	0,0
geant	42	99200,0	99200,0	0,0	99200,0	0,0
geant	56	123600,0	123600,0	0,0	123600,0	0,0
media				0,0		2,1

za negativamente il processo di risoluzione, annullando di fatto gli effetti positivi che si ottengono a livello di rilassamento continuo.

Tabella 6.6: Valori delle soluzioni del rilassamento continuo relativo alla formulazione *multi-flow* con l'aggiunta dei Vincoli (4.12), (4.13), (4.15) separatamente. Un valore positivo del gap% indica un miglioramento del valore trovato.

		multi-flow	multi-flow + (4.12)		multi-flow + (4.13)		multi-flow + (4.15)	
rete	$\|K\|$	Ψ relax	Ψ relax	gap%	Ψ relax	gap%	Ψ relax	gap%
polska	10	34200,0	31206,2	9,6	34200,0	0,0	31533,3	8,5
polska	21	46820,0	46820,0	0,0	46820,0	0,0	46820,0	0,0
polska	28	53703,0	53703,0	0,0	53703,0	0,0	53703,0	0,0
polska	36	63650,0	63650,0	0,0	63650,0	0,0	63650,0	0,0
polska	45	71760,0	71760,0	0,0	71760,0	0,0	71760,0	0,0
abilene	12	27760,0	27760,0	0,0	27760,0	0,0	27760,0	0,0
abilene	20	35760,0	35760,0	0,0	35760,0	0,0	35760,0	0,0
abilene	30	51600,0	51600,0	0,0	51600,0	0,0	51600,0	0,0
abilene	42	59742,0	59742,0	0,0	59742,0	0,0	59742,0	0,0
abilene	56	75420,0	75420,0	0,0	75420,0	0,0	75420,0	0,0
atlanta	12	39809,1	39400,0	1,0	39809,1	0,0	39400,0	1,0
atlanta	20	57200,0	56480,0	1,3	57200,0	0,0	56480,0	1,3
atlanta	30	73072,0	72048,1	1,4	73072,0	0,0	72048,1	1,4
atlanta	42	89224,0	88080,3	1,3	89224,0	0,0	88080,3	1,3
atlanta	56	91210,0	90241,0	1,1	91210,0	0,0	90241,0	1,1
geant	12	46200,0	38371,7	20,4	46200,0	0,0	39600,0	16,7
geant	20	56200,0	52971,7	6,1	56200,0	0,0	54200,0	3,7
geant	30	74200,0	74200,0	0,0	74200,0	0,0	74200,0	0,0
geant	42	99200,0	99200,0	0,0	99200,0	0,0	99200,0	0,0
geant	56	123600,0	123600,0	0,0	123600,0	0,0	123600,0	0,0
media				2,1		0,0		1,7

Tabella 6.7: Confronto tra le formulazioni *multi-flow* e le sue due varianti con l'aggiunta dei Vincoli (4.12) e (4.15). I vincoli aggiuntivi non portano lo stesso miglioramento riscontrato a livello di rilassamento continuo.

		multi-flow				multi-flow + (3.9)				multi-flow + (3.11)			
rete	$\|K\|$	Ψ	time	gap%	Ψ_{up}	Ψ	time	gap%	Ψ_{up}	Ψ	time	gap%	Ψ_{up}
polska	10	30600,0	11,5	0,0	30600,0	30600,0	8,2	0,0	30600,0	30600,0	8,1	0,0	30600,9
polska	21	46057,1	3600,0	1,1	46553,0	46057,1	3600,0	1,5	46725,0	46057,1	3600,0	0,9	46486,2
polska	28	52563,6	3600,0	2,0	53632,0	52563,6	3600,0	2,2	53701,7	-	-	-	53703,0
polska	36	61457,1	3600,0	3,5	63591,7	-	-	-	63595,0	-	-	-	63650,0
polska	45	-	-	-	71672,0	-	-	-	71707,6	-	-	-	71672,0
abilene	12	26800,0	0,7	0,0	26800,0	26800,0	1,1	0,0	26800,0	26800,0	1,0	0,0	26800,0
abilene	20	34800,0	9,5	0,0	34800,0	34800,0	17,5	0,0	34800,0	34800,0	9,3	0,0	34800,0
abilene	30	49655,6	94,9	0,0	49659,5	49655,6	147,7	0,0	49660,4	49655,6	248,8	0,0	49660,4
abilene	42	56218,2	776,3	0,0	56223,8	56218,2	1070,5	0,0	56223,6	56218,2	1118,3	0,0	56223,8
abilene	56	70931,3	3600,0	0,8	71529,1	70931,3	3600,0	1,6	72038,5	70093,2	3600,0	3,0	72222,2
atlanta	12	38200,0	10,0	0,0	38200,0	38200,0	14,4	0,0	38200,0	38200,0	8,2	0,0	38200,0
atlanta	20	54600,0	3600,1	1,1	55200,0	54200,0	3600,0	3,1	55900,0	54600,0	3600,0	0,5	54866,7
atlanta	30	69600,0	3600,0	0,7	70100,0	69600,0	3600,0	1,2	70426,9	69600,0	3600,0	1,5	70668,0
atlanta	42	-	-	-	87012,0	-	-	-	87012,0	-	-	-	87012,0
atlanta	56	-	-	-	90101,5	-	-	-	89145,7	-	-	-	89121,8
geant	12	35600,0	544,1	0,0	35600,0	35600,0	236,4	0,0	35600,0	35600,0	217,1	0,0	35600,0
geant	20	48800,0	3600,0	8,6	52994,5	-	-	-	50840,4	-	-	-	50200,0
geant	30	-	-	-	74200,0	-	-	-	74200,0	-	-	-	74200,0
geant	42	-	-	-	99200,0	-	-	-	99200,0	-	-	-	99200,0
geant	56	-	-	-	123600,0	-	-	-	123600,0	-	-	-	123600,0
media				0,3				0,7				0,5	

6.4.2 Vincoli MMF alternativi

Prendiamo ora in considerazione i vincoli MMF alternativi proposti nella Sezione 4.2. Definiamo la formulazione *mmf-bis*, costituita dalla formulazione *multi-flow* con i Vincoli (4.19) e (4.27), opportunamente linearizzati, inseriti al posto dei vincoli (2.9)-(2.14).

In Tabella 6.8 sono riportati i risultati computazionali relativi alle due formulazioni. La formulazione *mmf-bis* risulta molto difficile da risolvere. Infatti solo per le istanze più piccole, 4 su 20, è stata trovata una soluzione ammissibile ed in tempi di gran lunga superiori rispetto alla formulazione *multi-flow*. Guardando poi la colonna *nodes*, che riporta il numero di nodi dell'albero di branching esplorati durante la risoluzione, emerge il fatto che in più della metà delle istanze il risolutore non è andato oltre al nodo radice. In 3 istanze non è stato risolto neanche il rilassamento continuo al nodo radice dopo un'ora di esecuzione.

In base alle precedenti osservazioni, escludiamo la possibilità di includere nella formulazione *multi-flow* i nuovi vincoli MMF che, visti i precedenti risultati con i vincoli di flusso aggiuntivi, comprometterebbero l'efficienza del risolutore apportando un onere computazionale eccessivo.

6.4.3 Vincoli sulle capacità residue

Analizziamo ora l'effetto dei vincoli sulle capacità residue presentati nella Sezione 4.2. A tale proposito definiamo due differenti formulazioni:

- *slack A*, modello *multi-flow* più Vincoli (4.38);
- *slack B*, modello *multi-flow* più Vincoli (4.37).

I risultati riportati in Tabella 6.9, mostrano che entrambe le formulazioni alternative proposte sono più complesse della formulazione *multi-flow*. In particolare la formulazione *slack B* riporta la soluzione per sole 5 istanze, mantenendo irrisolte tutte quelle della rete `geant`.

Questi dati confermano la difficoltà incontrata nel migliorare la formulazione *multi-flow* imponendo nuove diseguaglianze valide.

Tabella 6.8: Confronto tra le formulazioni *multi-flow* e *mmf-bis*. La formulazione con i vincoli mmf alternativi risulta eccessivamente complessa, lasciando irrisolte la quasi totalità delle istanze.

		multi-flow				mmf-bis			
rete	$\|K\|$	Ψ	time	gap%	Ψ_{up}	Ψ	time	gap%	Ψ_{up}
polska	10	30600,0	11,5	0,0	30600,0	30600,0	1036,8	0,0	30600,0
polska	21	46057,1	3600,0	1,1	46553,0	-	-	-	46820,0
polska	28	52563,6	3600,0	2,0	53632,0	-	-	-	53703,0
polska	36	61457,1	3600,0	3,5	63591,7	-	-	-	63650,0
polska	45	-	-	-	71672,0	-	-	-	71760,0
abilene	12	26800,0	0,7	0,0	26800,0	26800,0	18,1	0,0	26800,0
abilene	20	34800,0	9,5	0,0	34800,0	-	-	-	34800,0
abilene	30	49655,6	94,9	0,0	49659,5	-	-	-	51600,0
abilene	42	56218,2	776,3	0,0	56223,8	-	-	-	59742,0
abilene	56	70931,3	3600,0	0,8	71529,1	-	-	-	-
atlanta	12	38200,0	10,0	0,0	38200,0	38200,0	124,2	0,0	38200,0
atlanta	20	54600,0	3600,1	1,1	55200,0	-	-	-	56200,0
atlanta	30	69600,0	3600,0	0,7	70100,0	-	-	-	72006,0
atlanta	42	-	-	-	87012,0	-	-	-	89224,0
atlanta	56	-	-	-	90101,5	-	-	-	-
geant	12	35600,0	544,1	0,0	35600,0	35200,0	3600,0	22,7	43200,0
geant	20	48800,0	3600,0	8,6	52994,5	-	-	-	56200,0
geant	30	-	-	-	74200,0	-	-	-	74200,0
geant	42	-	-	-	99200,0	-	-	-	99200,0
geant	56	-	-	-	123600,0	-	-	-	-

Tabella 6.9: Confronto tra le formulazioni *multi-flow*, *slack A* e *slack B*. I vincoli sulle capacità residue introdotti appesantiscono la formulazione *multi-flow*, limitando il numero di istanze risolte.

		multi-flow				slack A				slack B			
rete	$\|K\|$	Ψ	time	gap%	Ψ_{up}	Ψ	time	gap%	Ψ_{up}	Ψ	time	gap%	Ψ_{up}
polska	10	30600,0	11,5	0,0	30600,0	30600,0	40,6	0,0	30600,0	30600,0	256,9	0,0	30600,0
polska	21	46057,1	3600,0	1,1	46553,0	-	-	-	46820,0	-	-	-	46820,0
polska	28	52563,6	3600,0	2,0	53632,0	-	-	-	53703,0	-	-	-	53703,0
polska	36	61457,1	3600,0	3,5	63591,7	-	-	-	63650,0	-	-	-	63650,0
polska	45	-	-	-	71672,0	-	-	-	71710,1	-	-	-	71756,3
abilene	12	26800,0	0,7	0,0	26800,0	26800,0	3,3	0,0	26800,0	26800,0	11,1	0,0	26800,0
abilene	20	34800,0	9,5	0,0	34800,0	34800,0	130,8	0,0	34800,0	34800,0	781,0	0,0	34800,0
abilene	30	49655,6	94,9	0,0	49659,5	49655,6	1893,9	0,0	49655,6	-	-	-	51546,9
abilene	42	56218,2	776,3	0,0	56223,8	52866,7	3600,0	12,8	59648,0	-	-	-	59742,0
abilene	56	70931,3	3600,0	0,8	71529,1	-	-	-	75420,0	-	-	-	75420,0
atlanta	12	38200,0	10,0	0,0	38200,0	38200,0	37,3	0,0	38200,0	38200,0	219,3	0,0	38200,0
atlanta	20	54600,0	3600,1	1,1	55200,0	54200,0	3600,0	3,3	56000,0	53933,3	3600,0	4,2	56200,0
atlanta	30	69600,0	3600,0	0,7	70100,0	67200,0	3603,1	5,4	70847,3	-	-	-	72020,3
atlanta	42	-	-	-	87012,0	-	-	-	88052,7	-	-	-	88058,0
atlanta	56	-	-	-	90101,5	-	-	-	90219,7	-	-	-	91210,0
geant	12	35600,0	544,1	0,0	35600,0	35600,0	3216,7	0,0	35600,0	-	-	-	44700,0
geant	20	48800,0	3600,0	8,6	52994,5	42800,0	3600,0	31,3	56200,0	-	-	-	56200,0
geant	30	-	-	-	74200,0	-	-	-	74200,0	-	-	-	74200,0
geant	42	-	-	-	99200,0	-	-	-	99200,0	-	-	-	99200,0
geant	56	-	-	-	123600,0	-	-	-	123600,0	-	-	-	123600,0

6.4.4 Considerazioni finali

I risultati computazionali presentati in questa sezione si sono purtroppo rivelati non in linea con le aspettative. L'obiettivo era quello di esplorare vincoli alternativi alla ricerca di disuguaglianze valide che migliorassero la formulazione *multi-flow*.

I vincoli alternativi di flusso ci hanno consentito di includere alcune disuguaglianze abbastanza stringenti da migliorare il rilassamento continuo al nodo radice per un buon numero di istanze, senza però dimostrarsi altrettanto efficaci in fase risolutiva del problema misto-intero.

In base ai risultati ottenuti, per migliorare le soluzioni ammissibili della formulazione *multi-flow*, decidiamo di impiegare metodi euristici che aiutino la convergenza durante il procedimento di Branch-and-Bound, in modo da fornire limiti inferiori buoni di una soluzione ottima.

6.5 Valutazione dell'impatto delle euristiche

Presentiamo in questa sezione i risultati ottenuti applicando le euristiche introdotte nel Capitolo 5 all'algoritmo di Branch-and-Bound durante la risoluzione della formulazione *multi-flow*.

Quello che ci chiediamo è se un approccio euristico possa giocare un ruolo significativo in termini di dimensione dell'istanza che si riesce a risolvere, affrontando in modo particolare le istanze più grosse a cui la formulazione *multi-flow* non è riuscita a fornire una soluzione ammissibile.

6.5.1 Euristica cammini poco sovrapposti

Prendiamo ora in considerazione l'euristica che mira ad ottenere un insieme di cammini con un basso grado di sovrapposizione. Chiamiamo *E-disg* la variante della formulazione *multi-flow* che utilizza questa euristica per generare una soluzione ammissibile al nodo radice. Per ogni istanza abbiamo eseguito l'euristica 10 volte al nodo radice, fornendo 10 ordinamenti differenti delle coppie origine-destinazione in K, generati aleatoriamente. È stata poi fornita a Gurobi la soluzione con il maggior valore della funzione obiettivo.

Sottolineiamo il fatto che al risolutore vengono forniti i valori delle variabili x_{ij}^{st} e f_{ij}^{st} determinate tramite l'euristica. Le altre variabili della formulazione *multi-flow* sono automaticamente calcolate dal risolutore partendo dai valori forniti, in modo da definire una soluzione completa.

In Tabella 6.10 sono riportati con i risultati ottenuti con la formulazione *E-disg* messi a confronto con quelli della formulazione *multi-flow*. Per la formulazione con

euristica è presente la colonna Ψ_0, che riporta il valore trovato dall'euristica al nodo radice, e la colonna *gap* Ψ_0, nella quale troviamo il gap percentuale tra la soluzione iniziale Ψ_0 e quella finale riportata in Ψ, inteso come miglioramento ottenuto a fine risoluzione.

Per la prima volta, grazie all'aggiunta del metodo euristico al nodo radice, siamo stati in grado di trovare una soluzione ammissibile per tutte le istanze. Per le istanze che sono state risolte con entrambe le formulazioni, di cui viene riportato il gap medio, la formulazione *multi-flow* presenta un valore leggermente migliore, così come per le sole istanze con $|K| \leq 20$, per le quali anche i tempi di esecuzione sono più ristretti. Osservando invece i gap medi per le istanze con $|K| > 20$, dove nelle istanze non risolte è stato considerato correttamente un gap del 100%, notiamo quanto sia abissale la differenza tra le due formulazioni, presentando un valore del 50,59% contro un 6,60% in favore della soluzione che impiega l'euristica al nodo radice.

Visti questi risultati, riteniamo che l'euristica risolutiva di Gurobi, quando funziona, è abbastanza efficace, ma si rivela poco utile negli altri casi. La nostra euristica, invece, guida il risolutore verso una buona porzione dell'albero di branching, in cui è presenta una soluzione ammissibile, permettendo di ottenere buoni risultati anche con istanze più grosse. Infatti, osservando le colonne E_0 e *gap* E_0, si nota come sia determinante fornire una soluzione ammissibile al risolutore al nodo radice, poiché aiuterà il processo di Branch-and-Bound, definendo un miglioramento medio della soluzione di circa il 20%.

Tabella 6.10: Confronto tra la formulazione *multi-flow* e la formulazione *E-disg* in cui viene utilizzata l'euristica per il calcolo dei cammini poco sovrapposti al nodo radice. I risultati evidenziano un notevole miglioramento in termini di miglior soluzione trovata che deriva dall'introduzione dell'euristica.

		multi-flow				e-disg					
rete	$\|K\|$	Ψ	time	gap%	Ψ_{up}	Ψ	time	gap%	Ψ_{up}	Ψ_0	gap%
pol	10	30600,0	11,5	0,0	30600,0	30600,0	14,0	0,0	30600,0	28200,0	8,51
pol	21	46057,1	3600,0	1,1	46553,0	46057,1	3600,0	1,0	46535,0	40133,3	14,76
pol	28	52563,6	3600,0	2,0	53632,0	52563,6	3600,0	1,8	53502,6	45466,7	15,61
pol	36	61457,1	3600,0	3,5	63591,7	61473,3	3600,0	3,5	63646,6	52730,0	16,58
pol	45	-	-	100,0	71672,0	69714,3	3600,0	2,7	71628,0	60950,0	14,38
abi	12	26800,0	0,7	0,0	26800,0	26800,0	0,8	0,0	26800,0	22200,0	20,72
abi	20	34800,0	9,5	0,0	34800,0	34800,0	4,2	0,0	34800,0	28133,3	23,70
abi	30	49655,6	94,9	0,0	49659,5	49655,6	381,3	0,0	49660,3	44766,7	10,92
abi	42	56218,2	776,3	0,0	56223,8	56218,2	1030,2	0,0	56223,7	45604,0	23,27
abi	56	70931,3	3600,0	0,8	71529,1	69701,3	3600,0	4,8	73081,3	59442,3	17,26
atl	12	38200,0	10,0	0,0	38200,0	38200,0	11,9	0,0	38200,0	29800,0	28,19
atl	20	54600,0	3600,1	1,1	55200,0	54600,0	3600,0	1,1	55176,9	42200,0	29,38
atl	30	69600,0	3600,0	0,7	70100,0	69600,0	3600,0	1,4	70568,0	54233,3	28,33
atl	42	-	-	100,0	87012,0	84733,3	3600,2	2,7	87012,0	64405,3	31,56
atl	56	-	-	100,0	90101,5	79363,6	3600,0	13,7	90201,2	62059,0	27,88
gea	12	35600,0	544,1	0,0	35600,0	35600,0	1991,3	0,0	35600,0	32200,0	10,56
gea	20	48800,0	3600,0	8,6	52994,5	47400,0	3600,1	13,4	53733,3	38800,0	22,16
gea	30	-	-	100,0	74200,0	68700,0	3600,0	8,0	74200,0	55800,0	23,12
gea	42	-	-	100,0	99200,0	88900,0	3600,0	11,6	99200,0	74050,0	20,05
gea	56	-	-	100,0	123600,0	95893,3	3600,0	28,9	123600,0	89213,3	7,49
media				30,89				4,73			19,72
$K \leq 20$				1,35				1,93			
$K > 20$				50,59				6,60			

6.5.2 Euristica di arrotondamento

Consideriamo ora l'euristica di arrotondamento presentata nella Sottosezione 5.2.2 e definiamo come *Round root* la formulazione *multi-flow* che la utilizza al nodo radice. Per ogni istanza il processo di arrotondamento viene lanciato 10 volte in modo da selezionare la soluzione migliore in termini di funzione obiettivo.

In Tabella 6.11 sono riportati i risultati computazionali relativi alle formulazioni *multi-flow* e *Round root*. Come accaduto impiegando l'euristica dei cammini poco sovrapposti, anche la formulazione che utilizza l'euristica di arrotondamento al nodo radice presenta una soluzione ammissibile per tutte le istanze. I gap medi relativi alle istanze risolte da entrambi le formulazioni e a quelle con $|K| \leq 20$ mostrano una maggior efficacia della formulazione senza euristica per istanze di piccole dimensioni, a fronte anche di tempi di risoluzione più ristretti. Invece, osservando le istanze con $|K| > 20$, emerge l'importanza di utilizzare un approccio euristico in grado di fornire una soluzione ammissibile al processo di Branch-and-Cut del risolutore, portando ad una riduzione del gap medio delle soluzioni finali dal 50,59% della formulazione *multi-flow* al 6,35%.

L'euristica di arrotondamento si è dimostrata efficace almeno quanto quella per il calcolo dei cammini poco sovrapposti, riportando un gap medio leggermente inferiore per le istanze piccole, ma leggermente superiore per quelle grandi.

Tabella 6.11: Confronto tra la formulazione *multi-flow* e la formulazione *Round root* che utilizza l'euristica di arrotondamento al nodo radice.

		multi-flow				round root					
rete	$\|K\|$	Ψ	time	gap%	Ψ_{up}	Ψ	time	gap%	Ψ_{up}	Ψ_0	gap%
pol	10	30600,0	11,5	0,0	30600,0	30600,0	10,7	0,0	30600,0	25800,0	18,6
pol	21	46057,1	3600,0	1,1	46553,0	46057,1	3600,0	1,5	46725,0	42533,3	8,3
pol	28	52563,6	3600,0	2,0	53632,0	52563,6	3600,0	2,2	53703,0	46033,3	14,2
pol	36	61457,1	3600,0	3,5	63591,7	61432,0	3600,0	3,4	63540,0	55088,9	11,5
pol	45	-	-	100,0	71672,0	69711,5	3600,0	2,7	71628,0	58764,7	18,6
abi	12	26800,0	0,7	0,0	26800,0	26800,0	1,0	0,0	26800,0	21866,7	22,6
abi	20	34800,0	9,5	0,0	34800,0	34800,0	4,1	0,0	34800,0	29200,0	19,2
abi	30	49655,6	94,9	0,0	49659,5	49655,6	123,7	0,0	49660,4	48100,0	3,2
abi	42	56218,2	776,3	0,0	56223,8	56218,2	1132,5	0,0	56223,8	45688,9	23,0
abi	56	70931,3	3600,0	0,8	71529,1	70572,5	3600,0	1,9	71900,7	60405,6	16,8
atl	12	38200,0	10,0	0,0	38200,0	38200,0	11,8	0,0	38200,0	35600,0	7,3
atl	20	54600,0	3600,1	1,1	55200,0	54600,0	3600,0	1,5	55400,0	46266,7	18,0
atl	30	69600,0	3600,0	0,7	70100,0	69600,0	3600,0	1,5	70668,0	58080,0	19,8
atl	42	-	-	100,0	87012,0	84500,0	3600,0	3,0	87012,0	73373,3	15,2
atl	56	-	-	100,0	90101,5	76614,1	3603,1	16,9	89530,5	59752,7	28,2
gea	12	35600,0	544,1	0,0	35600,0	35600,0	661,9	0,0	35600,0	27600,0	29,0
gea	20	48800,0	3600,0	8,6	52994,5	48400,0	3600,0	13,9	55120,0	31000,0	56,1
gea	30	-	-	100,0	74200,0	61900,0	3603,1	19,9	74200,0	45350,0	36,5
gea	42	-	-	100,0	99200,0	89503,3	3600,0	10,8	99200,0	80050,0	11,8
gea	56	-	-	100,0	123600,0	108580,0	3600,1	13,8	123600,0	103580,0	4,8
media				30,89				4,65		media	19,14
$K \leq 20$				1,35				2,10			
$K > 20$				50,59				6,35			

6.5.3 Confronto tra i due metodi euristici

Le due euristiche proposte si sono rivelate entrambe efficaci nell'aiutare il risolutore ad esplorare una buona porzione dell'albero di branching. Le piccole differenze riportate in termini di valore della funzione obiettivo trovata al nodo radice possono essere trascurate considerando che entrambi i metodi, nel generare i cammini, sono soggetti ad una componente aleatoria (ordine in cui vengono considerate le coppie origine-destinazione per l'euristica dei cammini poco sovrapposti, pesi degli archi campionati sulla distribuzione $U(0, 1 - x_{ij}^{st})$ per l'euristica di arrotondamento). Infatti, osservando i risultati riportati in Tabella 6.12, in cui compaiono i valori trovati da entrambe le euristiche al nodo radice ed il loro rapporto (colonna *ratio*), non è possibile stabilire che un metodo sia migliore dell'altro per una singola esecuzione.

La vera differenza risiede nel fatto che l'euristica di arrotondamento utilizza, come base di partenza per la generazione aleatoria dei pesi del grafo, una soluzione frazionaria proveniente del rilassamento continuo in nodo di branching. Questa tende a migliorare nel tempo, portandoci a considerare che un'esecuzione multipla dell'euristica, anche in nodi successivi a quello radice, possa aiutarci a trovare soluzioni ammissibili migliori in termini di valore della funzione obiettivo.

Viste queste considerazioni, eseguiamo alcune prove variando la frequenza con cui l'euristica di arrotondamento viene chiamata. In particolare definiamo le formulazioni *Round 10*, *Round 100* e *Round 1000* in cui l'euristica di arrotondamento viene eseguita rispettivamente ogni 10, 100, 1000 nodi dell'albero di branching.

Riportiamo i risultati ottenuti in Tabella 6.13, indicando nella colonna *exec* il numero di esecuzioni dell'euristica nell'arco dell'intervallo temporale ed includendo anche i dati relativi alla formulazione *Round root* per effettuare un confronto. I gap medi delle tre formulazioni che eseguono più volte l'euristica di arrotondamento si abbassano sensibilmente rispetto alla formulazione che utilizza l'euristica esclusivamente al nodo radice, passando da valori medi maggiori del 4% a valori inferiori al 3%. La differenza è ancor più sostanziale se si prendono in considerazione le sole istanze con $|K| > 20$, dove si passa da un gap medio del 6,3% della formulazione *Round root* ad un valore del 3,1% della formulazione *Round 10*. L'utilizzo ripetuto dell'euristica di arrotondamento aiuta l'algoritmo di Branch-and-Cut del risolutore, migliorando la qualità delle soluzioni ammissibile trovate, in particolar modo per le istanze di maggiori dimensioni.

Osservando invece il numero di esecuzioni dell'euristica durante l'intero processo di risoluzione, riportato nella colonna *exec* per ciascuna formulazione, si nota come questo, in apparenza, non influisca proporzionalmente sulla qualità della funzione obiettivo. Valutiamo quindi l'andamento del valore della funzione obiettivo e

Tabella 6.12: Confronto tra le soluzioni fornite al nodo radice dalle euristiche impiegate nei modelli *E-disg* e *Round root*.

		E-disg	round root	
rete	$\|K\|$	Ψ_0	Ψ_0	ratio
polska	10	28200,0	25800,0	1,09
polska	21	40133,3	42533,3	0,94
polska	28	45466,7	46033,3	0,99
polska	36	52730,0	55088,9	0,96
polska	45	60950,0	58764,7	1,04
abilene	12	22200,0	21866,7	1,02
abilene	20	28133,3	29200,0	0,96
abilene	30	44766,7	48100,0	0,93
abilene	42	45604,0	45688,9	1,00
abilene	56	59442,3	60405,6	0,98
atlanta	12	29800,0	35600,0	0,84
atlanta	20	42200,0	46266,7	0,91
atlanta	30	54233,3	58080,0	0,93
atlanta	42	64405,3	73373,3	0,88
atlanta	56	62059,0	59752,7	1,04
geant	12	32200,0	27600,0	1,17
geant	20	38800,0	31000,0	1,25
geant	30	55800,0	45350,0	1,23
geant	42	74050,0	80050,0	0,93
geant	56	89213,3	103580,0	0,86
media				1,00
$K \leq 20$				1,02
$K > 20$				0,98

del limite superiore nel tempo.

Tabella 6.13: Risultati ottenuti variando la frequenza di esecuzione dell'euristica di arrotondamento.

		round root			round 10					round 100					round1000				
rete	$\|K\|$	Ψ	time	gap%	Ψ	time	gap%	Ψ_{up}	exec	Ψ	time	gap%	Ψ_{up}	exec	Ψ	time	gap%	Ψ_{up}	exec
pol	10	30600	10,7	0,0	30600	13,6	0,0	30600	24	30600	20,5	0,0	30600	6	30600	14,9	0,0	30600	1
pol	21	46057	3600,0	1,5	46057	3600,0	0,9	46466	10821	46057	3600,0	1,0	46535	1505	46057	3600,0	0,9	46487	133
pol	28	52564	3600,0	2,2	52564	3600,0	2,0	53632	5400	52564	3600,0	2,2	53703	537	52564	3600,0	2,0	53632	51
pol	36	61432	3600,0	3,4	61432	3600,0	2,8	63151	6522	61432	3600,0	2,7	63078	587	61702	3600,0	3,0	63540	59
pol	45	69712	3600,0	2,7	69579	3600,1	2,8	71536	3365	69712	3600,1	2,7	71581	346	69712	3600,0	2,6	71495	50
abi	12	26800	1,0	0,0	26800	0,9	0,0	26800	1	26800	1,0	0,0	26800	1	26800	1,1	0,0	26800	1
abi	20	34800	4,1	0,0	34800	14,4	0,0	34800	75	34800	3,1	0,0	34800	2	34800	23,2	0,0	34800	3
abi	30	49656	123,7	0,0	49656	270,2	0,0	49660	2320	49656	130,5	0,0	49660	215	49656	129,7	0,0	49660	20
abi	42	56218	1132,5	0,0	56218	2511,4	0,0	56224	7529	56218	1246,9	0,0	56224	1252	56218	482,9	0,0	56224	40
abi	56	70573	3600,0	1,9	70829	3600,0	2,0	72264	14437	69683	3600,0	3,5	72117	2038	70931	3600,0	1,1	71745	215
atl	12	38200	11,8	0,0	38200	12,2	0,0	38200	2	38200	11,7	0,0	38200	1	38200	11,7	0,0	38200	1
atl	20	54600	3600,0	1,5	54600	3600,0	1,0	55150	8170	54600	3600,0	1,0	55150	867	54600	3600,0	1,1	55200	87
atl	30	69600	3600,0	1,5	69600	3600,0	1,6	70733	7092	69400	3600,0	1,8	70668	431	69600	3600,0	1,5	70668	82
atl	42	84500	3600,0	3,0	84400	3600,1	3,1	87012	3653	84600	3600,1	2,9	87012	368	84000	3600,1	3,6	87012	38
atl	56	76614	3603,1	16,9	82418	3604,5	9,4	90159	569	80986	3600,0	11,2	90092	103	79990	3600,3	12,7	90169	3
gea	12	35600	661,9	0,0	35600	719,8	0,0	35600	69	35600	667,9	0,0	35600	8	35600	890,2	0,0	35600	2
gea	20	48400	3600,0	13,9	48667	3600,0	15,5	56200	317	48200	3600,0	14,1	55000	83	47700	3600,0	10,3	52600	8
gea	30	61900	3603,1	19,9	69800	3600,1	6,3	74200	218	68400	3600,0	8,5	74200	40	68500	3600,0	8,3	74200	3
gea	42	89503	3600,0	10,8	94800	3600,0	4,6	99200	202	95040	3601,3	4,4	99200	9	94767	3600,0	4,7	99200	2
gea	56	108580	3600,1	13,8	120830	3600,0	2,3	123600	141	119256	3600,0	3,6	123600	11	117033	3607,9	5,6	123600	2
media				4,6			2,7					3,0					2,9		
$K \leq 20$				2,1			2,2					2,0					1,5		
$K > 20$				6,3			3,1					3,6					3,8		

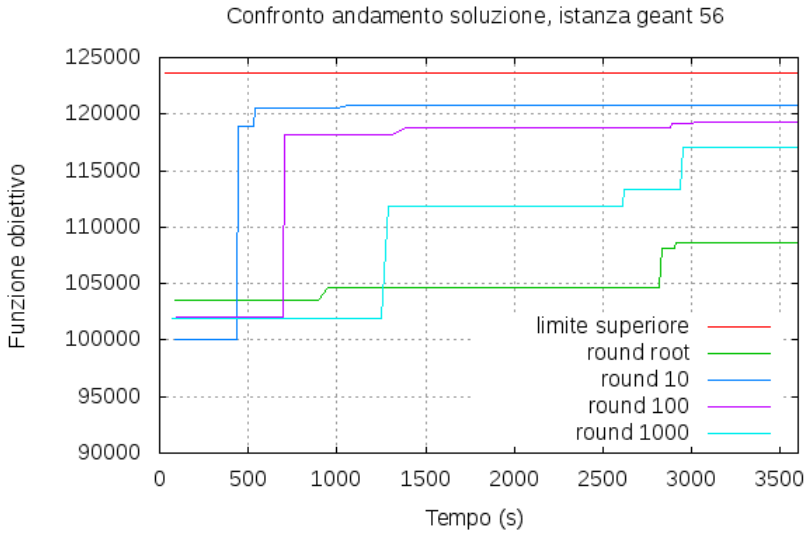

Figura 6.5: Andamento nel tempo della soluzione determinata dai modelli *Round root*, *Round 10*, *Round 100* e *Round 1000* per l'istanza `geant 56`

In Figura 6.5 è riportato il grafico relativo all'andamento del valore della miglior soluzione ammissibile nel tempo, determinato risolvendo le formulazioni *Round root*, *Round 10*, *Round 100* e *Round 1000*, e l'andamento del limite superiore che, in questo caso, rimane costante ed uguale per tutte le formulazioni. Sono stati considerati i dati relativi all'istanza `geant 56` che presenta un'andamento tipico osservato anche in altre istanze di grandi dimensioni.

I valori relativi alla formulazione *Round 10* convergono più rapidamente verso il limite superiore rispetto a quelli delle altre formulazioni. La differenza è maggiore con le formulazioni *Round root* e *Round 1000*, dove neanche a fine esecuzione riusciamo a raggiungere il valore che la formulazione *Round 10* ci restituisce in circa 500s. Considerando che, come riportato in Tabella 6.13, la formulazione *Round 1000* esegue solo 2 volte l'euristica di arrotondamento, beneficiando successivamente dei miglioramenti dovuti al metodo risolutivo di Gurobi, riteniamo che l'euristica studiata sia abbastanza efficiente per istanze di grandi dimensioni su tempi più ristretti rispetto al limite inizialmente posto di 3600s.

6.5.4 Confronto con approccio RPF

In [3, 4], dove si studia il problema MMF-CTE, viene proposto un approccio euristico basato su una formulazione "ristretta", chiamata *Restricted Path Formulation (RPF)*. In pratica viene risolto una formulazione semplificata dove la scelta dei cammini per ogni coppia origine-destinazione può essere fatta esclusivamente all'interno

Tabella 6.14: Confronto tra formulazione *RPF* e formulazione *Round 10*. Le soluzioni relative alla formulazione con euristica di arrotondamento sono nettamente migliori di quelle della formulazione *RPF*.

		RPF		round 10			
rete	$\|K\|$	Ψ	time	Ψ	time	gap% Ψ	time ratio
polska	10	30600,0	0,0	30600,0	13,6	0,0	0,00
polska	21	44914,3	8,3	46057,1	3600,0	-2,5	0,00
polska	28	51107,1	301,8	52563,6	3600,0	-2,8	0,08
polska	36	60440,0	267,0	61432,0	3600,0	-1,6	0,07
polska	45	68108,3	1265,2	69578,9	3600,1	-2,1	0,35
abilene	12	25733,3	0,2	26800,0	0,9	-4,0	0,23
abilene	20	31633,3	0,9	34800,0	14,4	-9,1	0,06
abilene	30	44917,5	0,7	49655,6	270,2	-9,5	0,00
abilene	42	49511,1	18,9	56218,2	2511,4	-11,9	0,01
abilene	56	63600,0	210,4	70829,4	3600,0	-10,2	0,06
atlanta	12	33200,0	0,1	38200,0	12,2	-13,1	0,01
atlanta	20	47400,0	59,6	54600,0	3600,0	-13,2	0,02
atlanta	30	63133,3	540,0	69600,0	3600,0	-9,3	0,15
atlanta	42	78204,8	3593,2	84400,0	3600,1	-7,3	1,00
atlanta	56	76860,0	3593,3	82418,2	3604,5	-6,7	1,00
geant	12	34800,0	0,1	35600,0	719,8	-2,2	0,00
geant	20	45100,0	1,0	48666,7	3600,0	-7,3	0,00
geant	30	65950,0	3593,1	69800,0	3600,1	-5,5	1,00
geant	42	87760,0	3591,5	94800,0	3600,0	-7,4	1,00
geant	56	106651,4	3585,5	120830,0	3600,0	-11,7	1,00
media						-6,9	0,04
$K \leq 20$						-6,4	0,00
$K > 20$						-7,2	0,16

di un insieme di cammini pregenerati aleatoriamente. Infatti, per ogni coppia (s, t) si definisce un insieme di cammini M, con $|M| = \phi^{st}_{max}\omega$, dove ϕ^{st}_{max} corrisponde al valore del taglio minimo (flusso massimo) tra s e t con costo 1 associato ad ogni arco, ed ω ad un valore intero (nelle prove effettuate utilizzeremo un valore di $\omega = 2$).

Confrontiamo questa formulazione con la formulazione che utilizza l'euristica di arrotondamento, considerando in particolare la formulazione *Round 10*. In Tabella 6.14 sono riportati i dati relativi all'esecuzione delle due formulazioni, imponendo come tempo limite 3600s per entrambi. Nelle colonne Ψ e *time* abbiamo, rispettivamente, il valore della miglior soluzione trovata e il tempo di esecuzione, mentre nella colonna *gap%* Ψ il valore del gap % tra la soluzione della formulazione *RPF* e quella della formulazione *Round 10*, e nella colona *time ratio* il rapporto tra i due tempi di esecuzione. Osservando i valori del gap% Ψ, si nota che la formulazione che utilizza l'euristica di arrotondamento presenta valori della funzione obiettivo mediamente migliori di circa il 7%. Il nostro approccio risulta quindi essere sensibilmente più efficiente di quello basato su *RPF* in termini di qualità della soluzione trovata.

D'altro canto è doveroso notare che la formulazione *RPF* riporta tempi di esecuzione notevolmente più contenuti per un numero considerevole di istanze. Questo è dovuto al fatto che il risolutore si trova a risolvere una formulazione ristretta, quindi meno complessa della formulazione *multi-flow*, riuscendo a trovare una soluzione ottima in minor tempo. L'ottimo della formulazione *RPF* non corrisponde però all'ottimo della formulazione *multi-flow*, determinando quindi una qualità peggiore delle soluzioni trovate per il problema MMF-CTE.

Considerati questi aspetti, proponiamo di combinare la formulazione *RPF* con l'euristica di arrotondamento, nel tentativo di migliorare la qualità delle soluzioni trovate. L'idea è quella di fornire alla formulazione *RPF* un insieme di cammini "migliori" di quelli pregenerati aleatoriamente, utilizzando i cammini generati tramite il processo di arrotondamento eseguito al nodo radice nella formulazione multi-flusso. Vengono quindi eseguiti i seguenti passi:

- viene avviato il risolutore con la formulazione *multi-flow*, chiamando l'euristica di arrotondamento al nodo radice.

- l'euristica viene eseguita 100 volte, in modo da generare un insieme di cammini distinti per ogni coppia (s, t). In questo modo cerchiamo di ottenere un numero di cammini pari a $|M| = \phi^{st}_{max}\omega$ per ogni coppia (s, t). Visto che nel processo di arrotondamento i pesi sugli archi sono influenzati dai valori frazionari della variabile x^{st}_{ij} ottenuti dalla soluzione del rilassamento continuo, può capitare che il numero di cammini trovati per ogni coppia (s, t) sia $|M| < \phi^{st}_{max}\omega$.

- Una volta generati i cammini per ogni coppia (s, t), viene risolto la formulazione *RPF* utilizzando il nuovo insieme di cammini.

In Tabella 6.15 sono riportati i risultati ottenuti applicando alla formulazione *RPF* il metodo appena descritto, messi a confronto con quelli relativi alla formulazione *RPF* con pregenerazione aleatoria dei cammini. Osservando i valori del gap % e del rapporto tra i tempi di esecuzione si nota che, combinando la formulazione *RPF* con l'euristica di arrotondamento, si ottengono soluzioni leggermente migliori in termini di valore della funzione obiettivo a fronte di tempi di esecuzione nettamente inferiori. Questo accade poiché, definendo per ogni coppia (s, t) un numero di cammini generalmente inferiore a $|M| < \phi^{st}_{max}\omega$, si riduce la complessità della formulazione *RPF*, mantenendo però una buona qualità della soluzione grazie ad una scelta più assennata dei cammini rispetto ad una pregenerazione aleatoria.

Tabella 6.15: Confronto tra formulazione *RPF* con cammini generati aleatoriamente e formulazione *RPF* con cammini generati da arrotondamento.

		RPF		RPF + round			
rete	$\|K\|$	Ψ	time	Ψ	time	gap% Ψ	time ratio
polska	10	30600,0	0,0	30600,0	1,3	0,0%	0,02
polska	21	44914,3	8,3	45200,0	3,2	-0,6%	2,62
polska	28	51107,1	301,8	50977,8	10,6	0,3%	28,40
polska	36	60440,0	267,0	60580,0	8,0	-0,2%	33,26
polska	45	68108,3	1265,2	67417,1	41,9	1,0%	30,16
abilene	12	25733,3	0,2	26800,0	0,7	-4,0%	0,32
abilene	20	31633,3	0,9	31955,6	1,2	-1,0%	0,74
abilene	30	44917,5	0,7	48100,0	3,2	-6,6%	0,23
abilene	42	49511,1	18,9	49566,7	5,0	-0,1%	3,81
abilene	56	63600,0	210,4	63366,3	11,0	0,4%	19,06
atlanta	12	33200,0	0,1	38200,0	1,9	-13,1%	0,04
atlanta	20	47400,0	59,6	52600,0	4,3	-9,9%	13,95
atlanta	30	63133,3	540,0	68800,0	8,5	-8,2%	63,59
atlanta	42	78204,8	3593,2	83590,0	94,7	-6,4%	37,94
atlanta	56	76860,0	3593,3	78327,8	378,0	-1,9%	9,51
geant	12	34800,0	0,1	32200,0	9,4	8,1%	0,01
geant	20	45100,0	1,0	42800,0	29,0	5,4%	0,04
geant	30	65950,0	3593,1	64600,0	60,5	2,1%	59,41
geant	42	87760,0	3591,5	92553,3	73,2	-5,2%	49,04
geant	56	106651,4	3585,5	117433,3	226,0	-9,2%	15,87
media						-2,5%	2,82
$K \leq 20$						-1,9%	0,19
$K > 20$						-2,8%	16,79

6.5.5 Considerazioni finali

In questa sezione volevamo verificare se un approccio euristico potesse giocare un ruolo significativo nel migliorare la qualità delle soluzioni trovate per la formulazione *multi-flow*.

I risultati computazionali hanno confermato le nostre aspettative, dimostrando quanto un approccio euristico studiato appositamente per il problema MMF-CTE sia fondamentale per risolvere le istanze più grosse. Siamo stati così in grado di fornire una soluzione ammissibile a tutte le istanze, riportando un soddisfacente gap medio minore del 3% grazie all'impiego di un'euristica di arrotondamento di supporto all'algoritmo di Branch-and-Bound nella risoluzione della formulazione multi-flusso. Gli algoritmi euristici implementati riescono a condurre il risolutore verso una buona porzione dell'albero di branching, consentendo di trovare buone soluzioni ammissibili per ogni istanza, differentemente da quanto accade utilizzando esclusivamente l'euristica del risolutore.

Inoltre abbiamo confrontato il metodo di arrotondamento con un altro approc-

cio proposto in letteratura per il problema MMF-CTE, quello basato sulla formulazione ristretta RPF. L'euristica di arrotondamento applicata al Branch-and-Bound nella risoluzione del modello multi-flusso ha portato a delle soluzioni mediamente migliori del 7% rispetto a quelle determinate dalla risoluzione della formulazione RPF. È stata così proposto di combinare l'euristica di arrotondamento alla formulazione RPF, ottenendo miglioramenti sensibili sia in termini di miglior soluzione che di tempi di esecuzione.

Capitolo 7

Conclusioni e sviluppi futuri

In questa tesi è stato studiato il problema di instradamento in reti di telecomunicazioni in cui i flussi vengono allocati secondo un criterio di Max-Min Fairness, imposto dai protocolli di rete. L'obiettivo è quello di massimizzare la sommatoria pesata dei flussi di tutte le coppie origine-destinazione, sottostando ai vincoli MMF. Dato che il problema risulta difficile, per migliorare la qualità delle soluzioni e ridurre i tempi di esecuzione, ci siamo concentrati sullo studio di vincoli alternativi e disuguaglianze valide da aggiungere alla formulazione estesa di PLMI, e metodi euristici da applicare ad ogni nodo del procedimento di branch-and-bound.

Si è analizzato in dettaglio il problema dei sotto-cicli presentando tre diverse formulazioni estese per la loro eliminazione. Inoltre è stato proposto un approccio alternativo basato sull'inserimento iterativo di piani di taglio nel procedimento di Branch-and-Cut. Abbiamo poi introdotto nuovi vincoli validi per il nostro problema, definendo così due formulazioni alternative per i vincoli di bilanciamento del flusso e per quelli di MMF, e nuove diseguaglianze valide. Infine sono stati proposti due algoritmi euristici studiati per migliorare la qualità soluzioni. Sfruttando l'algoritmo waterfilling, si è sviluppato un metodo per la ricerca di cammini poco sovrapposti ed un metodo basato su tecniche di arrotondamento delle soluzioni ottenute dai rilassamenti continui, entrambi volti a supportare l'algoritmo di Branch-and-Bound.

I risultati computazionali presentati suggeriscono che la formulazione estesa *multi-flow* risulta essere la più efficiente per la rimozione dei sotto-cicli nel nostro problema, contrariamente a quanto noto per il TSP, dove risulta più efficiente l'approccio basato sulla generazione iterativa di piani di taglio. Effettuando invece un'analisi sui vincoli alternativi inseriti, si nota che l'elevata complessità della formulazione *multi-flow* annulla l'effetto di rafforzamento ottenuto a livello del rilassamento continuo, dimostrandosi meno efficace. L'impiego dei metodi euristici, invece, ci ha permesso di trovare una soluzione ammissibile per le istanze di dimen-

sioni maggiori. In particolare, iterando l'euristica di arrotondamento in più nodi dell'albero di branching, si ottengono i risultati più interessanti in termini di gap tra la miglior soluzione e il miglior limite superiore trovati.

Come sviluppi futuri proponiamo un'euristica di supporto per l'algoritmo di Branch-and-Bound più articolata, da applicare alla formulazione ristretta RPF, che sia in grado di determinare un insieme di cammini che aiutino ad esplorare buone porzioni dell'albero di branching. In alternativa si potrebbero sviluppare metodi meta-euristici, come proposto in [16] per problemi di condivisione delle risorse in una logica di *proportional fairness*, definendo così un'euristica in grado di determinare buone soluzioni senza l'utilizzo di risolutori di PLMI. Infine si potrebbero considerare altre funzioni obiettivo che tengano conto del risparmio energetico, problema attuale e di forte interesse in diversi ambiti.

Appendice A

Codice

Riportiamo si seguito alcuni esempi di formulazioni e procedure implementati con il linguaggio `python` interfacciato con il risolutor di PLMI *Gurobi*.

A.1 Formulazione estesa multi-flusso

Riportiamo il codice relativo alla definizione della formulazione *multi-flow*.

```
m = Model('multi-flusso')

# VARIABLE DEFINITION
for s,t in commodities:

        phi[s, t] = m.addVar( name = 'phi_%s_%s' % (s, t))

        for i,j in arcs:

            f[s, t, i, j] = m.addVar(ub = capacity[i, j], vtype = GRB.
    CONTINUOUS, name = 'f_%s_%s_%s_%s' % (s, t, i, j))
            x[s, t, i, j] = m.addVar(vtype = GRB.BINARY, name = 'x_%s_%
    s_%s_%s' % (s, t, i, j))
            y[s, t, i, j] = m.addVar(vtype = GRB.BINARY, name = 'y_%s_%
    s_%s_%s' % (s, t, i, j))

            for n in nodes:
                u[s, t, i, j, n] = m.addVar(vtype = GRB.CONTINUOUS,
    name = 'u_%s_%s_%s_%s_%s' % (s, t, i, j, n))


        for n in nodes:
            z[s, t, n] = m.addVar(vtype = GRB.CONTINUOUS, name = 'z_%s_
    %s_%s' % (s, t, n))
```

```
21      for i, j in arcs:
            uu[i, j] = m.addVar(vtype = GRB.CONTINUOUS, lb = 0, name = 'uu_
        %s_%s' % (i, j))
23
    m.update()
25
    # OBJECTIVE FUNCTION
27  m.setObjective(quicksum(phi[s, t] for s,t in commodities), GRB.MAXIMIZE
        )
29  ## CONSTRAINTS
31  # arc capacity constraints
    for i,j in arcs:
33      m.addConstr(quicksum(f[s, t, i, j] for (s,t) in commodities) <=
        capacity[i, j])
35  # flow balance constraints
    for n in nodes:
37      for s,t in commodities:
            if n == s:
39              m.addConstr(quicksum(f[s, t, i, j] for (i,j) in arcs if i
        == n ) -
                            quicksum(f[s, t, i, j] for (i,j) in arcs if j
        == n ) == phi[s, t])
41
            if n == t:
43              m.addConstr(quicksum(f[s, t, i, j] for (i,j) in arcs if i
        == n ) -
                            quicksum(f[s, t, i, j] for (i,j) in arcs if j
        == n ) == - phi[s, t])
45
            if n != s and n != t:
47              m.addConstr(quicksum(f[s, t, i, j] for (i,j) in arcs if i
        == n) -
                            quicksum(f[s, t, i, j] for (i,j) in arcs if j
        == n) == 0)
49
    # activation constraints
51  for i,j in arcs:
        for s,t in commodities:
53          m.addConstr(f[s, t, i, j] <= capacity[i, j] * x[s, t, i, j])
55
    # MMF classic constraints
```

```
57
for s,t in commodities:
59      m.addConstr(quicksum(y[s, t, i, j] for i, j in arcs) >= 1) # set
        covering

61      for i,j in arcs:
            m.addConstr(quicksum(f[o, d, i, j] for o, d in commodities) >=
63                      capacity[i, j] * y[s, t, i, j])

65          m.addConstr(uu[i,j] >= f[s, t, i, j])

67          m.addConstr(f[s, t, i, j] >= uu[i, j] - capacity[i, j] * (1 - y
        [s, t, i, j]))


69
# each communication routed to a single path
71 for n in nodes:
        for s, t in commodities:
73          m.addConstr(quicksum(x[s, t, i, j] for i, j in arcs if i == n)
        <= 1)

75 # constraints for LP relaxations and Branch-and-Bound convergence
        acceleration
   cap_min = min(capacity.itervalues())
77 cardinality_dem = len(commodities)

79 for i, j in arcs:
        for s, t in commodities:
81          m.addConstr(phi[s, t] >= cap_min / cardinality_dem)

83          m.addConstr(y[s, t, i, j] <= x[s, t, i, j])


85
# TSP constraints multiflow type
87
for s, t in commodities:
89      for n in nodes:
            if n != s:
91
                # ham_2
93              m.addConstr(quicksum(u[s, t, i, j, n] for i, j in arcs if i
        == s) == z[s, t, n])
                # ham_3
95              m.addConstr(quicksum(u[s, t, i, j, n] for i, j in arcs if j
        == s) == 0)
                # ham_4
```

```
            m.addConstr(quicksum(u[s, t, i, j, n] for i, j in arcs if j
    == n) == z[s, t, n])
            # ham_5
            m.addConstr(quicksum(u[s, t, i, j, n] for i, j in arcs if i
    == n) == 0)
            # ham_7
            m.addConstr(quicksum(u[s, t, i, j, n] for i, j in arcs if i
    == t) == 0)
            # ham_8
            m.addConstr(quicksum(x[s, t, i, j] for i, j in arcs if j ==
    n) == z[s,t, n])
            # ham_6
            for h in nodes:
                if h != s and h != n:
                    m.addConstr(quicksum(u[s, t, i, j, n] for i, j in
    arcs if j == h) == quicksum(u[s, t, i, j, n] for i, j in arcs if i
    == h))
            for i, j in arcs:
                # ham_1
                m.addConstr(u[s, t, i, j, n] <= x[s, t, i, j])

m.update()
```

code/multiflow.py

A.2 Algoritmo Waterfilling

Presentiamo in questa sezione il codice relativo all'algoritmo waterfilling per l'allocamento dei flussi MMF dati i cammini.

```
def waterfill(paths, arcs, cap, commodities):
    # init output flows
    f = {(s, t, i, j): 0 for s, t in commodities for i, j in arcs }


    x = {(s, t, i, j): 0 for s, t in commodities for i, j in arcs }

    comm = copy.copy(commodities)
    capacity = copy.copy(cap)

    saturated = []

    # count commodities for each arc
    for s,t in commodities:
        path = paths[s, t]
```

```
        for i, j in path:
            x[s, t, i, j] = 1

    crow = {}
    for i, j in arcs:
        crow[i, j] = sum(x[s, t, i, j] for s, t in commodities)

    # main cycle
    while len(comm) > 0:

        # find the arch with the smallest delta available
        most_crowded = []

        for i, j in crow.keys():

            if crow[i,j] == 0:
                continue

            delta = float(capacity[i, j]) / int(crow[i, j])

            if len(most_crowded) == 0:
                most_crowded.append(i)
                most_crowded.append(j)
                most_crowded.append(delta)
                continue

            if delta < most_crowded[2]:
                most_crowded[0] = i
                most_crowded[1] = j
                most_crowded[2] = delta

        if len(most_crowded) == 0:
            break


        delta = most_crowded[2]

        for s, t in comm:
            if (s, t) in saturated:
                continue
            count +=1
            if (most_crowded[0], most_crowded[1]) in paths[s, t]:
                for i, j in paths[s, t]:
                    f[s, t, i, j] = delta
                    capacity[i, j] = capacity[i, j] - delta
```

```
62                # remove the commodity
                  comm.pop(comm.index((s,t)))
64                saturated.append((s,t))

66        # remove the arc used as most_crowded
          del crow[most_crowded[0], most_crowded[1]]
68
          # update the crow count (remove the commodities with a
    saturated arc)
70        for i, j in arcs:
              crow[i, j] = sum(x[s, t, i, j] for s, t in comm  )
72
    return f
```

code/waterfill.py

A.3 Generazione dei piani di taglio per l'eliminazione dei sotto-cicli

Presentiamo in questa sezione il codice relativo alla procedura per la generazione di piani di taglio per la rimozione dei sotto-cicli.

```
1  def subtourelim(model, where):

3      commodities = model._commodities
       arcs = model._arcs
5      nodes = model._nodes
       capacity = model._capacity
7      x = model._x

9      # se viene trovata una soluzione intera
       if where == GRB.callback.MIPSOL:
11       for s, t in commodities:
               selected = [[] for i in range(len(nodes))]
13
           # recupero gli archi selezionati
15         for i, j in arcs:
               sol = model.cbGetSolution(model._x[s, t, i, j])
17             if sol > 0.1:
                   selected[i].append(j)
19
           # cerco cicli nella cammino tra s e t
21         cicli = graph.findcycle(selected, 0)

23         # costruisco il taglio e lo aggiungo alla formulazione
```

```
            for ciclo in cicli:
25              expr = quicksum(model._x[s, t, i, j] for i, j in ciclo)
                model.cbLazy(expr <= len(ciclo)-1)
```

code/subtour.py

A.4 Procedura di arrotondamento

Presentiamo in questa sezione il codice relativo alla funzione di arrotondamento.

```
def rounding(model, round_type):
  commodities = model._commodities
  arcs = model._arcs
  nodes =  model._nodes
  current = model.cbGet(GRB.callback.MIPNODE_NODCNT)
  x = {}
  paths = {}

  for s, t in commodities:
    for i, j in arcs:
      sol = model.cbGetNodeRel(model._x[s, t, i, j])

      x[s, t, i, j] = random.uniform(0, 1 - sol)

  for s, t in commodities:
    G = {}
    for n in nodes:
      G[n] = {}
      for i,j in arcs:
        if i == n:
          G[n][j] = x[s,t,i,j]d

    p = graph.path(graph.shortestPath(G, s, t, model._rounding_random))

    paths[s,t] = p

  return paths
```

code/round.py

Bibliografia

[1] J. Kleinberg, Y. Rabani, and E. Tardos. Fairness in routing and load balancing. In *Foundations of Computer Science, 1999. 40th Annual Symposium on*, pages 568–578, 1999.

[2] D. Nace and M. Pioro. Max-min fairness and its applications to routing and load-balancing in communication networks: a tutorial. *Communications Surveys Tutorials, IEEE*, 10(4):5–17, 2008.

[3] E. Amaldi, A. Capone, S. Coniglio, and L.G. Gianoli. Network optimization problems subject to max-min fair flow allocation. *Communications Letters, IEEE*, 17(7):1463–1466, 2013.

[4] E. Amaldi, S. Coniglio, L. G. Gianoli, and C. U. Ileri. On single-path network routing subject to max-min fair flow allocation. *Electronic Notes in Discrete Mathematics*, 41(0):543 – 550, 2013.

[5] D. Bertsekas and R. Gallager. *Data networks (2nd ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1992.

[6] M. Pioro and D. Medhi. *Routing, Flow, and Capacity Design in Communication and Computer Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[7] P. Nilsson. *Fairness in Comunication and Computer Network Design*. Tesi PhD, Lund University, Svezia, 2006.

[8] W. Ogryczak, M. Pioro, and A. Tomaszewski. Telecommunications network design and max-min optimization problem. *Journal of Telecommunications and Information Technology*, 3:1–14, 2005.

[9] A. Tomaszewski. A polynomial algorithm for solving a general max-min fairness problem. *European Transactions on Telecommunications*, 16(3):233–240, 2005.

[10] M. Pioro. Fair routing and related optimization problems. In *Advanced Computing and Communications, 2007. ADCOM 2007. International Conference on*, pages 229–235, 2007.

[11] E. Danna, A. Hassidim, H. Kaplan, A. Kumar, Y. Mansour, D. Raz, and M. Segalov. Upward max min fairness. In *INFOCOM, 2012 Proceedings IEEE*, pages 837–845, 2012.

[12] E. Danna, S. Mandal, and A. Singh. A practical algorithm for balancing the max-min fairness and throughput objectives in traffic engineering. In *INFOCOM, 2012 Proceedings IEEE*, pages 846–854, 2012.

[13] A. Coluccia, A. D'Alconzo, and F. Ricciato. On the optimality of max-min fairness in resource allocation. *Annales des Télécommunications*, 67(1-2):15–26, 2012.

[14] A.J. Ormans and H.P. Williams. A survey of different integer programming formulations of the travelling salesman problem. *Optimisation, Econometric and Financial Analysis (Advances in Computational Management Science)*, pages 93–108, 2006.

[15] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski. Sndlib 1.0—survivable network design library. *Networks*, 55(3):276–286, 2010.

[16] Mario Köppen, Kaori Yoshida, Kei Ohnishi, and Masato Tsuru. Meta-heuristic approach to proportional fairness. *Evolutionary Intelligence*, 5(4):231–244, 2012.