UNIVERSITÀ DEGLI STUDI DI MILANO

DIPARTIMENTO DI INFORMATICA

Corso di Laurea Magistrale in
Scienze e Tecnologie dell'Informazione

Algoritmi di programmazione matematica per lo spatial cloaking

RELATORE
Prof. Giovanni Righini

CORRELATORE
Prof. Alberto Ceselli

TESI DI LAUREA DI

Diego Valorsi

Matr. 790944

Anno Accademico 2012/2013

Alla mia famiglia

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

La ploriferazione di applicazioni di *geosocial networking* dei giorni nostri, come *Google Latitude* e *Geo-Twitter* impongono importanti sfide per la protezione della privacy a riguardo della posizione geografica degli utenti. Sviluppato dalla nostra università, il *framework PROBE (Privacy-preserving Obfuscation Environment)* [1,2,3] definisce un modello di protezione delle posizioni sensibili. Un utente considera una posizione geografica come sensibile quando non vuol far sapere di essere o essere stato in tale posizione. Utilizzando *PROBE* un utente può definire delle posizioni come sensibili; di conseguenza quando l'utente si trova realmente in queste posizioni, il suo dispositivo mobile non trasmetterà una posizione geografica precisa, ma piuttosto trasmetterà informazioni geografiche approssimative con lo scopo di nascondere la reale posizione dell'utente.

1.1 Il problema

La modellazione del problema proposta da *PROBE* trasforma le posizioni geografiche in un digrafo a griglia $G = (\mathcal{N}, \mathcal{A})$ con 4-connettività, dove per ogni nodo (o cella) $i \in \mathcal{N}$ è associata la probabilità ρ_i che una generica persona ha di trovarsi nella cella i. La somma delle probabilità ρ_i in $\mathcal{N}$ è 1.

Un utente può decidere di definire delle griglie non sovrapposte contenute in G come posizioni sensibili, a queste griglie farò in seguito riferimento come regioni base. Ogni regione base è composta da un insieme di nodi (o celle) a cui in seguito farò riferimento come nodi (o celle) base, ne consegue che una cella base corrisponde sempre ad una posizione sensibile (o cella sensibile). Ad ogni cella è associato quindi un dato binario s_i, se $s_i = 1$, i è una cella

base, altrimenti i non è una cella base, e in questo caso viene detta cella esterna.

Una possibile rappresentazione grafica di un'istanza del problema è mostrata nella figura 1.1.

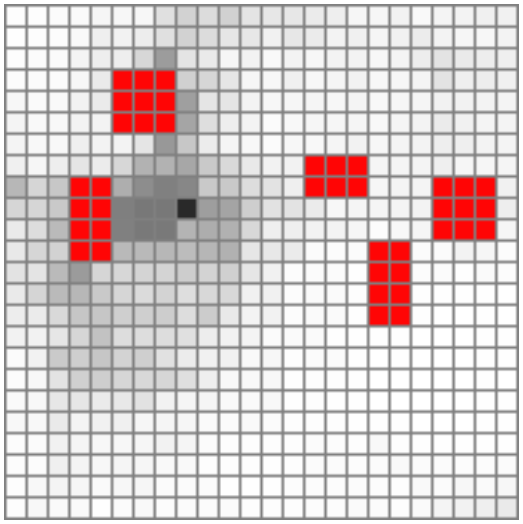

Figura 1.1: Rappresentazione di un'istanza del problema

Il colore rosso indica le posizioni sensibili, i livelli di grigio nelle celle esterne corrispondono al valore di probabilità ρ_i associato alla cella.

Viene definita sensitività σ di un insieme di celle $\mathcal{R}$, la seguente probabilità condizionata:

$$\sigma(\mathcal{R}) = \begin{cases} \dfrac{\sum_{i \in R} s_i \rho_i}{\sum_{i \in R} \rho_i} & \text{se } \sum_{i \in R} \rho_i \neq 0 \\ 0 & \text{altrimenti} \end{cases}$$

La sensitività $\sigma(\mathcal{R})$ di un insieme di celle $\mathcal{R}$ rappresenta il livello di offuscamento con il quale sono "nascoste" le celle sensibili presenti in $\mathcal{R}$. Più $\sigma(\mathcal{R})$ è piccolo, più alto è il livello di offuscamento delle celle sensibili in $\mathcal{R}$ e di conseguenza la posizione sensibile risulta maggiormente "nascosta". Una regione che non contiene celle sensibili ha sensitività pari a 0.

L'idea è quella di utilizzare un parametro di soglia sulla sensitività τ, e partendo da una regione sensibile (celle base) definire una regione connessa di dimensioni minime (minimo numero di celle) che connetta un insieme di celle $\mathcal{R}$ tale che $\sigma(\mathcal{R}) \leq \tau$. L'insieme di celle $\mathcal{R}$ costituisce la nuova posizione offuscata da trasmettere.

Il problema può quindi essere modellato associando una variabile binaria x_i ad ogni nodo $i \in \mathcal{N}$. Per semplificare la notazione, il vincolo su $\sigma(\mathcal{R})$ si può riformulare come segue:

$$\frac{\sum_{i \in R} s_i \rho_i}{\sum_{i \in R} \rho_i} \leq \tau$$

quindi:

$$\sum_{i \in R} s_i \rho_i \leq \tau \sum_{i \in R} \rho_i$$

perché il termine $\sum_{i \in R} \rho_i$ è strettamente positivo per qualsiasi regione $\mathcal{R}$ che include almeno una cella base, possiamo riformulare il vincolo come:

$$\sum_{i \in R} \rho_i (s_i - \tau) \leq 0$$

e quindi:

$$\sum_{i \in R} \rho_i (s_i - \tau) x_i \leq 0$$

per semplificare la notazione introduco i pesi $w_i = \rho_i(\tau - s_i)$ e riscrivo il vincolo come:

$$\sum_{i \in R} w_i x_i \geq 0 \qquad (1.1)$$

Nel modello definito da *PROBE*, ogni regione connessa che costituisce la una nuova posizione offuscata da trasmettere può essere formata da una o più arborescenze che soddisfano il vincolo di sensitività (1.1). In vincolo di sensitività (1.1) è anche chiamato vincolo di soglia.

Le possibili radici delle arborescenze sono le celle base, le arborescenze non possono condividere nessuna cella.

Sia $F = (\mathcal{P}, \mathcal{H})$ una foresta dove $\mathcal{P}$ rappresenta l'insieme dei nodi e $\mathcal{H}$ l'insieme delle arborescenze, *PROBE* definisce come miglior soluzione la foresta F a cui corrisponde il minimo rapporto tra il numero di nodi presenti nella foresta F e il numero di arborescenze che forma la foresta F. Ognuna delle arborescenze contenuta in F deve soddisfare il vincolo di sensitività (1.1).

$$min \ \frac{|\mathcal{P}|}{|\mathcal{H}|} \qquad (1.2)$$

La funzione obiettivo definita da *PROBE* (1.2) è quindi non lineare, un possibile modo che consente di trovare la soluzione ottima senza risolvere problemi non lineari, è quello di calcolare la foresta F di k arborescenze che minimizza il numero di nodi, e di costruire il fronte di pareto al variare del numero k di arborescenze. Il numero di arborescenze k che formano la foresta F sarà quindi un vincolo aggiuntivo da trattare.

Le figure 1.2 e 1.3 rappresentano due possibili soluzioni. La soluzione in figura 1.2 utilizza 11 arborescenze e 123 nodi, ha quindi un valore di funzione obiettivo pari a $\frac{123}{11} \approx 11.18$. La soluzione in figura 1.3 utilizza invece 9 arborescenze e 114 nodi, ed ha quindi un valore di funzione obiettivo pari a $\frac{114}{9} \approx 12.66$. Nel modello definito da *PROBE*, la soluzione in figura 1.2 è pertanto migliore della soluzione in figura 1.3.

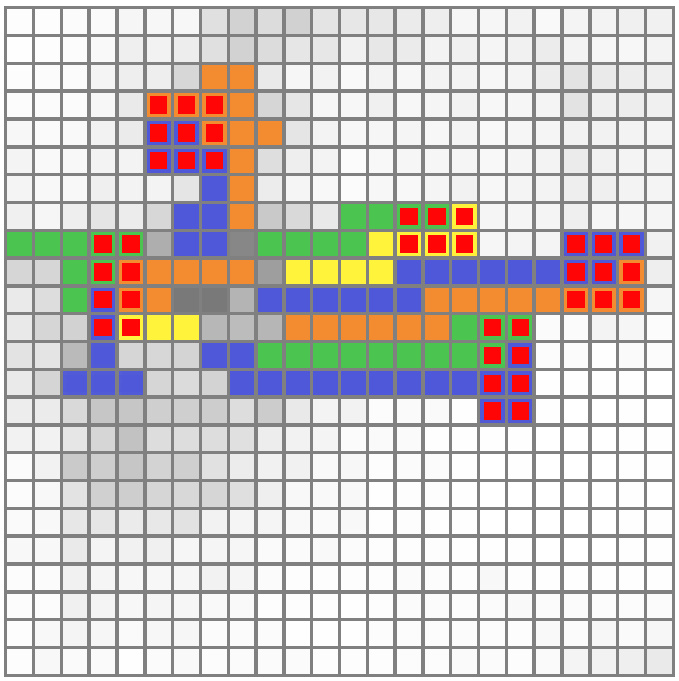

Figura 1.2: Una possibile soluzione composta da 11 arborescenze

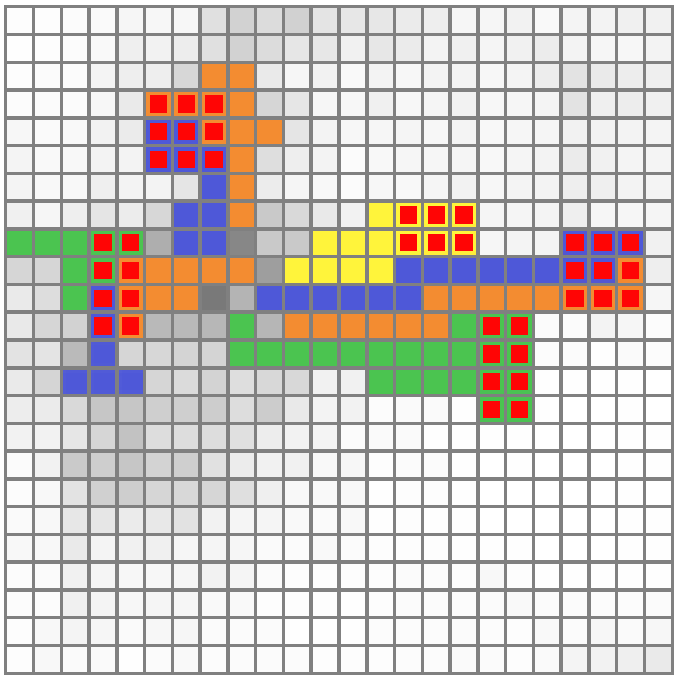

Figura 1.3: Una possibile soluzione composta da 9 arborescenze

1.2 Soluzione proposta e letteratura

Per risolvere il problema appena descritto, sarà necessario calcolare una foresta composta dal minimo numero di nodi per ogni possibile valore intero del parametro k, k definisce il numero di arborescenze di cui è composta la foresta.

Un problema simile, che ha ricevuto molta attenzione nell'ambito dell'ottimizzazione combinatoria, nel *networking* e nei sistemi distribuiti è il *Generalized Steiner Problem (GSP)*, formulato da Krarup[4].

Dato un grafo con pesi non negativi e un insieme di coppie di vertici, l'obiettivo è costruire un sottografo a costo minimo, in modo che ogni coppia dell'insieme di vertici in input sia connessa da un cammino.

Per il *GSP* è stato mostrato una algoritmo polinomiale con una garanzia di approssimazione di $2(1-\frac{1}{n})$ [5,6] ed altri algoritmi di approssimazione per il caso in cui non sono ammessi duplicati nei collegamenti [7,8,9].

Per determinare la foresta di k arborescenze con minimo numero di nodi propongo l'utilizzo di un algoritmo branch & price.

L'algoritmo branch & price, che verrà poi descritto nel capitolo 3, necessità di una procedura che generi arborescenze a costo minimo (dove il

costo equivale al numero di nodi che formano l'arborescenza) che rispettano il vincolo di sensitività (1.1) a partire da un certo nodo radice $r \in \mathcal{N}$.

La figura 1.4 mostra una possibile arborescenza a costo minimo che soddisfa il vincolo di sensitività (o vincolo di soglia). In questo caso, la radice r dell'arborescenza, è il nodo base con il nucleo di colore verde.

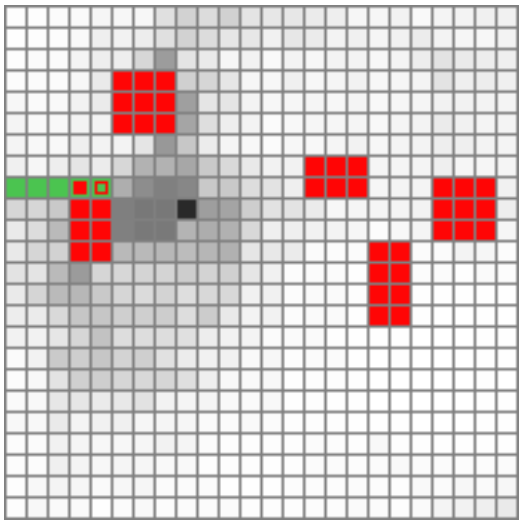

Figura 1.4: Una possibile arborescenza a costo minimo a partire da una certa radice r

Per il calcolo di tale arborescenza propongo l'utilizzo di un algoritmo branch & cut che verrà descritto nel capitolo 2.

Le varianti del problema di trovare l'arborescenza a costo minimo sono molto studiate in letteratura perché hanno importanti applicazioni in diversi ambiti: ad esempio trovano impiego nella progettazione di reti di comunicazione, oppure nella progettazione di circuiti elettrici. In tutti i casi è richiesta una copertura a costo minimo e dei vincoli da rispettare.

Problemi in cui si cercano alberi di copertura a costo minimo aventi una struttura molto simile sono:

- *Steiner Tree/Arborescence Problem (SP)*, caso particolare del sopracitato *GSP*, nel quale esiste un insieme di nodi terminali, ed ogni coppia di nodi terminali deve essere connessa. Lo *SP* è stato tra le prime dimostrazioni di problema *NP-completo* da parte di Karp. Per le innumerevoli varianti dello *SP* sono stati presentati algoritmi esatti basati su branch & cut [9,10,27], euristiche [11,12] ed algoritmi di approssimazione [13,14].

- *Generalized Minimum Spanning Tree/Arborescence Problem (GMSTP)*, in questo problema un sottoinsieme dei nodi del grafo costituiscono dei *cluster* disgiunti, ed è necessario connettere con una struttura ad albero esattamente un nodo per ogni *cluster* partendo da una certa radice. Ogni arco ha associato un costo, l'obiettivo consiste nel minimizzare il costo associato agli archi. In [16] è proposto un algoritmo branch & bound per la soluzione esatta di *GMSTP*, in [17], per la variante con premi associati ai nodi, è proposto un algoritmo branch & cut e degli algoritmi euristici che sfruttano tecniche come la ricerca locale e gli algoritmi genetici.

Al problema di trovare l'arborescenza di costo minimo che soddisfa il vincolo di soglia (1.1), farò in seguito riferimento con *MATC* (*Minimum Arborescence with Threshold Constraint*). Al problema di trovare la foresta di k arborescenze (ognuna soddisfa il vincolo di soglia (1.1)) con il minimo numero di nodi farò invece riferimento con *MKFTC* (*Minimum K Forest with Threshold Constraints*).

1.3 Strumenti utilizzati

Entrambi gli algoritmi presentati sono stati sviluppati secondo il paradigma della programmazione orientata agli oggetti. Entrambi utilizzano come solutore CPLEX 12.6 [18].

L'algoritmo di branch & price utilizza come framwork SCIP 3.01 [19], SCIP utilizza CPLEX 12.6 come *LP solver*. Il codice è stato scritto con il linguaggio C++ e compilato con Apple LLMV 5.1 con impostazioni di default.

I test sono stati condotti su MacBook Pro Late 2013 con processore quadcore a 2.6 GHz dotato di 16 GB di RAM.

Capitolo 2

Algoritmo branch and cut

In questo capitolo mostro i dettagli dell'algoritmo branch & cut che permette di risolvere il problema *MATC*.

2.1 Basi della metodologia branch and cut

Il branch & cut [20] è un metodo di ottimizzazione combinatoria utilizzato per risolvere i problemi di programmazione lineare intera, nasce dalla combinazione di due tecniche: branch & bound e piani di taglio.

La tecnica branch & bound ottiene una soluzione ottima di un problema di programmazione lineare intera eseguendo un'enumerazione implicita di tutte le possibili soluzioni, l'enumerazione è ottenuta tramite algoritmi di *branching*, ovvero esplorando un albero in cui ogni nodo rappresenta un sottoproblema più "facile" da risolvere.

La radice dell'albero rappresenta il problema iniziale in cui tutte le variabili sono libere all'interno del loro dominio. L'insieme delle soluzioni corrispondenti al problema radice S_0 viene partizionato in K sottoinsiemi (disgiunti per ragioni di efficienza) che rappresentano le soluzioni di K sottoproblemi $S_1 \dots S_K$, tali che: $\bigcup_{i=1}^{K} S_i = S_0$.

In modo ricorsivo, poi, ogni sottoproblema S_i è radice di un nuovo albero. Per ogni ramo dell'albero, la generazione dei sottoproblemi figli è arrestata se tutte le variabili sono fissate (nodo foglia, rappresenta una possibile soluzione) oppure, se la soluzione ammissibile duale (bound duale) calcolata per ogni sottoproblema ha un valore peggiore o coincidente della miglior soluzione ammissibile trovata (bound primale).

La metodologia dei piani di taglio si basa sul rafforzamento progressivo della formulazione, fino alla determinazione di una soluzione ottima che ri-

spetti i vincoli di interezza. Il piano di taglio generato ad ogni iterazione deve rispettare due condizioni: escludere il punto di ottimo continuo corrente x^* e non escludere nessuno dei punti interi interni alla regione ammissibile x.

Combinando tali tecniche si eliminano i loro principali difetti, in particolare:

- si garantisce un rafforzamento dinamico del problema rispetto al branch and bound
- si garantisce l'eliminazione del fenomeno del *tailing off* tipico dei piani di taglio grazie alla possibilità di eseguire branching

2.2 Modello matematico per MATC

Dati:

1. Siano $\mathcal{N} = \{1 \dots N\}$ ed $\mathcal{A} = \{1 \dots A\}$ rispettivamente l'insieme dei nodi e degli archi di un digrafo $G = (\mathcal{N}, \mathcal{A})$ privo di autoanelli
2. Sia $r \in \mathcal{N}$ il nodo radice nel quale deve essere radicata l'arborescenza
3. Ad ogni nodo $i \in \mathcal{N}$ è associato un peso $w_i \in \mathbb{R}$

Variabili:

1. Per ogni nodo $i \in \mathcal{N}$ è presente una variabile binaria x_i che indica la presenza in soluzione del nodo i
2. Per ogni arco $j \in \mathcal{A}$ è presente una variabile binaria y_j che indica la presenza in soluzione dell'arco j

Vincoli:

1. Per ogni nodo $i \in \mathcal{N} \setminus r$ in soluzione ($x_i = 1$), deve essere presente in soluzione esattamente un arco $j \in \mathcal{A}$ ($y_j = 1$) tra l'insieme degli archi entranti nel nodo i
2. L'insieme dei nodi in soluzione ($x_i = 1$), e l'insieme degli archi in soluzione ($y_j = 1$) devono formare un digrafo connesso
3. La radice $r \in \mathcal{N}$ è in soluzione
4. La somma dei pesi w_i associati ai nodi in soluzione ($x_i = 1$) deve valere almeno 0 (vincolo di soglia)

Obiettivo:

1. Minimizzare il numero di archi che formano l'arborescenza: $\sum_{j=1}^{A} y_j$

Modello:

$$
\begin{array}{lll}
min & \sum_{j=1}^{A} y_j & & (2.1) \\
s.t. & & & \\
& \sum_{j \in \delta^-(i)} y_j = x_i & \forall i \in \mathcal{N} \setminus r & (2.2) \\
& \sum_{j \in \delta^-(S)} y_j \geq x_i & \forall S \subseteq \mathcal{N} \setminus r, \forall i \in S & (2.3) \\
& x_r = 1 & & (2.4) \\
& \sum_{i=1}^{N} w_i x_i \geq 0 & & (2.5) \\
& x_i \in \{0,1\} & \forall i \in \mathcal{N} & (2.6) \\
& y_j \in \{0,1\} & \forall j \in \mathcal{A} & (2.7)
\end{array}
$$

L'obiettivo (2.1) è la minimizzazione del numero di archi che formano l'arborescenza, poiché in ogni nodo di una arborescenza è presente per definizione esattamente in un solo arco entrante, sostituendo la presente funzione obiettivo con la minimizzazione del numero totale di nodi che formano l'arborescenza si ottiene un modello equivalente.

I vincoli (2.2) indicano che un nodo fa parte della soluzione solo se raggiunto da esattamente un arco entrante.

I vincoli (2.3) sono i vincoli di connettività, garantiscono che ogni sottoinsieme di nodi presente nell'arborescenza (radice esclusa) sia raggiunto da almeno un arco entrante.

Il vincolo (2.4) impone la presenza dalla radice nell'arborescenza.

Il vincolo (2.5) è il vincolo di soglia, obbliga la raccolta di un insieme di pesi che valga almeno 0.

Infine i vincoli (2.6) e (2.7) impongono le condizioni di integralità.

2.3 Schema di risoluzione

Il problema è di per se difficile da risolvere a causa delle condizioni di integralità sulle variabili (2.6) e (2.7) e del vincolo di soglia (2.5), inoltre, il modello matematico soprastante è reso inutilizzabile nella pratica a causa dell'insieme di vincoli di connettività (2.3); tale insieme ha infatti cardinalità esponenziale. In letteratura, un modo noto di risolvere problemi con una simile struttura [10,11] è quello di trascurare i vincoli di connettività (2.3) e di rilassare le condizioni di integralità sulle variabili x e y.

Ne risulta un modello di *PL* facile da risolvere con l'algoritmo del simplesso (*CPLEX*). In seguito è mostrato il modello rilassato utilizzato dal branch & cut.

Modello rilassato:

$$
\begin{array}{lll}
min & \sum_{j=1}^{A} y_j & (2.8) \\
s.t. & & \\
& \sum_{j \in \delta^-(i)} y_j = x_i \qquad \forall i \in \mathcal{N} \setminus r & (2.9) \\
& x_r = 1 & (2.10) \\
& \sum_{i=1}^{N} w_i x_i \geq 0 & (2.11) \\
& 0 \leq x_i \leq 1 \qquad \forall i \in \mathcal{N} & (2.12) \\
& 0 \leq y_j \leq 1 \qquad \forall j \in \mathcal{A} & (2.13) \\
& x_i \geq y_j \qquad \forall i \in \mathcal{N} \setminus r, \forall j \in \delta^+(i) & (2.14) \\
& \sum_{j \in \delta^+(r)} y_j \geq 1 & (2.15) \\
& 2y_k + 2y_h \leq x_u + x_v \qquad \forall u \in \mathcal{N} : h = (u,v) \in \mathcal{A}, k = (v,u) \in \mathcal{A} & (2.16)
\end{array}
$$

Come si può notare, nel modello rilassato ho aggiunto ulteriori vincoli che ne rafforzano la formulazione. In particolare, il vincolo (2.14) obbliga un arco $j \in \mathcal{A}$ a far parte della soluzione rilassata ($y_j > 0$) solo se il nodo corrispondente alla sua coda è nella soluzione rilassata ($x_i > 0 : j = (i,v)$).

Il vincolo (2.15) impone che la somma degli archi uscenti dalla radice sia almeno 1. Il vincolo (2.16) impedisce la presenza di cicli di ordine 2.

I vincoli di connettività (2.3) trascurati nel modello rilassato, impongono (nel caso discreto) che una unità di flusso venga conservata per ogni cammino che si estende dalla radice fino al raggiungimento di ogni nodo terminale, la funzione obiettivo che tende a minimizzare il numero di archi (o nodi) garantisce l'assenza di cicli in soluzione.

Una possibile soluzione di tale modello rilassato difficilmente sarà costituita da un'arborescenza che conserva la quantità di flusso che si estende dalla radice ad ogni nodo terminale; è più verosimile che la soluzione del rilassamento contenga diverse componenti connesse e cicli.

Per migliorare la soluzione rilassata ed ottenere quindi dei bound duali più stringenti vengono aggiunti iterativamente dei vincoli violati in (2.3). Tali vincoli sono generati risolvendo il problema del flusso massimo a partire dalla soluzione frazionaria del problema rilassato. La procedura che genera vincoli violati in (2.3) risolvendo il problema del flusso massimo prende il nome di algoritmo di separazione. I vincoli in (2.3) aggiunti dinamicamente

al modello corrispondono al taglio minimo del digrafo corrispondente alla soluzione frazionaria del modello rilassato.

Flusso massimo e taglio minimo rappresentano infatti un caso molto semplice di teoria della dualità [21,22,23].

L'aggiunta di tutti e soli i vincoli in (2.3) violati corrisponde all'utilizzo di quella parte dell'insieme dei vincoli di connettività realmente necessari per il raggiungimento della soluzione ottima.

Nel continuo, la soluzione del modello rilassato comprensiva dei vincoli di connettività garantisce, in generale, di ottenere un digrafo connesso costituito da archi e nodi di valore frazionario. Non esiste alcuna garanzia che la soluzione frazionaria sia priva di cicli.

Lo schema generale dell'algoritmo branch & cut consiste nel generare un albero di branching dove in ogni sottoproblema viene iterativamente alternata la risoluzione del modello rilassato e dell'algoritmo di separazione sul digrafo G, dove la capacità di ogni arco $j \in \mathcal{A}$ coincide con il valore che tale arco assume nella soluzione corrente del modello rilassato y_j, analogamente, il flusso assegnato ad ogni nodo $i \in \mathcal{N}$ coincide con il valore che tale nodo assume nella soluzione corrente del modello rilassato.

L'esecuzione dell'algoritmo del flusso massimo avviene impostando come nodo sorgente il nodo radice $r \in \mathcal{N}$ e come nodo destinazione un qualsiasi nodo attualmente in base nel modello rilassato ($i \in \mathcal{N} : x_i > 0$). Chiaramente, viene trovato un nuovo taglio da aggiungere al rilassamento quando il flusso che raggiunge il nodo destinazione i è inferiore al valore di tale nodo (x_i) nella soluzione corrente del modello rilassato.

La scelta tra l'eseguire il branch o continuare ad aggiungere vincoli di taglio rappresenta una decisione determinante circa i tempi di esecuzione dell'algoritmo e verrà affrontata in seguito.

Lo schema relativo all'algoritmo branch & cut è esposto nel seguente algoritmo.

Data: Modello rilassato (2.8)-(2.16) M
Result: Una soluzione ottima per $MATC$
while *soluzione ottima non dimostrata* **do**
  estrai un sottoproblema p dalla lista dei sottoproblemi aperti;
  risolvi il modello rilassato M per p;
  while $\exists i \in \mathcal{N} : flusso_{ri} < x_i$ **do**
    esegui l'algoritmo di separazione ad aggiungi i vincoli violati al modello rilassato M;
    risolvi il modello rilassato M per p;
  end
  if $LB < UB$ **then**
    genera i sottoproblemi di p e inseriscili nella lista dei sottoproblemi aperti;
  end
end

Algorithm 1: Schema algortimo branch & cut

2.4 Preprocessing

Prima di eseguire l'algortimo branch & cut è possibile valutare l'esistenza di nodi dominati che sicuramente possono essere esclusi da una soluzione ottima. Nel caso in cui esistano nodi dominati, essi vengono eliminati dal digrafo insieme agli archi interessati, in questo modo si ottiene un digrafo di dimensioni minori.

Il criterio che utilizzo per etichettare un certo nodo come dominato, è quello di calcolare tutti i cammini minimi che partono dal nodo radice e terminano in ogni altro nodo del digrafo. Fra tutti questi cammini minimi è di particolare interesse il cammino $\mathcal{C}$, definito come il più corto (in termini di numero di archi utilizzati), il cui peso soddisfa il vincolo di soglia (o sensitività) sui pesi w_i associati ai nodi (2.5).

Il peso di un cammino è definito come la somma dei pesi w_i associati ai nodi che compongono il cammino stesso. Si inferisce che tutti i rimanenti nodi, il cui cammino minimo, ha una lunghezza (numero di archi utilizzati) non migliore di $\mathcal{C}$ possono essere scartati, perché non faranno mai parte di arborescenze che utilizzano un numero di archi inferiore di $\mathcal{C}$.

La ricerca del cammino minimo più corto che soddisfa il vincolo di soglia può essere eseguita modificando opportunamente l'algoritmo di visita in ampiezza di un grafo, dato che non esistono costi associati agli archi. L'estensione delle etichette avviene secondo il seguente criterio di dominanza: dati U cammini minimi distinti $\mathcal{U}_i$, (essendo cammini minimi, tutti i cam-

mini $\mathcal{U}_i$ hanno la stessa lunghezza e non esistono altri cammini di lunghezza minore) che dal nodo radice raggiungono un certo nodo u, il cammino $\mathcal{U}_d$ domina tutti i restanti cammini $\mathcal{U}_k, \forall k \in U : k \neq d$, se il peso associato al cammino minimo $\mathcal{U}_d$ è non inferiore ai pesi associati a tutti gli altri cammini minimi $\mathcal{U}_k$. Per ogni nodo visitato è memorizzato il peso massimo tra i pesi dei cammini minimi che lo raggiungo. Non appena viene raggiunto il valore di soglia 0 l'algoritmo termina. Il seguente algoritmo rappresenta ciò che ho appena descritto:

Data: Grafo $G = (\mathcal{N}, \mathcal{A})$; pesi $w_i \; \forall i \in \mathcal{N}$
Result: Lista dei nodi dominati l
$l := \emptyset$;
for i *in* $\mathcal{N}$ **do**
  $e_i :=$ TRUE;
  $maxP_i := -\infty$;
end
$e_r :=$ FALSE;
$maxP_r := w_r$;
$coda :=$ insert$(i \in \mathcal{N} : \exists(r, i) \in \mathcal{A})$;
$lunghezza := 1$;
$nodoFinale :=$ -1;
$end :=$ FALSE;
while $c \neq \emptyset$ *OR* end **do**
  $n :=$ extract$(coda)$;
  for $i \in \mathcal{N} : \exists(n, i) \in \mathcal{A}$ **do**
    if e_i **then**
      $coda :=$ insert(i);
      $maxP_i := max(maxP_i, maxP_n + w_n)$;
      $e_i :=$ FALSE;
      if $maxP_i > 0$ **then**
        $end :=$ TRUE;
        $nodoFinale := i$;
      end
    end
    $lunghezza := lunghezza + 1$;
  end
end
if end **then**
  da $lunghezza$ e $nodoFinale$ trova i nodi dominati e inseriscili in l;
end

Algorithm 2: Algoritmo per la ricerca di nodi dominati

L'algoritmo ha la stessa complessità computazionale dell'algoritmo di visita in ampiezza, ovvero $O(A)$ dato che il numero di archi domina il numero di nodi.

2.5 Aggiunta dei tagli violati

Ogni volta che la soluzione del modello rilassato vìola uno o più vincoli di connettività (2.3), è possibile aggiungere al modello i vincoli di taglio minimo corrispondenti, in modo che la successiva soluzione del modello rilassato soddisfi i tagli inseriti e migliori il bound duale.

Il taglio minimo tra un nodo sorgente $s \in \mathcal{N}$ e un nodo destinazione $t \in \mathcal{N}, t \neq s$ nel digrafo G è calcolato visitando il digrafo residuo ottenuto eseguendo l'algoritmo del flusso massimo da s a t in G dove il valore associato ai nodi $i \in \mathcal{N}$ e la capacità associata agli archi $j \in \mathcal{A}$ coincidono con il valore che assumono nella soluzione modello rilassato.

L'insieme di archi con capacità residua 0 trovati dall'algoritmo di visita sono il taglio minimo C_{st} che divide s da t.

Il vincolo corrispondente al taglio minimo C_{st} da aggiungere al modello rilassato ha la seguente forma:

$$\sum_{j \in C_{st}} y_j \geq x_t$$

In letteratura sono state studiate diverse tipologie di taglio, tra i più efficaci e utilizzati si parla di *nested cuts* [11] e *back cuts* [12].

In genere tali tipologie di vincoli vengono combinati per aggiungere un maggior numero di vincoli di taglio durante la stessa attivazione dell'algoritmo di separazione.

2.5.1 *Nested cuts*

L'idea di base dei *nested cuts* è quella di generare più di un taglio per ogni nodo $i \in \mathcal{N}$ che vìola il vincolo di flusso.

I *nested cuts* sono una sequenza di tagli, dove ogni taglio C_v, ad esclusione del primo, è definito per mezzo del taglio C_u che lo precede nell'ordinamento e dal quale gli archi ($j \in C_u$) sono considerati come se avessero capacità residua pari a 1. Il nuovo taglio C_v sarà quindi formato dal successivo insieme di archi con capacità residua pari a 0.

Un possibile insieme di *nested cuts* è quello rappresentato nella figura 2.2.

Come si può notare in 2.2A, dopo aver eseguito l'algoritmo del flusso massimo, il nodo destinazione t non rispetta il vincolo di flusso e il primo

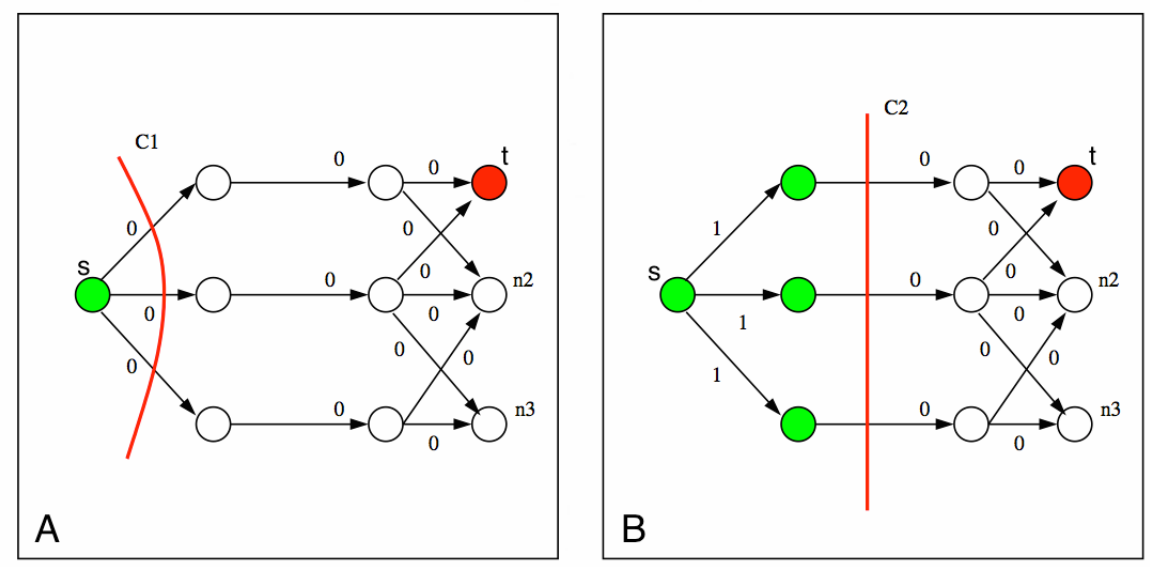

Figura 2.1: Vincoli *nested cuts*

taglio trovato dall'algoritmo di visita è il taglio $C1$. In 2.2B la capacità residua degli archi in $C1$ è impostata ad 1, e di conseguenza viene trovato il successivo *nested cut* $C2$.

2.5.2 *Back cuts*

Prendendo come esempio la figura 2.3, si può notare come in certe situazioni il taglio minimo non sia unicamente definito e come il taglio generato sia in prossimità del nodo sorgente s.

Per aumentare la diversità dei vincoli di taglio inseriti e migliorare più rapidamente il bound duale possono essere utilizzati i *back cuts*. I *back cuts* permettono di inserire tagli in prossimità del nodo destinazione semplicemente calcolando il flusso inverso tra i nodi s e t. Il flusso inverso è calcolato impostando come nodo sorgente ogni nodo $i \in \mathcal{N} : x_i > 0$ e come nodo destinazione il nodo radice.

Combinando *back cuts* o tagli di tipo *backward* con i tagli "normali" che da qui in poi chiamerò tagli di tipo *forward* si può ottenere la gradita situazione osservabile in figura 2.4. Ovviamente anche i *back cuts* possono essere combinati con i *nested cuts*.

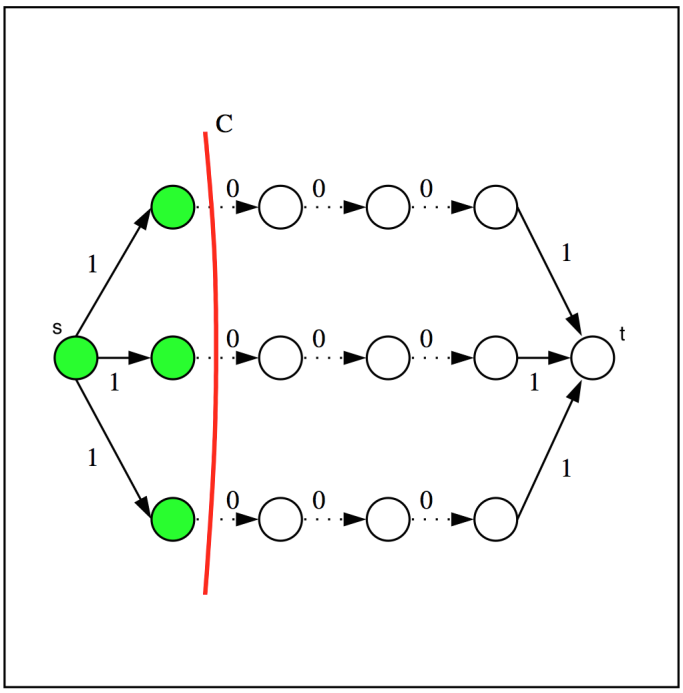

Figura 2.2: Taglio non unicamente definito

2.5.3 Diverse strategie di generazione tagli

Per ottenere bound duali più stringenti cercando al contempo di limitare il numero di vincoli inseriti nel modello rilassato, ho analizzato diverse strategie che combinano le tipologie di tagli introdotte precedentemente. Le strategie si differenziano per il criterio di generazione dei tagli per ogni nodo $i \in \mathcal{N}$ che vìola il vincolo sul flusso che si estende dalla radice, e per la combinazione delle diverse tipologie di taglio.

In tutte le strategie vengono utilizzati i *nested cuts*; tutte le strategie terminano quando ogni nodo $i \in \mathcal{N} : x_i > 0$ soddisfa il vincolo sul flusso che si estende dalla radice. Terminata la procedura di generazione dei tagli viene eseguito il branching.

- Strategia A: l'algoritmo di separazione genera per ogni nodo che vìola il vincolo di flusso un insieme di *nested cuts* di tipo *forward* o *backward*. La scelta tra la tipologia *forward* o *backward* avviene con probabilità $\frac{1}{2}$. Come criterio di arresto della generazione dei *nested cuts* utilizzo il raggiungimento della componente connessa a cui appartiene il nodo destinazione t

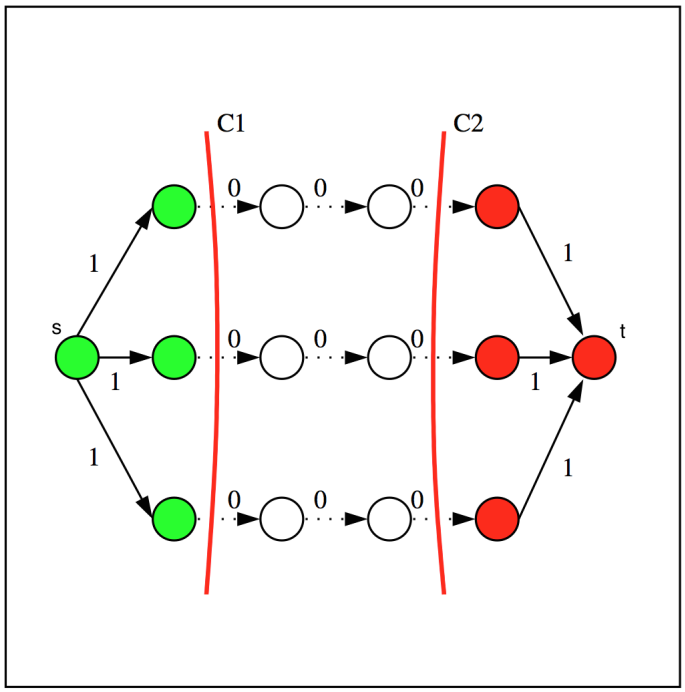

Figura 2.3: Combinazione di tagli *backward* e *forward*

- Strategia B: l'unica differenza rispetto alla strategia A, è il criterio di arresto della generazione dei *nested cuts*. In questo caso la generazione termina al raggiungimento del nodo destinazione t

- Strategia C: l'algoritmo di separazione genera per ogni nodo che vìola il vincolo di flusso un insieme di *nested cuts* sia di tipo *forward*, sia di tipo *backward*. Come criterio di arresto della generazione dei *nested cuts* utilizzo il raggiungimento della componente connessa a cui appartiene il nodo destinazione t

- Strategia D: l'unica differenza rispetto alla strategia C, è il criterio di arresto della generazione dei *nested cuts*. In questo caso la generazione termina al raggiungimento del nodo destinazione t

Il mio intento è quello di raggiungere, nel minor tempo possibile, in ogni nodo dell'albero di branching, una soluzione continua in cui ogni nodo soddisfa il vincolo sul flusso che si estende dalla radice verso ogni altro nodo in soluzione $i \in \mathcal{N} : x_i > 0$. Oltre ad ottenere bound duali più stringenti, in questa situazione, il digrafo corrispondente alla soluzione frazionaria del

rilassamento risulta essere connesso, ed è possibile sfruttarne la struttura per eseguire delle euristiche ed ottenere "buone" soluzioni ammissibili.

Come mostrerò in seguito, un vantaggio derivante dal calcolo di una soluzione primale in ogni sottoproblema, è la possibilità di ottenere in tempi brevi soluzioni "molto buone" troncando l'algoritmo branch & cut al nodo radice.

2.5.4 Confronto tra le strategie di generazione tagli

Per confrontare l'efficienza delle diverse strategie di generazione di tagli utilizzo dei grafici (figure 2.5, 2.6, 2.7, 2.8) che mostrano l'andamento del numero di vincoli aggiunti da ogni strategia, l'andamento del bound duale (LB) e i tempi richiesti da *CPLEX* per risolvere il modello rilassato.

Nella figura 2.9 confronto i LB raggiunti dalla diverse strategie al variare del tempo; nella figura 2.10, sempre al variare del tempo, sono paragonati il numero dei vincoli aggiunti dalla diverse strategie.

La strategia che considero migliore è quella che, al crescere della dimensione dell'istanza, tende a raggiungere bound duali più stringenti nel minor tempo.

Sono riportati i grafici relativi ad una singola istanza abbastanza complessa (griglia 25x25), dato che la tendenza è risultata pressoché identica per tutte le istanze di dimensioni uguali o maggiori. Per valutare l'efficacia dei vincoli inseriti per ogni nodo che non rispetta il vincolo di flusso, e per valutare come aumentano i tempi richiesti da *CPLEX* al crescere del numero di vincoli inseriti nel modello, eseguo iterativamente le seguenti azioni:

1. estraggo in modo casuale uniforme un nodo che vìola il vincolo sul flusso che si estende dalla radice
2. eseguo l'algoritmo di separazione per il nodo scelto al passo 1 trovando i vincoli di taglio
3. aggiungo al modello rilassato i vincoli di taglio generati al passo 2
4. eseguo l'ottimizzazione del modello rilassato

Per poter avere una base di confronto, per ogni strategia la procedura è interrotta dopo 200 secondi.

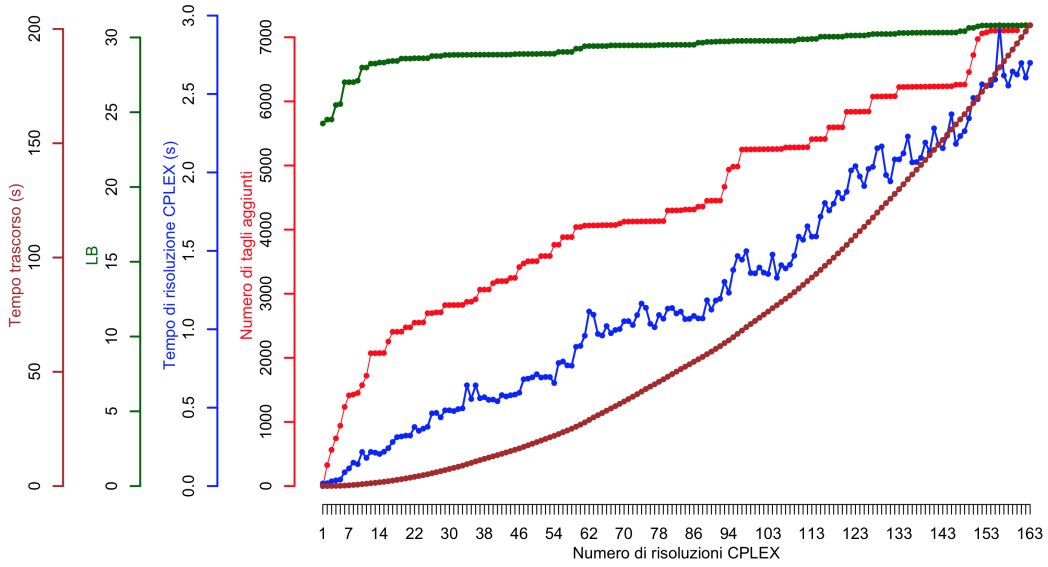

Figura 2.4: Valutazione della strategia A

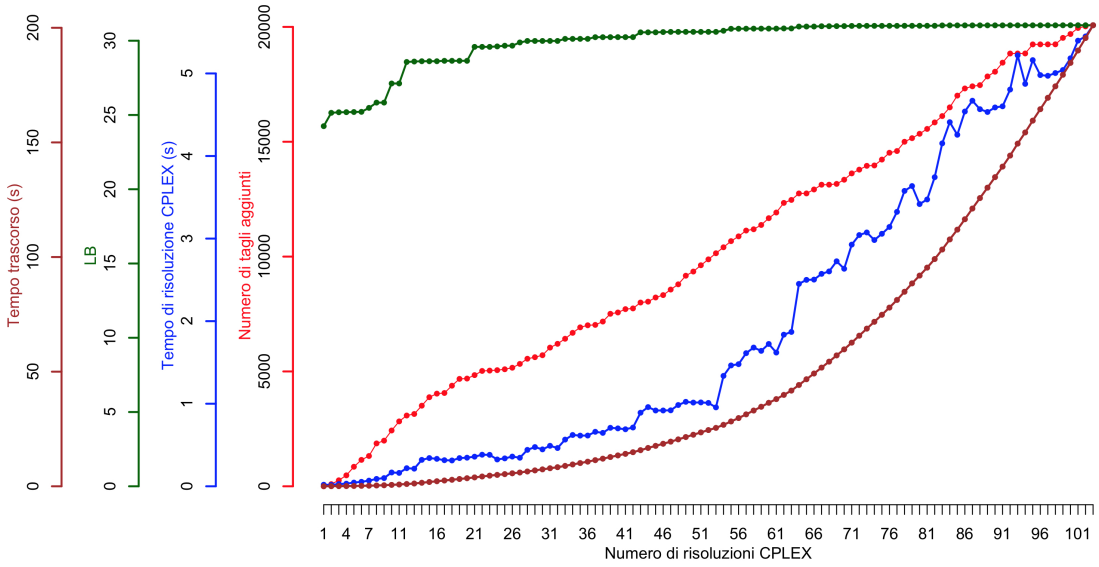

Figura 2.5: Valutazione della strategia B

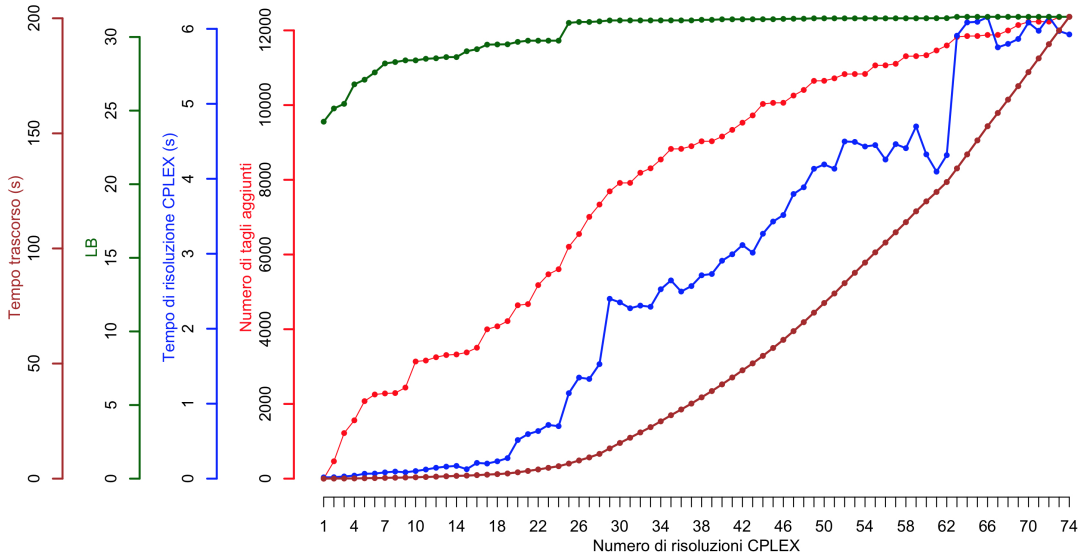

Figura 2.6: Valutazione della strategia C

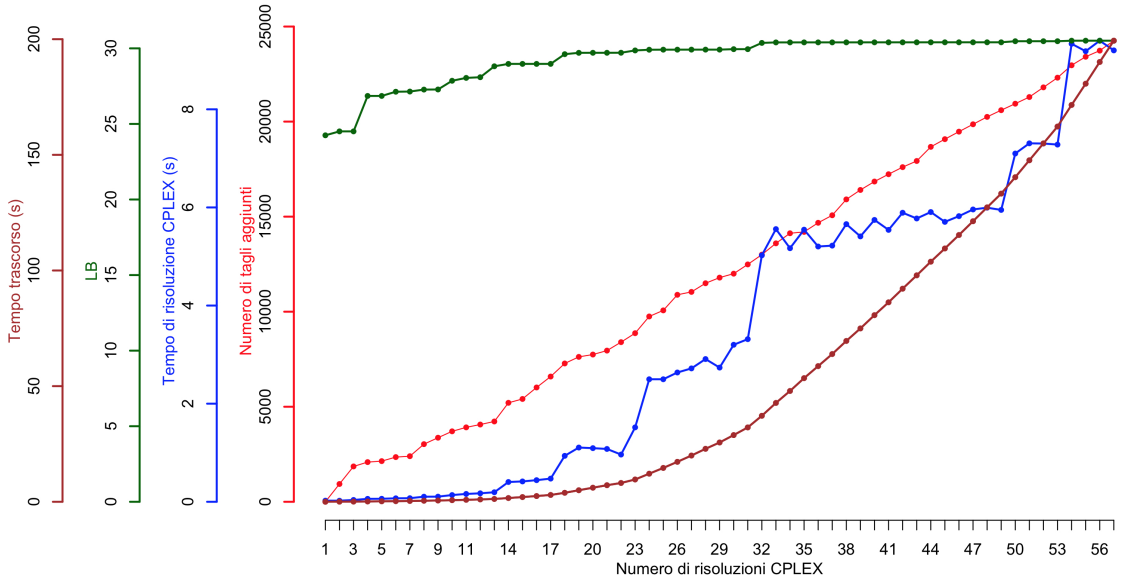

Figura 2.7: Valutazione della strategia D

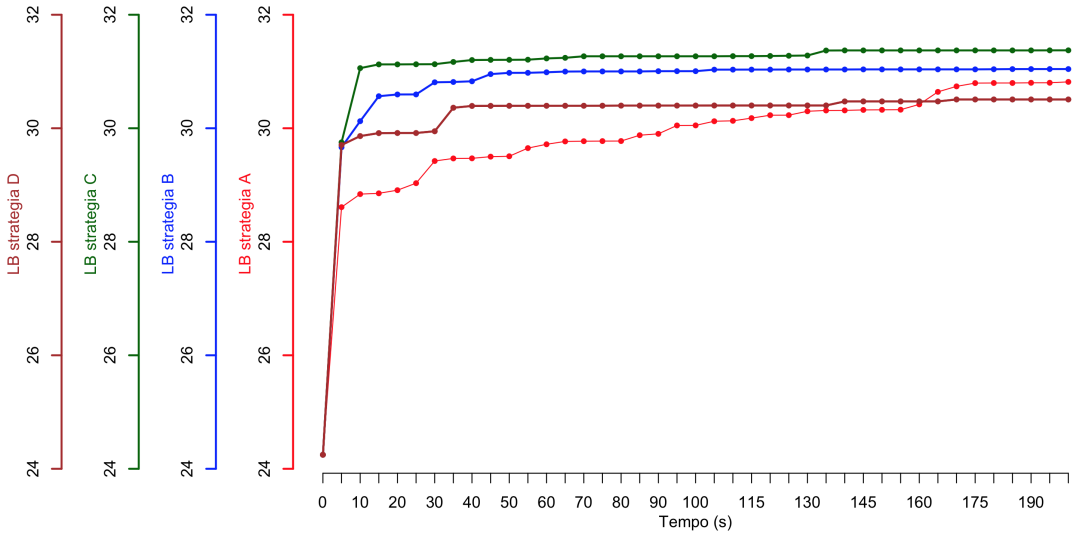

Figura 2.8: Confronto dei LB raggiunti dalle diverse strategie

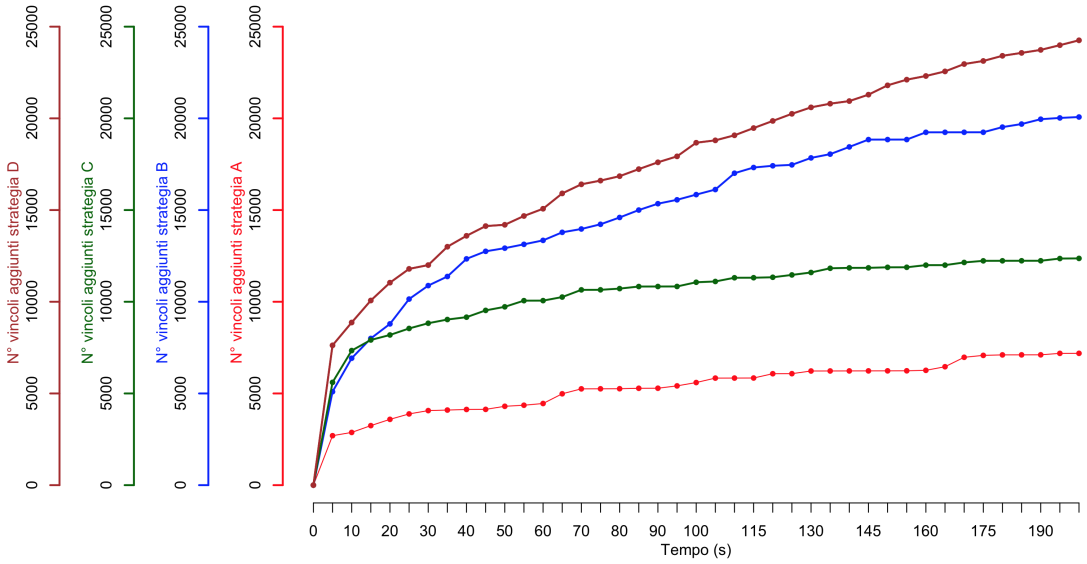

Figura 2.9: Confronto del numero di vincoli aggiunti dalle diverse strategie

I risultati finali delle diverse strategie sono riportati in tabella 2.1 per maggiore chiarezza.

Le strategie A, B e D raggiungono circa lo stesso valore di LB, la strategia C raggiunge un LB migliore del 2% rispetto alle altre. Oltretutto, in questo caso, la strategia C già dopo circa 10 secondi raggiunge un valore di LB che

	A	**B**	**C**	**D**
LB	30.86	30.95	31.57	30.82
Numero tagli	7188	20073	12364	24256

Tabella 2.1: Risultati delle diverse strategie

le altre strategie non riescono a raggiungere nemmeno al termine dei 200 secondi.

In dettaglio, la strategia A tende ad aggiungere "pochi" tagli per ogni nodo che vìola il vincolo di flusso, infatti risolve il modello rilassato un numero di volte superiore a tutte le altre strategie.

La strategia B è dominata dalla strategia A: raggiungono circa gli stessi valori di LB ma la strategia B aggiunge molti tagli in più, tanti dei quali evidentemente inutili. La strategia D è la strategia che aggiunge più vincoli, e di conseguenza è anche quella che richiede i tempi più lunghi per risolvere il rilassamento.

A conferma della maggior efficacia dei tagli aggiunti dalla strategia C, riporto nelle figure 2.11 e 2.12 dei grafici simili, che mostrano i tempi che impiegano le diverse strategie per raggiungere, nel problema radice, una situazione in cui ogni nodo soddisfa il vincolo di flusso.

Le figura 2.11 è relativa ad una istanza di dimensioni medie (griglia 20x20); la figura 2.12 è invece relativa ad una istanza più complessa (griglia 25x25). Il confronto tra due istanze di diversa dimensione permette di valutare la tendenza delle strategie al crescere della complessità delle istanze.

In questi casi, e nell'algoritmo branch & cut, il rilassamento non viene risolto ogni qual volta vengono aggiunti dei tagli, ma viene risolto solo quando sono stati aggiunti i tagli per tutti i nodi che violano il vincolo sul flusso che si estende dalla radice.

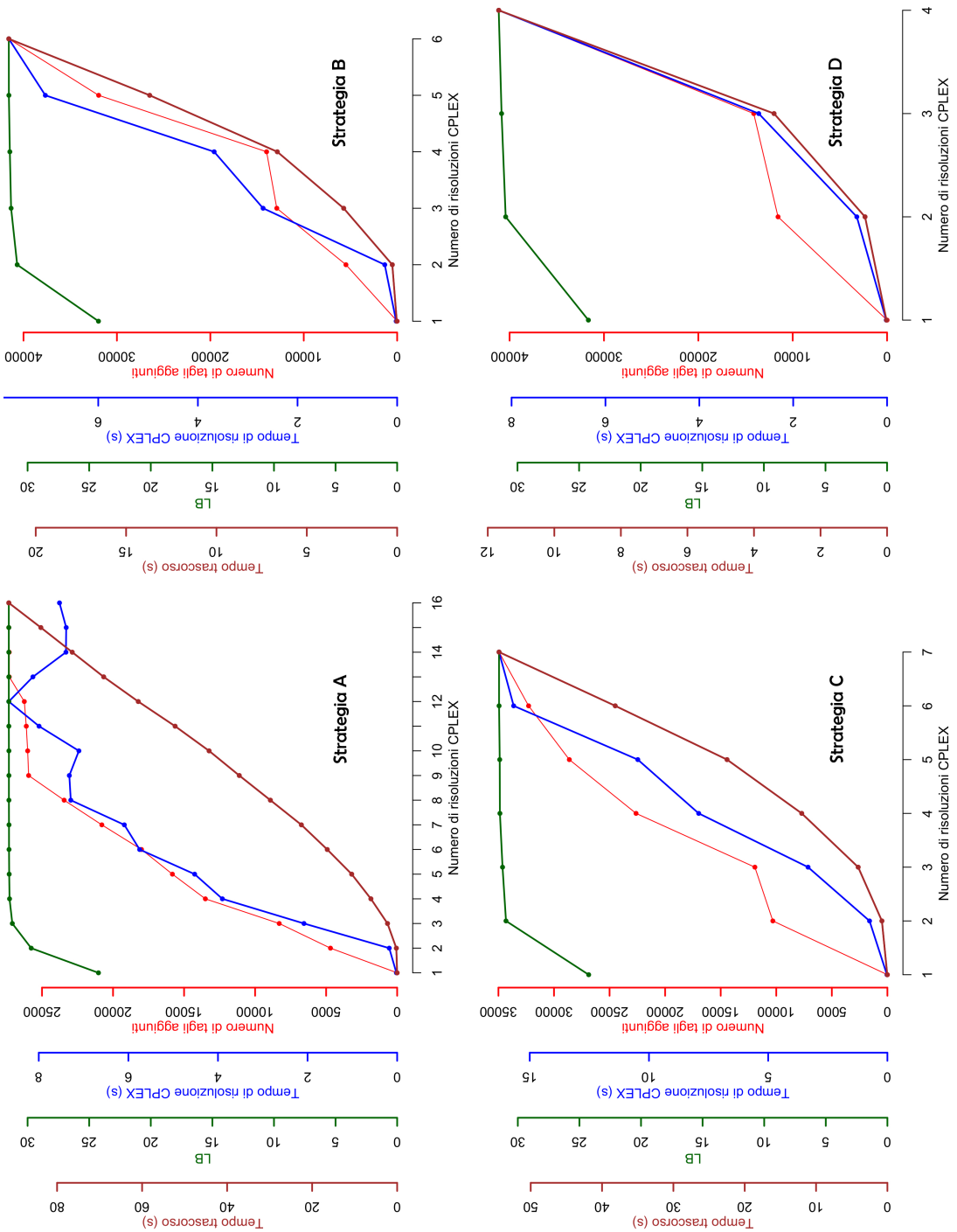

Figura 2.10: Le strategie su un'istanza 20x20

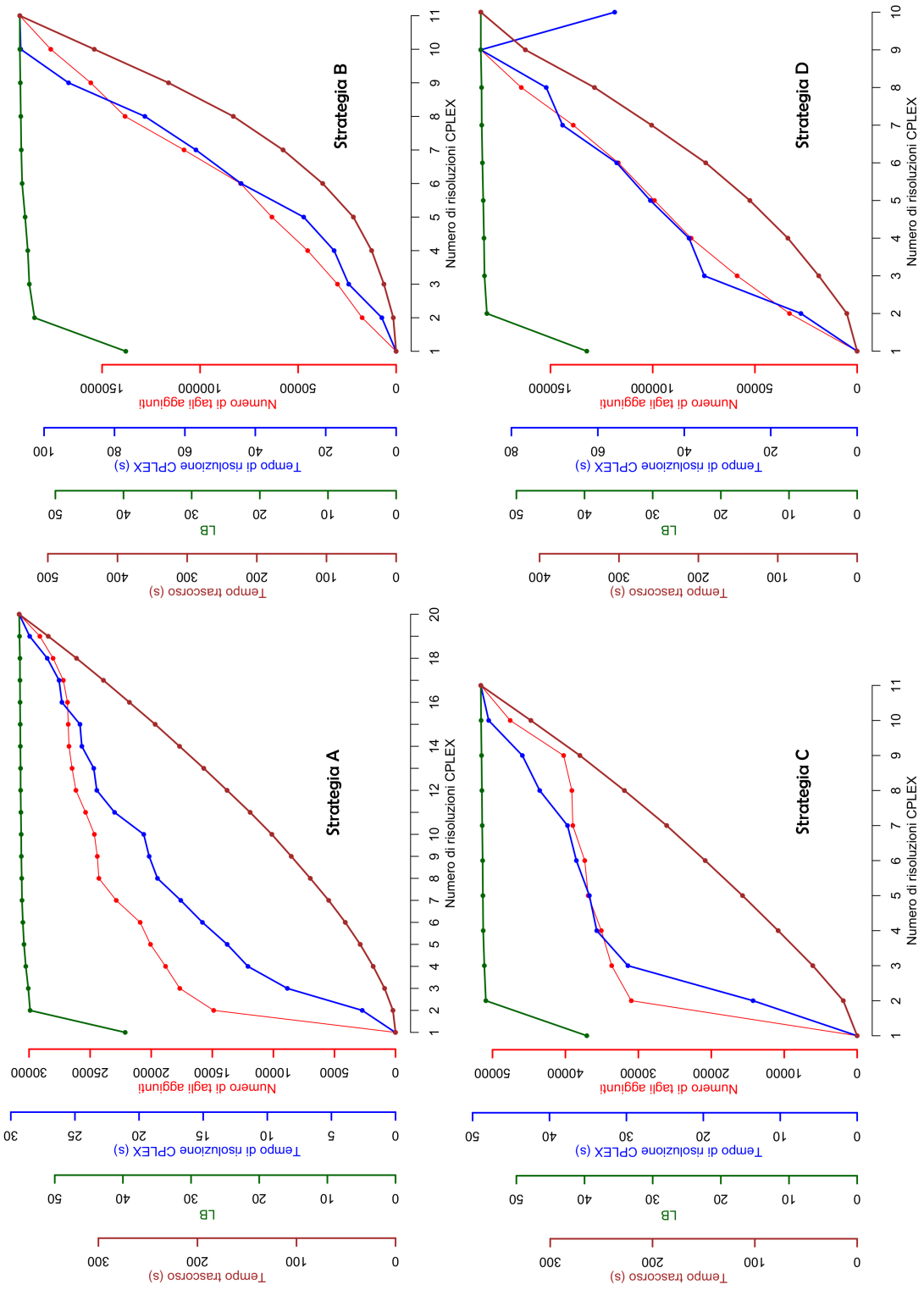

Figura 2.11: Le strategie su un'istanza 25x25

Dal confronto dei grafici nelle figure 2.11 e 2.12 emerge che il *trade off* migliore tra il numero di vincoli aggiunti e il numero di volte che bisogna risolvere il rilassamento dipende dalla complessità della particolare istanza del problema. Per istanze meno complesse (figura 2.11) le strategie B e D che aggiungono un numero maggiore di tagli sono più efficienti (impiegano rispettivamente 20 e 12 secondi) delle strategie A e C (impiegano rispettivamente 80 e 50 secondi). Quando l'istanza è più complessa (figura 2.12) conviene invece limitare il numero di vincoli aggiunti e risolvere un numero maggiore di volte un rilassamento più snello. Come si può notare dalla strategia A in figura 2.11, e dalla strategia D in figura 2.12, all'aumentare dei vincoli inseriti, non sempre i tempi richiesti da *CPLEX* per risolvere il rilassamento sono monotoni crescenti.

2.5.5 Algoritmo di separazione

L'algoritmo di separazione utilizzato genera i vincoli di taglio violati utilizzando la strategia C. Lo pseudocodice dell'algoritmo di separazione è il seguente:

Data: Modello rilassato (2.8)-(2.16) M
Result: Modello rilassato con formulazione rafforzata M
c := TRUE;
while c **do**
    c := FALSE;
    risolvi il modello rilassato M;
    dalla soluzione del modello rilassato M definisci la rete di flusso forward G_f e la rete di flusso backward G_b;
    for $i \in N : x_i > 0$ **do**
        f := maxFlow(G_f, r, i);
        if $f < x_i$ **then**
            usando la strategia C trova i tagli violati in G_f e in G_b e aggiungili al modello rilassato M;
            c := TRUE;
        end
    end
end

Algorithm 3: L'algoritmo di separazione

2.6 Euristica

Come già accennato precedentemente, in ogni sottoproblema dell'albero di branching si aggiungono tagli fino ad ottenere una soluzione del rilassamento in cui ogni nodo soddisfa il vincolo sul flusso che si estende dal nodo radice.

In tale situazione può essere molto conveniente sfruttare la struttura della soluzione del rilassamento per ottenere soluzioni euristiche, la soluzione euristica che genero in ogni sottoproblema corrisponde alla soluzione ottima intera della struttura frazionaria del modello rilassato.

Questa soluzione ottima è ottenuta mediante un modello matematico di PLI risolto da *CPLEX* sull'arborescenza derivante dal digrafo connesso della soluzione frazionaria del modello rilassato.

Preprocessing sul digrafo connesso frazionario

Quando ogni nodo $i \in \mathcal{N}$ soddisfa il vincolo di flusso che si estende dalla radice nel problema rilassato, il digrafo $G' = (\mathcal{N}', \mathcal{A}')$ costituito dagli archi $j \in \mathcal{A} : y_j > 0$ e dai nodi $i \in \mathcal{N} : x_i > 0$ è connesso ma può contenere cicli.

Per generare un modello di PLI il più possibile efficiente, vengono eliminati tutti gli archi $j \in \mathcal{A}'$ in eccesso, gli archi in eccesso sono quelli che raggiungono nodi $i \in \mathcal{N}'$ già visitati dall'algoritmo di visita in ampiezza. Eliminati gli archi in eccesso si ottiene un digrafo $G'' = (\mathcal{N}'', \mathcal{A}'')$ dove ogni nodo (ad eccezione della radice) $i \in \mathcal{N}''$ è raggiunto da un unico arco $j \in \mathcal{A}''$. G'' corrisponde quindi ad una arborescenza.

È importante far notare come l'eliminazione degli archi in accesso non influisce sulla soluzione ottima intera della struttura frazionaria del modello rilassato, questo perché l'insieme dei nodi (a cui sono associati i pesi w_i) non subisce variazioni.

Modello matematico sull'arborescenza frazionaria

Poiché in una arborescenza, ogni nodo ad esclusione del nodo radice è raggiunto da un unico arco, è possibile costruire un modello di PLI che, a partire dall'arborescenza G'', trovi l'arborescenza di costo minimo in G'' che soddisfa il vincolo di soglia (2.5) considerano solamente gli archi $j \in \mathcal{A}''$ e trasferendo il peso $w_i : i \in \mathcal{N}''$ associato ai nodi all'arco $j \in \mathcal{A}'' : j = (u, i)$.

Il modello di PLI è quindi costituito dalle sole variabili t_j associate ad ogni arco $j \in \mathcal{A}''$.

Il modello risulta essere estremamente rapido (tempi in figura 2.13). Anche quando il numero di archi è considerevole (tra i 60 e i 100 archi) i tempi superano in rari casi il centesimo di secondo.

$$
\begin{aligned}
min \quad & \textstyle\sum_j t_j & & (2.17)\\
s.t. \quad & & & \\
& \textstyle\sum_j w_j t_j \geq 0 & & (2.18)\\
& t_i \geq t_j & \forall j \in \delta^+(q) : i = (v, q) & (2.19)\\
& t_j \in \{0, 1\} & \forall j \in \mathcal{A}'' & (2.20)
\end{aligned}
$$

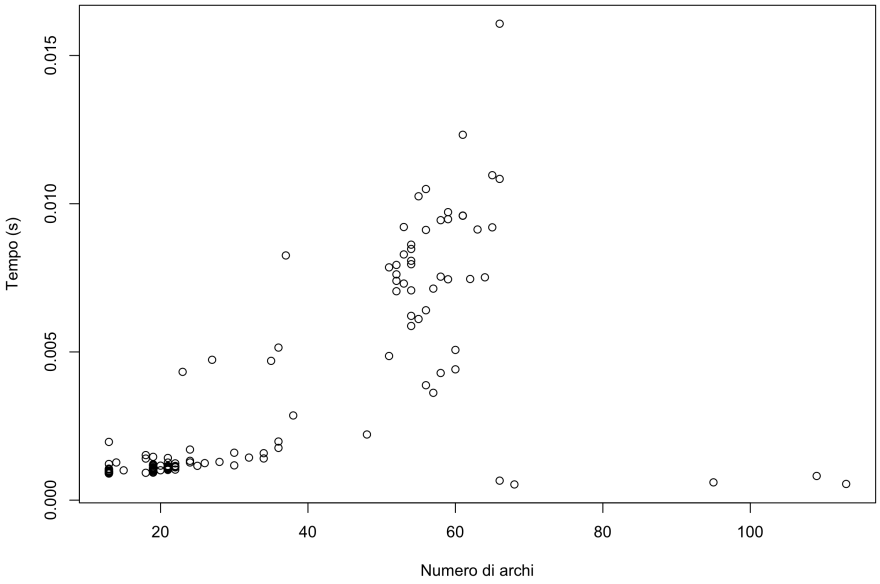

Figura 2.12: I tempi di risoluzione del modello rispetto al numero di archi

2.7 Strategie di visita e di branching

Viene utilizzato un branching dicotomico sulle variabili x_i associate ai nodi $i \in \mathcal{N}$. Eseguo il branching quando ogni nodo $i \in \mathcal{N} : x_i > 0$ soddisfa il vincolo di flusso che si estende dalla radice, la scelta del nodo $i \in \mathcal{N} : x_i > 0$ ricade sul nodo più frazionario, ovvero sul nodo i avente il valore x_i più prossimo a $\frac{1}{2}$. Sono state valutate diverse strategie di visita dei sottoproblemi:

1. *Best first search* valuta i sottoproblemi in ordine crescente di LB, la struttura dati che ho utilizzato per l'estrazione, ad ogni iterazione, del sottoproblema con minimo valore di LB è uno heap

2. *Depth first search* è la visita in profondità; la struttura dati necessaria per la sua l'implementazione è lo stack, ad ogni iterazione è valutato l'ultimo problema generato

3. *Breadth first search* è la visita in ampiezza: la struttura dati necessaria per la sua l'implementazione è la coda, i problemi sono valutati nello stesso ordine con cui sono generati.

4. *Mix search* è una strategia che ho proposto, motivata dall'idea che un fissaggio a 1 sia più importante di un fissaggio a 0. Utilizzando la struttura dati *deque* inserisco in testa i sottoproblemi con vincolo di branch a 1 e in coda i sottoproblemi in cui il vincolo di branch è eseguito con valore 0

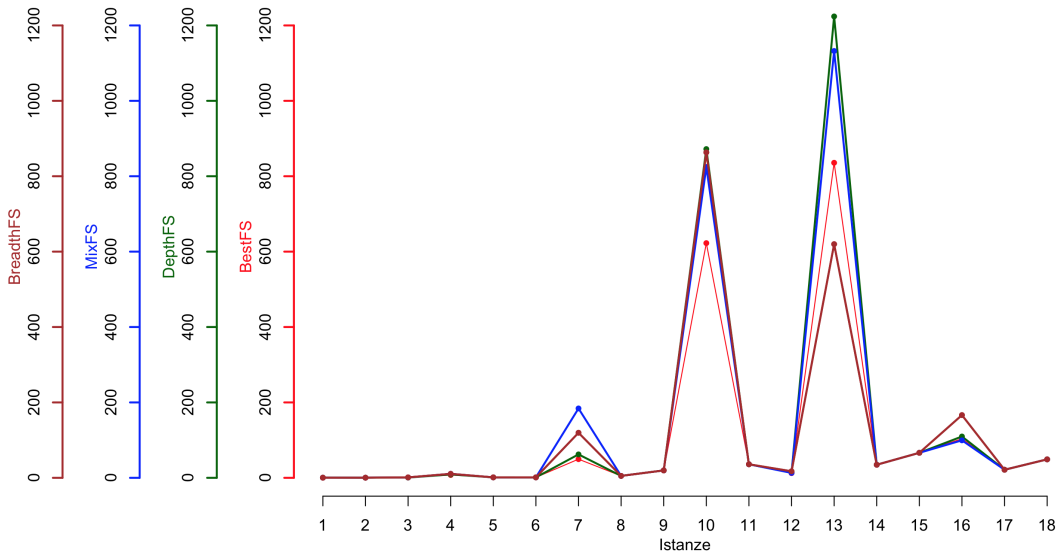

Figura 2.13: I tempi di risoluzione delle diverse strategie di visita

Come mostra la figura 2.14, non esistono strategie di visita sempre migliori delle altre. La strategia *BestFS* risulta essere preferibile nella maggior parte dei casi.

2.8 Generazione delle istanze

Per valutare le prestazioni dell'algoritmo branch & cut ho generato diverse istanze per il problema *MATC*.

Le prestazioni dell'algoritmo branch & cut sono state valutate anche per una variante del problema *MATC* che corrisponde al caso in cui ad ogni cella venga semplicemente assegnata una probabilità ≥ 0 (probabilità radice $= 0$) senza passare per il calcolo dei pesi w_i, in questo caso la soglia da

raggiungere è una certa percentuale di probabilità indicata sempre con τ. É pertanto possibile valutare la complessità introdotta dall'utilizzo di pesi w_i sia positivi che negativi.

I dati sono stati generati nel modo seguente:

- Nei test dove ad ogni cella è assegnata una probabilità ≥ 0 senza utilizzare i pesi w_i, le regole seguite per la generazione delle istanze sono le seguenti:
  - La griglia è quadrata, e viene popolata con valori di probabilità (somma pari a 1) generati da gaussiane bivariate, la cella della griglia che costituisce il centro della gaussiana è scelto in modo casuale uniforme
  - Cercando di replicare le istanze reali, la generazione dei valori di probabilità è arrestata quando almeno il 70% delle celle ha associata una probabilità > 0
  - La posizione del nodo radice è scelta in modo casuale uniforme tra le celle con probabilità 0
  - Ogni istanza è testata su tre diversi valori di soglia, corrispondenti rispettivamente al 30, 50 e 70 % di probabilità
- Nei test dove ad ogni cella è assegnato un peso w_i che può essere sia positivo che negativo, le regole seguite per la generazione delle istanze sono le seguenti:
  - La griglia è quadrata, e viene popolata con valori di probabilità (somma pari a 1) generati da gaussiane bivariate, la cella della griglia che costituisce il centro della gaussiana è scelto in modo casuale uniforme
  - Cercando di replicare le istanze reali, la generazione dei valori di probabilità è arrestata quando almeno il 70% delle celle ha associata una probabilità > 0
  - Una volta arrestata la generazione dei valori di probabilità ρ_i, viene scelta in modo casuale la dimensione, la posizione e la radice di una regione base. La posizione è scelta in modo che, ad ogni cella base della regione base, corrisponda un valore di probabilità > 0.

    Cercando di replicare le istanze reali, la dimensione della regione base varia casualmente tra un minimo di 2 righe e colonne, ad un massimo di 4 righe e colonne.

– I valori di probabilità sono convertiti in pesi w_i mediante la relazione $w_i = \rho_i(\tau - s_i)$.

2.9 Risultati sperimentali

I risultati ottenuti vengono suddivisi in tre tabelle. Nella prima tabella (2.2) mostro i risultati ottenuti nel caso in cui ad ogni nodo è semplicemente assegnata una probabilità ρ_i (non sono calcolati i pesi w_i), in questo caso il valore di soglia da raggiungere è una percentuale di probabilità (la radice ha associata una probabilità pari a 0). Ho scelto di mostrare anche questo caso per fornire un'idea di quanto l'introduzione di pesi w_i sia positivi che negativi complichi il problema.

Nelle altre due tabelle (2.3 e 2.4) mostro i risultati ottenuti per il problema *MATC* ovvero nel caso in cui le probabilità ρ_i sono convertite in pesi w_i che possono assumere sia valori positivi, sia valori negativi.

In particolare, nelle istanze della tabella 2.3, la regione base in cui è inserita la radice ha dimensioni 2x2, nelle istanze della tabella 2.4, la regione base in cui è inserita la radice ha dimensioni 4x4. In entrambi i casi la posizione della radice all'interno della regione base è scelta in modo casuale uniforme.

Poiché per la risoluzione di *MKFTC* non risulta essere valido l'utilizzo del preprocessing così come descritto in 2.4 (il perché è spiegato in 3.3), i risultati presentanti sono ottenuti senza utilizzare la fase di preprocessing.

Il significato dei vari campi delle tabelle è il seguente:

1. Istanza: è il nome dell'istanza; la coppia di interi separati dal carattere "x" indica la dimensione della griglia, il valore numerico finale rappresenta la soglia τ

2. LBR: è il valore di LB ottenuto nel nodo radice quando ogni nodo $i \in \mathcal{N}$ soddisfa il vincolo di flusso

3. TCR: è il tempo, espresso in secondi, necessario a *CPLEX* per risolvere iterativamente il rilassamento fino ad ottenere la situazione in cui ogni nodo $i \in \mathcal{N}$ soddisfa il vincolo di flusso. Coincide con il tempo cumulato utilizzato da *CPLEX* per ottenere il valore corrispondente del campo LBR

4. NCR: è il numero dei vincoli corrispondenti al taglio minimo aggiunti al rilassamento per arrivare alla situazione in cui ogni nodo $i \in \mathcal{N}$ soddisfa il vincolo di flusso

5. UBR: è il valore del bound primale ottenuto nel nodo radice mediante il modello di *PLI* descritto in 2.5.2
6. OS: è il valore della soluzione ottima dell'istanza
7. GAPC: è la percentuale rappresentante la distanza tra la soluzione ottima e il LBR non arrotondato all'intero superiore (poiché la funzione obiettivo minimizza il numero di nodi presenti nell'arborescenza, possiamo arrotondare ogni LB all'intero superiore)
8. GAPI: è la percentuale rappresentante la distanza tra la soluzione ottima e il LBR arrotondato all'intero superiore
9. NS: è il numero di sottoproblemi generati
10. T: è il tempo, espresso in secondi, necessario all'algoritmo branch & cut per risolvere il problema *MATC*

Istanza	LBR	TCR	NCR	UBR	OS	GAPC	GAPI	NS	T
1-dataSAP1-15x15-0.3	19.96	0.06	2238	21	21	4.96	4.76	4	0.39
2-dataSAP1-15x15-0.5	35.94	0.05	4305	36	36	0.17	0.00	0	0.36
3-dataSAP1-15x15-0.7	58.95	0.38	2759	60	60	1.75	1.67	2	1.00
4-dataSAP2-15x15-0.3	15.05	1.41	3869	17	17	11.46	5.88	20	7.52
5-dataSAP2-15x15-0.5	26.58	0.91	3696	27	27	1.56	0.00	0	1.22
6-dataSAP2-15x15-0.7	41.17	0.68	7486	42	42	1.98	0.00	0	1.21
7-dataSAP3-15x15-0.3	23.12	0.45	3155	24	24	3.67	0.00	0	1.03
8-dataSAP3-15x15-0.5	29.30	0.87	5367	31	30	2.33	0.00	0	2.45
9-dataSAP3-15x15-0.7	42.43	1.04	6948	43	43	1.33	0.00	0	2.11
10-dataSAP1-20x20-0.3	21.07	3.72	6055	25	23	8.38	4.35	18	49.49
11-dataSAP1-20x20-0.5	35.50	3.57	13579	36	36	1.39	0.00	0	5.10
12-dataSAP1-20x20-0.7	56.75	15.51	34825	57	57	0.44	0.00	0	19.66
13-dataSAP2-20x20-0.3	31.49	42.32	26213	34	33	4.58	3.03	28	622.57
14-dataSAP2-20x20-0.5	54.50	18.14	37523	55	55	0.91	0.00	0	35.95
15-dataSAP2-20x20-0.7	94.33	0.36	30778	95	95	0.70	0.00	0	12.39
16-dataSAP3-20x20-0.3	18.08	0.03	1945	19	19	4.84	0.00	0	0.65
17-dataSAP3-20x20-0.5	27.89	0.78	3449	29	28	0.39	0.00	0	3.45
18-dataSAP3-20x20-0.7	65.43	2.45	12311	67	68	3.78	2.94	4	13.34
19-dataSAP1-25x25-0.3	31.52	17.94	26423	33	33	4.48	3.03	30	835.72
20-dataSAP1-25x25-0.5	64.20	4.13	42997	65	65	1.23	0.00	0	34.70
21-dataSAP1-25x25-0.7	113.00	15.32	71861	113	113	0.00	0.00	0	66.35
22-dataSAP2-25x25-0.3	35.65	33.69	33440	37	37	3.65	2.70	4	103.57
23-dataSAP2-25x25-0.5	67.14	3.10	25822	68	68	1.26	0.00	0	21.40
24-dataSAP2-25x25-0.7	109.00	0.49	69610	109	109	0.00	0.00	0	48.99
25-dataSAP3-25x25-0.3	36.13	3.03	14349	37	37	2.35	0.00	0	12.67
26-dataSAP3-25x25-0.5	50.17	5.44	17349	51	51	1.63	0.00	0	29.45
27-dataSAP3-25x25-0.7	66.78	9.78	24511	68	68	1.79	1.47	8	67.31

Tabella 2.2: Risultati ottenuti per il caso in cui ad ogni nodo sono associati solo valori ≥ 0

Istanza	**LBR**	**TCR**	**NCR**	**UBR**	**OS**	**GAPC**	**GAPI**	**NS**	**T**
1-dataSAW1-15x15-0.05	7.72	0.01	521	9	9	14.20	11.11	4	0.18
2-dataSAW1-15x15-0.1	4.67	0.03	818	7	7	33.33	28.57	42	2.36
3-dataSAW1-15x15-0.2	2.50	0.03	1015	5	5	50.00	40.00	84	53.97
4-dataSAW1-15x15-0.4	1.47	0.04	926	4	4	63.21	50.00	148	319.78
5-dataSAW2-15x15-0.05	9.48	8.86	8152	16	16	40.77	37.50	8	86.77
6-dataSAW2-15x15-0.1	4.00	0.08	1945	5	5	20.00	20.00	8	0.65
7-dataSAW2-15x15-0.2	2.43	0.13	1881	4	4	39.29	25.00	28	2.51
8-dataSAW2-15x15-0.4	1.28	0.12	1920	3	2	36.21	0.00	0	0.79
9-dataSAW1-20x20-0.05	10.63	44.22	19720	12	12	11.38	8.33	4	205.43
10-dataSAW1-20x20-0.1	5.54	35.98	12950	8	6	7.62	0.00	0	471.54
11-dataSAW1-20x20-0.2	2.00	0.01	0	2	2	0.00	0.00	0	0.02
12-dataSAW1-20x20-0.4	1.00	0.01	0	1	1	0.00	0.00	0	0.01
13-dataSAW2-20x20-0.05	7.19	0.71	7119	8	8	10.16	0.00	0	3.88
14-dataSAW2-20x20-0.1	3.15	0.14	1555	4	4	21.15	0.00	0	0.92
15-dataSAW2-20x20-0.2	2.25	0.36	1133	3	3	25.00	0.00	0	0.62
16-dataSAW2-20x20-0.4	1.31	0.27	2002	2	2	34.62	0.00	0	1.19
17-dataSAW1-25x25-0.05	32.50	5.92	19432	38	38	14.47	13.16	304	4670.56
18-dataSAW1-25x25-0.1	11.61	3.02	7175	12	12	3.28	0.00	0	6.98
19-dataSAW1-25x25-0.2	3.57	0.01	0	4	4	10.71	0.00	0	0.03
20-dataSAW1-25x25-0.4	1.00	0.02	0	1	1	0.00	0.00	0	0.03
21-dataSAW2-25x25-0.05	6.75	13.62	15319	12	7	3.60	0.00	0	355.66
22-dataSAW2-25x25-0.1	3.00	2.01	5427	3	3	0.00	0.00	0	4.45
23-dataSAW2-25x25-0.2	1.80	0.98	7372	2	2	10.00	0.00	0	5.56
24-dataSAW2-25x25-0.4	1.10	1.01	6467	2	2	45.24	0.00	0	5.12

Tabella 2.3: Risultati ottenuti per il caso in cui sono utilizzati i pesi w_i e regione base 2x2

Istanza	**LBR**	**TCR**	**NCR**	**UBR**	**OS**	**GAPC**	**GAPI**	**NS**	**T**
1-dataSAW1-15x15-0.05	9.13	0.12	1023	12	12	23.92	16.67	36	32.21
2-dataSAW1-15x15-0.1	5.03	0.08	956	8	8	37.13	25.00	22	9.34
3-dataSAW1-15x15-0.2	2.76	0.04	1145	7	6	54.00	50.00	48	65.45
4-dataSAW1-15x15-0.4	2.12	0.03	998	5	5	57.60	40.00	92	176.45
5-dataSAW2-15x15-0.05	13.55	15.78	12333	23	21	35.48	33.33	58	895.98
6-dataSAW2-15x15-0.1	5.34	2.23	4389	9	8	33.25	25.00	8	23.13
7-dataSAW2-15x15-0.2	3.12	0.39	2221	6	6	48.00	33.33	94	57.43
8-dataSAW2-15x15-0.4	2.62	0.23	2134	4	4	34.50	25.00	16	7.65
9-dataSAW1-20x20-0.05	9.34	13.78	13455	11	11	15.09	9.09	4	89.45
10-dataSAW1-20x20-0.1	4.12	14.31	9423	7	6	31.33	16.67	22	238.42
11-dataSAW1-20x20-0.2	3.11	0.31	764	4	4	22.25	0.00	0	12.56
12-dataSAW1-20x20-0.4	2.13	12.00	453	4	3	29.00	0.00	0	7.89
13-dataSAW2-20x20-0.05	7.56	0.56	5674	8	8	5.50	0.00	0	5.64
14-dataSAW2-20x20-0.1	3.14	0.11	1342	4	4	21.50	0.00	0	1.98
15-dataSAW2-20x20-0.2	2.02	0.09	674	3	3	32.67	0.00	0	1.54
16-dataSAW2-20x20-0.4	1.91	0.04	19	2	2	4.50	0.00	0	0.09
17-dataSAW1-25x25-0.05	39.98	7.98	20134	48	47	14.94	14.89	66	534.98
18-dataSAW1-25x25-0.1	21.32	5.98	9842	27	27	21.04	18.52	128	499.66
19-dataSAW1-25x25-0.2	8.34	2.28	2898	12	12	30.50	25.00	84	231.98
20-dataSAW1-25x25-0.4	3.24	0.89	958	4	4	19.00	0.00	0	12.23
21-dataSAW2-25x25-0.05	9.34	21.12	21656	14	13	28.15	23.08	28	756.23
22-dataSAW2-25x25-0.1	5.45	9.32	7853	8	7	22.14	14.29	12	161.79
23-dataSAW2-25x25-0.2	4.12	3.21	7664	5	5	17.60	0.00	0	98.34
24-dataSAW2-25x25-0.4	3.67	0.43	5234	4	4	8.25	0.00	0	3.72

Tabella 2.4: Risultati ottenuti per il caso in cui sono utilizzati i pesi w_i e regione base 4x4

Dal confronto dei risultati si nota come il problema, nel caso in cui sono utilizzati i pesi w_i che possono essere sia positivi che negativi, sia generalmente più difficile da risolvere. Lo testimoniano sopratutto le colonne relative al *gap* che mostrano una varianza molto più alta rispetto al caso in cui i pesi w_i non sono utilizzati. In particolare, più è grande la regione base, più il *gap* tra LB e UB tende ad aumentare.

Quando sono utilizzati i pesi w_i, la soluzione frazionaria del rilassamento combina generalmente molte più variabili rispetto al caso in cui ai nodi sono associati solo valori non negativi. Ciò determina sia un allargamento del *gap* tra LB e UB in ogni sottoproblema, sia la necessità di aggiungere più vincoli di taglio, dato che la soluzione è composta da piccole frazioni di tante variabili.

Come logica conseguenza, il problema che utilizza i pesi w_i richiede un tempo maggiore, soprattutto quando il valore di soglia τ è piccolo (0.05) ed al nodo radice è associato un elevato valore di probabilità ρ_i.

In questi casi infatti il peso w_i corrispondente alla radice $w_i = \rho_i(\tau - 1)$ è molto negativo (caso che si verifica nell'istanza *17-dataSAW1-25x25-0.05*). All'aumentare del valore di τ, i pesi w_i associati alle celle base sono meno negativi, i pesi w_i associati alle celle esterne sono più alti, e quindi il problema diventa più semplice, perché sono necessari meno nodi per soddisfare il vincolo di soglia.

Oltretutto, i test per i casi in cui non vengono utilizzati i pesi w_i sono stati generati in modo da complicare il processo risolutivo. La posizione della radice è infatti scelta in modo casuale in una posizione a cui corrisponde una probabilità ρ_i pari a 0. In questo modo si rende necessario l'attraversamento di nodi non favorevoli (non contribuiscono al raggiungimento del vincolo di soglia, $\rho_i = 0$) prima di raggiungere nodi con $\rho_i > 0$.

Per il caso in cui sono utilizzati i pesi w_i, è necessario invece collocare l'intera regione base in una porzione di griglia corrispondente a valori ρ_i strettamente positivi per come è definito il problema (se il valore ρ_i fosse 0 per una cella base, questa non avrebbe poi associato un peso w_i negativo).

Diventa pertanto più probabile, in questo caso, che nell'immediata vicinanza della radice siano presenti nodi con associati pesi $w_i > 0$. Infatti, le istanze con valore valore di $\tau > 0.05$, sono spesso risolte velocemente indipendentemente dalla dimensione dell'istanza.

Una nota positiva che si verifica in entrambe le situazioni, è il raggiungimento, in molti casi, della soluzione ottima direttamente nel nodo radice. La garanzia di ottimalità, come spesso accade negli algoritmi di branch & bound, è difficile da garantire tempi brevi.

Capitolo 3

Algoritmo branch and price

In questo capitolo descrivo l'algoritmo che calcola la soluzione esatta per il problema *MKFTC*. La formulazione adottata prevede l'utilizzo di una variabile binaria per ogni possibile arborescenza che soddisfa il vincolo di soglia.

Il numero di queste possibili arborescenze è esponenziale e quindi anche la formulazione contempla un numero esponenziale di variabili rispetto alla dimensione del problema. È possibile tuttavia considerare solo un sottoinsieme delle variabili e delle relative colonne nella matrice dei vincoli, ottenendo un problema ridotto.

Partendo da una soluzione di base ammissibile, si possono inserire nel problema ridotto solo le colonne corrispondenti a variabili candidate ad entrare in base. Il problema di determinare queste variabili candidate è detto *pricing problem*.

Il *pricing problem* va alla ricerca di colonne aventi un costo ridotto opportuno (negativo per i problemi di minimizzazione). La procedura di individuazione ed inserimento iterativo di opportune colonne nel problema ridotto è detta *column generation*. L'utilizzo del *column generation* nei problemi di ottimizzazione (mista) intera necessita dell'enumerazione implicita delle soluzioni e porta ad algoritmi di branch & price.

3.1 Modello matematico per MKFTC

Dati:

1. Siano $\mathcal{N} = \{1 \dots N\}$ ed $\mathcal{A} = \{1 \dots A\}$ rispettivamente l'insieme dei nodi e degli archi di un digrafo $G = (\mathcal{N}, \mathcal{A})$ privo di autoanelli, l'insieme dei nodi $\mathcal{N}$ è partizionato in due insiemi disgiunti $\mathcal{N} = \mathcal{B} \cup \mathcal{E}$

2. Sia $\mathcal{B} = \{1 \dots B\}$ l'insieme dei nodi base, ogni nodo $v \in \mathcal{B}$ è detto nodo base. L'insieme dei nodi base $\mathcal{B}$ rappresenta l'insieme dei nodi che possono essere radici di un'arborescenza

3. Sia l'insieme $\mathcal{E} = \{1 \dots E\}$ l'insieme dei nodi esterni, ogni nodo $e \in \mathcal{E}$ è detto nodo esterno

4. L'insieme $\mathcal{B}$ è a sua volta partizionato in B insiemi disgiunti di nodi $\mathcal{B} = \bigcup_{i=1}^{B} \mathcal{R}_i$, $\mathcal{R}_i = \{1 \dots R_i\}$. Ogni insieme $\mathcal{R}_i$ è detto regione base

5. Ad ogni nodo $e \in \mathcal{N}$ è associato un peso $w_e \in \mathbb{R}$

6. Sia k il numero di arborescenze da ricercare

Variabili:

1. Per ogni regione base $i \in \mathcal{B}$, per ogni nodo base $b \in \mathcal{R}_i$ è presente una variabile binaria z_b che indica se il nodo base b della regione base $\mathcal{R}_i$ è radice di una delle k arborescenze

2. Per ogni nodo $e \in \mathcal{N}$, per ogni regione base $i \in \mathcal{B}$, per ogni nodo base $b \in \mathcal{R}_i$ è presente una variabile binaria x_{eb} che indica se il nodo e appartenente all'arborescenza radicata nel nodo base b della regione base $\mathcal{R}_i$ è in soluzione

3. Per ogni arco $j \in \mathcal{A}$, per ogni regione base $i \in \mathcal{B}$, per ogni nodo base $b \in \mathcal{R}_i$ è presente una variabile binaria y_{jb} che indica se l'arco j appartenente all'arborescenza radicata nel nodo base b della regione base $\mathcal{R}_i$ è in soluzione

Vincoli:

1. La foresta deve essere costituita da esattamente K arborescenze

2. Ogni nodo base $b \in \mathcal{B}$ deve essere presente in soluzione e deve far parte di esattamente un'arborescenza. Se un base $b \in \mathcal{R}_i$ non è radice di un'arborescenza, deve far parte di un'altra arborescenza radicata nella stessa regione base $\mathcal{R}_i$. Un nodo base $b \in \mathcal{R}_i$ non radice può quindi essere raggiunto solamente da archi $j \in \mathcal{A}$ uscenti da un altro nodo base $c \in \mathcal{R}_i, c \neq b$.

3. Ogni nodo esterno $e \in \mathcal{E}$ può appartenere al più ad una sola arborescenza

4. Per ogni nodo $e \in \mathcal{N}$ in soluzione ($x_{eb} = 1$), deve essere presente in soluzione esattamente un arco $j \in \mathcal{A}$ ($y_{jb} = 1$) tra l'insieme degli archi entranti nel nodo e

5. L'insieme dei nodi in soluzione ($x_{eb} = 1$), e l'insieme degli archi in soluzione ($y_{jb} = 1$) devono formare un digrafo connesso per ognuna delle k arborescenze

6. La somma dei pesi w_e associati ai nodi in soluzione deve valere almeno 0 in ognuna delle K arborescenze

Obiettivo:

1. Minimizzare il numero di nodi che forma la foresta: $\sum_{e=1}^{E} \sum_{i=1}^{B} \sum_{b=1}^{R_i} x_{eb}$

Modello:

$$
\begin{array}{lll}
min & \sum_{e=1}^{E} \sum_{i=1}^{B} \sum_{b=1}^{R_i} x_{eb} & (3.1) \\
s.t. & & \\
& \sum_{i=1}^{B} \sum_{b=1}^{R_i} z_b = K & (3.2) \\
& \sum_{j \in \delta^-(e)} y_{jb} = x_{eb} & \forall i \in \mathcal{B}, \forall b \in \mathcal{R}_i, \forall e \in \mathcal{E} \cup \mathcal{R}_i \quad (3.3) \\
& \sum_{j \in \delta^-(S)} y_{jb} \geq x_{eb} & \forall i \in \mathcal{B}, \forall b \in \mathcal{R}_i, \forall e \in \mathcal{E} \cup \mathcal{R}_i, \\
& & \forall S \subseteq \mathcal{E} \cup \mathcal{R}_i, \forall e \in S \quad (3.4) \\
& \sum_{j \in \delta^-(b) \cap \mathcal{R}_i} y_{jb} = 1 - z_b & \forall i \in \mathcal{B}, \forall b \in \mathcal{R}_i \quad (3.5) \\
& \sum_{j \in \delta^-(S) \cap \mathcal{R}_i} y_{jb} \geq 1 - z_b & \forall i \in \mathcal{B}, \forall b \in \mathcal{R}_i, \forall S \subseteq \mathcal{R}_i, \forall b \in S \quad (3.6) \\
& \sum_{i=1}^{B} \sum_{b=1}^{R_i} x_{eb} \leq 1 & \forall e \in \mathcal{E} \quad (3.7) \\
& \sum_{e=1}^{N} w_e x_{eb} \geq -w_b z_b & \forall i \in \mathcal{B}, \forall b \in \mathcal{R}_i \quad (3.8) \\
& z_b + \sum_{f=1}^{R_i} x_{bf} = 1 & \forall i \in \mathcal{B}, \forall b \in \mathcal{R}_i \quad (3.9) \\
& x_{eb} \in \{0, 1\} & \forall e \in \mathcal{N}, \forall i \in \mathcal{B}, \forall b \in \mathcal{R}_i \quad (3.10) \\
& y_{jb} \in \{0, 1\} & \forall j \in \mathcal{A}, \forall i \in \mathcal{B}, \forall b \in \mathcal{R}_i \quad (3.11) \\
& z_b \in \{0, 1\} & \forall i \in \mathcal{B}, \forall b \in \mathcal{R}_i \quad (3.12)
\end{array}
$$

L'obiettivo (3.1) è la minimizzazione del numero di nodi esterni che formano la foresta, non è necessario includere i nodi base dato che sono vincolati a fare parte della soluzione e pertanto costituiscono una costante.

Il vincolo (3.2) obbliga la costruzione di una foresta con esattamente k arborescenze.

I vincoli (3.3) indicano che un nodo esterno fa parte della soluzione solo se raggiunto da esattamente un arco entrante in una delle k arborescenze.

I vincoli (3.4) sono i vincoli di connettività sui nodi esterni in ognuna delle k arborescenze.

I vincoli (3.5) impongo ad un nodo base $b \in \mathcal{R}_i$ non radice di essere raggiunto da esattamente un arco entrante, l'arco entrante deve appartenere alla regione base $\mathcal{R}_i$. Per un nodo base radice, questo vincolo impone che non esista nessun arco entrante.

I vincoli (3.6) sono i vincoli di connettività all'interno delle regioni base.

I vincoli (3.7) obbligano un nodo esterno ad appartenere al più ad un'arborescenza.

I vincoli (3.8) sono i vincoli di soglia per ogni arborescenza.

I vincoli (3.9) sono necessari per la corretta somma sui pesi di ogni arborescenza.

Infine vincoli (3.10), (3.11) e (3.12) impongono le condizioni di integralità su tutte le variabili.

3.2 Riformulazione secondo Dantzig - Wolfe

La regione di ammissibilità per *MKFTC* è l'intersezione di K insiemi indipendenti Ω^b che soddisfano i vincoli: (3.3), (3.4), (3.5), (3.6) e (3.8). Solo i vincoli (3.2), (3.7) e (3.9) pongono in relazione diversi insiemi di variabili, per la loro proprietà di determinare l'aggregazione dei K sottoproblemi descritti: essi vengono detti vincoli di unione. Dantzig e Wolfe [26] propongono un metodo generale per riformulare problemi di programmazione lineare in modo da ottenerne la disaggregazione in sottoproblemi distinti.

Sia $\mathbf{x}^b = (x_{1b} \dots x_{Nb})$ un vettore per le variabili riferite ai nodi $i \in \mathcal{N}$, sia $\mathbf{y}^b = (y_{1b} \dots y_{Ab})$ un vettore per le variabili riferite agli archi $j \in \mathcal{A}$, sia $\mathbf{p^i} = (0 \dots 0, \underbrace{1 \dots 1}_{\mathcal{R}_i}, 0 \dots 0)$ un vettore $\forall i \in \mathcal{B}$ composto da 1 in corrispondenza dei nodi $b \in \mathcal{R}_i$ e 0 altrove, sia z^b la variabile indicante la radice b della regione base $\mathcal{R}_i$ e sia $\Omega^b = \{(x^b, y^b, z^b) : (3.3), (3.4), (3.5), (3.6), (3.8)\} \forall i \in \mathcal{B}, \forall b \in \mathcal{R}_i$.

L'obiettivo è scegliere k punti (k arborescenze), appartenenti a Ω^b diversi (diverso nodo base), in modo da minimizzare il numero totale di nodi che formano la foresta.

Dalla formulazione iniziale deriviamo il seguente modello:

$$
\begin{array}{lll}
min & \sum_{i\in\mathcal{B}}\sum_{b\in\mathcal{R}_i} 1^T\mathbf{x}^b & (3.13)\\
s.t. & & \\
& \sum_{i\in\mathcal{B}}\sum_{b\in\mathcal{R}_i} z^b = k & (3.14)\\
& \sum_{i\in\mathcal{B}}\sum_{b\in\mathcal{R}_i} \mathbf{x}^b \leq \mathbf{1} & (3.15)\\
& z^b + (\mathbf{p^i})^T\mathbf{x}^b = \mathbf{1} \quad \forall i \in \mathcal{B}, \forall b \in \mathcal{R}_i & (3.16)\\
& (\mathbf{x}^b, \mathbf{y}^b, z^b) \in \Omega^b \quad \forall i \in \mathcal{B}, \forall b \in \mathcal{R}_i & (3.17)\\
& \mathbf{x}^b \in \{\mathbf{0}, \mathbf{1}\} \quad \forall i \in \mathcal{B}, \forall b \in \mathcal{R}_i & (3.18)\\
& \mathbf{y}^b \in \{\mathbf{0}, \mathbf{1}\} \quad \forall i \in \mathcal{B}, \forall b \in \mathcal{R}_i & (3.19)\\
& z_b \in \{0, 1\} \quad \forall i \in \mathcal{B}, \forall b \in \mathcal{R}_i & (3.20)
\end{array}
$$

Rimpiazzando Ω^b con $conv(\Omega^b)$ e rilassando le condizioni di integrità sulle variabili otteniamo il rilassamento sottostante:

$$
\begin{array}{lll}
min & \sum_{i\in\mathcal{B}}\sum_{b\in\mathcal{R}_i} 1^T\mathbf{x}^b & (3.21)\\
s.t. & & \\
& \sum_{i\in\mathcal{B}}\sum_{b\in\mathcal{R}_i} z^b = k & (3.22)\\
& \sum_{i\in\mathcal{B}}\sum_{b\in\mathcal{R}_i} \mathbf{x}^b \leq \mathbf{1} & (3.23)\\
& z^b + (\mathbf{p^i})^T\mathbf{x}^b = \mathbf{1} \quad \forall i \in \mathcal{B}, \forall b \in \mathcal{R}_i & (3.24)\\
& (\mathbf{x}^b, \mathbf{y}^b, z^b) \in conv(\Omega^b) \quad \forall i \in \mathcal{B}, \forall b \in \mathcal{R}_i & (3.25)\\
& \mathbf{0} \leq \mathbf{x}^b \leq \mathbf{1} \quad \forall i \in \mathcal{B}, \forall b \in \mathcal{R}_i & (3.26)\\
& \mathbf{0} \leq \mathbf{y}^b \leq \mathbf{1} \quad \forall i \in \mathcal{B}, \forall b \in \mathcal{R}_i & (3.27)\\
& 0 \leq z_b \leq 1 \quad \forall i \in \mathcal{B}, \forall b \in \mathcal{R}_i & (3.28)
\end{array}
$$

Definendo con Θ^b l'insieme dei punti estremi di $conv(\Omega^b)$, ottenuta introducendo le variabili continue $\lambda_l \geq 0$ abbiamo che:

$$(\mathbf{x}^b, \mathbf{y}^b, z^b) = \sum_{l\in\Theta^b}(\mathbf{x}^b, \mathbf{y}^b, z^b)^l\lambda_l,\ \sum_{l\in\Theta^b}\lambda_l = 1$$

Sostituendo in (3.21), (3.22), (3.23) e (3.24) ed aggiungendo i vincoli di convessità otteniamo il *master problem*:

$$
\begin{array}{lll}
min & \sum_{i\in\mathcal{B}}\sum_{b\in\mathcal{R}_i}\sum_{l\in\Theta^b} 1^T(\mathbf{x}^b)^l\lambda_l & (3.29)\\
s.t. & & \\
& \sum_{i\in\mathcal{B}}\sum_{b\in\mathcal{R}_i}\sum_{l\in\Theta^b}(z^b)^l\lambda_l = k & (3.30)\\
& \sum_{i\in\mathcal{B}}\sum_{b\in\mathcal{R}_i}\sum_{l\in\Theta^b}(\mathbf{x}^b)^l\lambda_l \leq \mathbf{1} & (3.31)\\
& \sum_{l\in\Theta^b}(z^b + (\mathbf{p^i})^T\mathbf{x}^b)^l\lambda_l = \mathbf{1} \quad \forall i\in\mathcal{B},\forall b\in\mathcal{R}_i & (3.32)\\
& (\mathbf{x}^b,\mathbf{y}^b,z^b) = \sum_{l\in\Theta^b}(\mathbf{x}^b,\mathbf{y}^b,z^b)^l\lambda_l \quad \forall i\in\mathcal{B},\forall b\in\mathcal{R}_i & (3.33)\\
& \lambda_l \geq 0 \quad \forall l\in\Theta^b & (3.34)\\
& \sum_{l\in\Theta^b}\lambda_l \leq 1 \quad \forall i\in\mathcal{B},\forall b\in\mathcal{R}_i & (3.35)
\end{array}
$$

Ogni colonna della matrice dei vincoli rappresenta quindi un'arborescenza ammissibile. La funzione obiettivo (3.29) minimizza il numero di nodi che formano la foresta. Il vincolo (3.30) impone che la foresta sia formata da esattamente k arborescenze.

I vincoli (3.31) impongono che un nodo esterno appartenga al più ad un'arborescenza. I vincoli (3.32) impongono la presenza in soluzione di tutte le celle base. I vincoli (3.33) sono i vincoli di collegamento. I vincoli (3.34) e (3.35) sono i vincoli di convessità.

3.2.1 Il rilassamento del *master problem*

Introducendo il termine $c_l = 1^T(\mathbf{x}^b)^l$, il rilassamento lineare del *master problem* diventa il seguente:

$$
\begin{array}{lll}
min & \sum_{i\in\mathcal{B}}\sum_{b\in\mathcal{R}_i}\sum_{l\in\Theta^b} c_l\lambda_l & (3.36)\\
s.t. & & \\
& \sum_{i\in\mathcal{B}}\sum_{b\in\mathcal{R}_i}\sum_{l\in\Theta^b}(z^b)^l\lambda_l = k & (3.37)\\
& \sum_{i\in\mathcal{B}}\sum_{b\in\mathcal{R}_i}\sum_{l\in\Theta^b}(\mathbf{x}^b)^l\lambda_l \leq \mathbf{1} & (3.38)\\
& \sum_{l\in\Theta^b}(z^b + (\mathbf{p^i})^T\mathbf{x}^b)^l\lambda_l = \mathbf{1} \quad \forall i\in\mathcal{B},\forall b\in\mathcal{R}_i & (3.39)\\
& (\mathbf{x}^b,\mathbf{y}^b,z^b) = \sum_{l\in\Theta^b}(\mathbf{x}^b,\mathbf{y}^b,z^b)^l\lambda_l \quad \forall i\in\mathcal{B},\forall b\in\mathcal{R}_i & (3.40)\\
& \lambda_l \geq 0 \quad \forall l\in\Theta^b & (3.41)\\
& \sum_{l\in\Theta^b}\lambda_l \leq 1 \quad \forall i\in\mathcal{B},\forall b\in\mathcal{R}_i & (3.42)
\end{array}
$$

Purtroppo il numero di colonne è esponenziale rispetto alle dimensioni dell'istanza e rende inutilizzabile la risoluzione del rilassamento lineare del *master problem*.

3.2.2 Il *master problem* ridotto

Come appena accennato, risulta improponibile utilizzare il metodo del simplesso per risolvere il rilassamento lineare del *master problem*, in quanto ad ogni iterazione si renderebbe necessario valutare il costo ridotto di un numero esponenziale di colonne.

È possibile tuttavia risolvere il rilassamento lineare del *master problem* con la tecnica di *column generation*. Rimuovendo dal *master problem* un opportuno insieme di colonne si ottiene un *master problem ridotto* (*RMP*). Il *RMP* deve contenere almeno una soluzione di base ammissibile. Risolto il rilassamento lineare di *RMP* (*L-RMP*), si sfrutta l'informazione derivante dalle variabili duali per determinare delle colonne, non ancora inserite in *RMP*, che avendo costo ridotto negativo, sono candidate ad entrare in base. Per calcolare il costo ridotto delle colonne si risolve il *pricing problem*, che consiste in un problema di ottimizzazione per ogni cella base che può essere radice di un'arborescenza.

Dal *master problem*:

$$
\begin{array}{lll}
min & \sum_{j\in\mathcal{N}}\sum_{i\in\mathcal{B}}\sum_{b\in\mathcal{R}_i}\sum_{l\in\Theta^{ib}} x^l_{jb}\lambda_l & (3.43)\\
s.t. & & \\
& \sum_{i\in\mathcal{B}}\sum_{b\in\mathcal{R}_i}\sum_{l\in\Theta^b} z^l_b\lambda_l = k & (3.44)\\
& \sum_{i\in\mathcal{B}}\sum_{b\in\mathcal{R}_i}\sum_{l\in\Theta^b} x^l_{jb}\lambda_l \leq 1 \quad \forall j\in\mathcal{N} & (3.45)\\
& \sum_{f\in\mathcal{R}_i}\sum_{m\in\Theta^f}(x_{bf})^m\lambda_m + \sum_{l\in\Theta^b} z^l_b\lambda_l = 1 \quad \forall i\in\mathcal{B}, \forall b\in\mathcal{R}_i & (3.46)\\
& \lambda_l \geq 0 \quad \forall l\in\Theta^b & (3.47)\\
& \sum_{l\in\Theta^b}\lambda_l \leq 1 \quad \forall i\in\mathcal{B}, \forall b\in\mathcal{R}_i & (3.48)
\end{array}
$$

Sia ξ la variabile duale associata al vincolo (3.44), siano μ_j le variabili duali associate ai vincoli (3.45), siano γ_b le variabili duali associate ai vincoli (3.46) e siano η_b le variabili duali associate ai vincoli (3.48).

Il costo ridotto π_l di una colonna l che rappresenta un'arborescenza radicata in una cella base b di una regione base $\mathcal{R}_i$ è il seguente:

$$\pi_l = \sum_{j\in\mathcal{N}} x^l_{jb} - \sum_{j\in\mathcal{N}}\mu_j x^l_{jb} - \sum_{i\in\mathcal{B}:b\in\mathcal{R}_i}\sum_{j\in\mathcal{R}_i}\gamma_j x^l_{jb} - \gamma_b - \xi - \eta_b$$

La variabile ξ e le variabili γ_b sono libere dato che corrispondono a vincoli di uguaglianza, le variabili μ_j e η_b sono ≤ 0 perché corrispondo a vincoli di $\leq$. Il *pricing problem* da risolvere per ogni cella base $l \in \bigcup_{i=1}^{B}\mathcal{R}_i$ è quindi il seguente:

$$\begin{array}{ll} min & \pi_l = \sum_{j \in \mathcal{N}} x^l_{jb} - \sum_{j \in \mathcal{N}} \mu_j x^l_{jb} - \sum_{i \in \mathcal{B}: b \in \mathcal{R}_i} \sum_{j \in \mathcal{R}_i} \gamma_j x^l_{jb} - \gamma_b - \xi - \eta_b \\ s.t. & \\ & x^l_{jb} \in \Theta^b \end{array}$$

Il problema di *pricing* è risolto dall'algoritmo di branch & cut esposto nel capitolo 2. Risolto il problema di *pricing*, viene inserito nel *L-RMP* l'insieme delle colonne individuate, aventi costo ridotto negativo, per valutare nuovamente il valore ottimo del rilassamento lineare ed ottenere un nuovo insieme di variabili duali.

Iterativamente si procede ricercando colonne di costo ridotto negativo. È importante notare come la successione dei valori del rilassamento lineare *L-RMP* sia monotona decrescente e converga al valore del rilassamento lineare del *master problem*.

Nel caso in cui la soluzione esatta del sottoproblema di *pricing*, per ogni cella base, non corrisponda a nessuna colonna di costo ridotto negativo, significa che la soluzione di base corrente è ottima, e il suo valore costituisce un bound duale valido per il *master problem*.

3.3 Preprocessing

Il preprocessing mostrato nella sezione 3.3 non può essere applicato nel problema *MKFTC* perché i cammini minimi di diverse regioni base potrebbero sovrapporsi. Ad ogni sovrapposizione, la porzione di digrafo minima che contiene la soluzione ottima si allarga, in quanto deve garantire la presenza della soluzione ottima per ognuna delle regioni sovrapposte.

È possibile però eseguire altre riduzioni, sia sul digrafo, sia sull'insieme delle celle base che possono essere radici di un'arborescenza (nodi sui quali è necessario risolvere il problema di *pricing*).

Dalla formulazione del problema sappiamo che ogni cella base, che non costituisce una radice, deve essere raggiunta da un arco appartenente alla stessa regione base (vincoli (3.5) e (3.6)). È possibile quindi eliminare tutti gli archi del digrafo che da celle esterne entrano nelle celle base. Sempre dai vincoli (3.5) e (3.6), ogni volta che viene risolto il problema di *pricing* per un certo nodo base, è possibile rimuovere dal digrafo tutte le altre regioni base.

Nelle regioni base, che sono costituite da sottogriglie rettangolari, può essere limitato il numero delle possibili radici, riducendo quindi il numero di sottoproblemi di *pricing* da risolvere ad ogni attivazione del *pricing problem*.

Infatti, qualsiasi sia la soluzione ottima, le celle base interne (non hanno archi uscenti verso le celle base esterne) saranno sempre contenute in arbore-

scenze che hanno come radice una cella base di contorno (ha almeno un arco uscente verso una cella esterna).

Il sottoproblema di *pricing* sarà quindi risolto solo per le celle base di contorno.

All'interno di una regione base, esistono tante arborescenze costituite dallo stesso insieme di nodi che si differenziano solamente per il nodo base che funge da radice. Per limitare il numero di queste arborescenze, e conseguentemente velocizzare il processo di risoluzione, è possibile eliminare una parte (inutile) di archi della regione base senza perdere nessuna delle possibili combinazioni di celle base che possono formare un'arborescenza.

Tale insieme di archi è eliminato perché non aggiunge diverse combinazioni di celle base, ma può generare soluzioni aventi lo stesso insieme di nodi e quindi identiche ai fini della risoluzione. Purtroppo, questo tipo di riduzione diventa molto complessa al crescere delle dimensioni e sarebbe quindi necessario uno studio più approfondito per identificare l'insieme minimale di archi.

Nella figura 3.1 mostro la riduzione eseguita quando il numero di righe o il numero delle colonne della regione base è 1. Nella figura 3.2 mostro la riduzione eseguita per il caso 2x2. Nella figura 3.3 è rappresentata la riduzione eseguita per i casi 2x3 o 3x2. In figura 3.4 è mostrata la riduzione per il caso 3x3. Nella figura 3.5 è rappresentata la riduzione per i casi 4x2 o 2x4. Infine, nelle figure 3.6 e 3.7 mostro le riduzioni eseguite quando le dimensioni sono maggiori di quelle descritte.

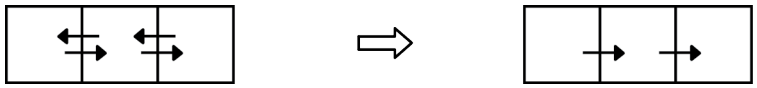

Figura 3.1: Riduzione quando il numero di righe o colonne è 1

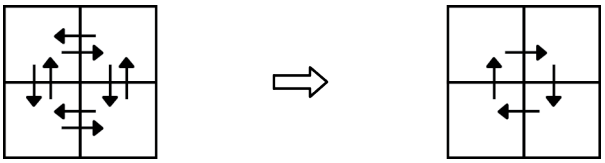

Figura 3.2: Riduzione per il caso 2x2

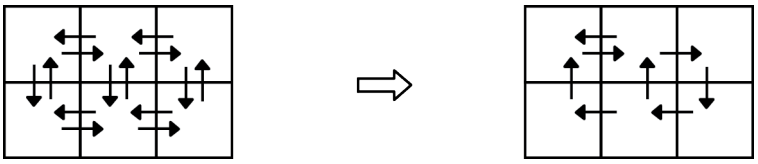

Figura 3.3: Riduzione per i casi 2x3 o 3x2

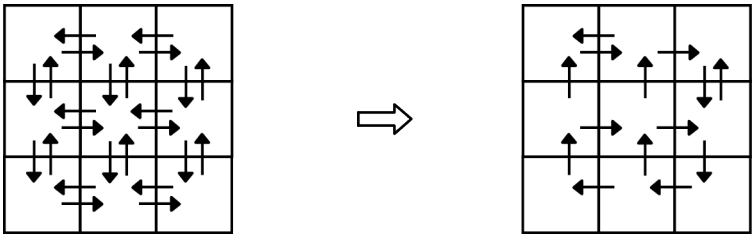

Figura 3.4: Riduzione per il caso 3x3

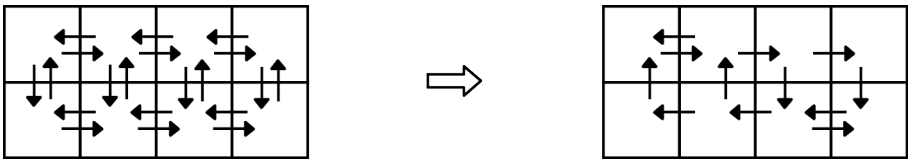

Figura 3.5: Riduzione per il caso 4x2

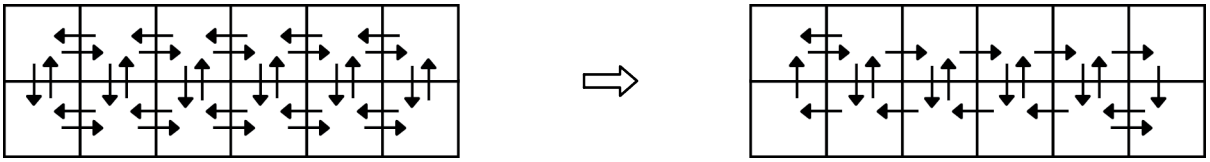

Figura 3.6: Riduzione per i casi 2xn o nx2

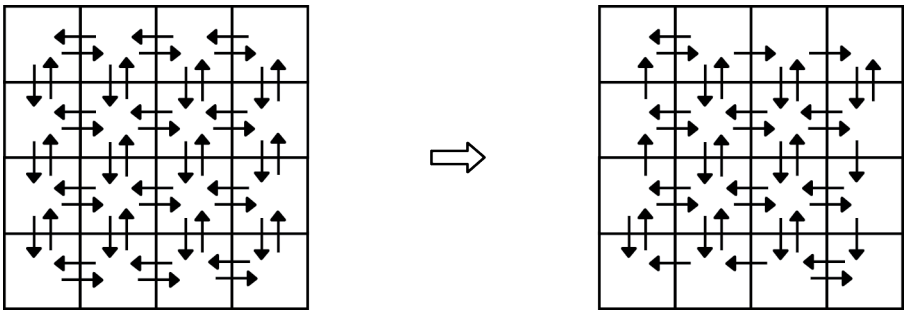

Figura 3.7: Riduzione per il caso mxn

3.4 Pricing per priorità

Per trovare colonne di costo ridotto negativo non è sempre necessario risolvere all'ottimo il *pricing problem*; per garantire la convergenza al valore ottimo del *L-RMP* ed ottenere quindi un *LB* valido è sufficiente rispettare la condizione di arresto del problema di *pricing*.

Idealmente, è necessario generare colonne ottime solamente nel caso in cui bisogna garantire la mancanza di colonne di costo ridotto negativo. È quindi possibile generare colonne tramite algoritmi euristici, finché tali colonne risultano essere vantaggiose.

Sfruttando questa possibilità eseguo un *pricing* per priorità, il cui obiettivo è generare colonne di costo ridotto negativo con il minimo sforzo computazionale.

Ad ogni attivazioni del problema di *pricing* valuto l'esistenza di colonne di costo ridotto negativo per ogni possibile cella base che può essere radice di un'arborescenza, la generazione delle colonne ad ogni attivazione del problema di *pricing* avviene nel seguente ordine:

1. generazione delle colonne in modo euristico mediante una strategia *greedy*[24,25], per ogni possibile cella base che può essere radice di un'arborescenza

2. generazione di una colonna in modo euristico mediante branch & cut troncato al nodo radice (dai risultati sperimentali del capitolo 2 ho mostrato come spesso, la soluzione ottima del problema *MATC*, è trovata nel nodo radice), per ogni possibile cella base che può essere radice di un'arborescenza

3. generazione di una colonna ottima mediante branch & cut, per ogni possibile cella base che può essere radice di un'arborescenza

La generazione di colonne da parte di algoritmi a maggiore priorità, ad ogni attivazione del problema di *pricing*, avviene solamente quando nessuna delle colonne generate dall'algoritmo con minore priorità ha associato un costo ridotto negativo.

Quando il problema di *pricing* è risolto in modo esatto e vengono trovate colonne di costo ridotto negativo, alla successiva iterazione non è necessario valutare la colonna associata ad ogni cella base per ogni possibile radice di un'arborescenza, ma è sufficiente valutare le radici delle ragioni in cui sono migliorati i valori delle variabili duali. Chiaramente, se migliora un valore duale che ha impatto su tutte le regioni base (ξ o γ_e), sarà necessario rivalutare in modo esatto tutte le possibili radici.

3.4.1 Pricing di tipo greedy

L'idea che sta alla base dell'algoritmo di *pricing* di tipo *greedy* è quella di espandere un'arborescenza non ammissibile inizialmente costituita dal solo nodo radice, fino ad ottenere un'arborescenza che soddisfa il vincolo di soglia.

Ad ogni passo, se è presente un nodo raggiungibile con associato un valore duale ≥ 1, tale nodo espande l'arborescenza.

Questo criterio è dettato dal fatto che nel problema di *pricing*, la funzione obiettivo tende a minimizzare il *trade off* tra il numero di nodi che forma l'arborescenza e i "premi" duali associati ai nodi.

Quando un nodo ha associato un valore duale pari a 1, il costo pagato per l'estensione dell'arborescenza è annullato; per valori duali > 1 risulta più conveniente allargare l'arborescenza.

Se ad un certo passo, nessun nodo nell'insieme dei nodi raggiungibili ha associato un valore duale ≥ 1 e contemporaneamente non è soddisfatto il vincolo di soglia, l'arborescenza è estesa con il nodo raggiungibile avente il maggior peso w_i.

Per l'estrazione efficiente del nodo raggiungibile con associato il massimo valore di w_i è utilizzato uno heap. Nello pseudocodice seguente, relativo all'algoritmo appena descritto, è passato come input un digrafo che, come vedremo in seguito, è necessario per motivi di branching.

Data: Digrafo aggiornato dalla strategia di branching $D = (\mathcal{Q}, \mathcal{W})$; pesi $w_i \; \forall i \in \mathcal{Q}$; valori delle variabili duali associate ai nodi d_i $w_i \; \forall i \in \mathcal{Q}$; valori delle variabili duali γ_b, ξ e η_b; nodo radice $r \in \mathcal{Q}$
Result: Una soluzione euristica s per $MATC$ con costo ridotto rc

```
for i ∈ Q do
|  e_i := TRUE;
end
e_r := FALSE;
p := w_r;
rc := 1 − d_r − γ_b − ξ − η_b;
coda := insert(i ∈ Q : ∃(r, i) ∈ W);
heap := ∅;
s := ∅ ∪ r;
while coda ≠ ∅ OR p < 0 do
    n := extract(coda);
    while coda ≠ ∅ do
        if d_n ≥ 1 then
            s := s ∪ n;
            e_n := FALSE;
            p := p + w_n;
            rc := rc - d_n + 1;
            for i ∈ Q : ∃(n, i) ∈ W do
                if e_i then
                    coda := insert(i);
                end
            end
        else
            heap := insert(n, w_n);
        end
    end
    if heap ≠ ∅ AND p < 0 then
        v := extract(heap);
        s := s ∪ v;
        e_v := FALSE;
        p := p + w_v;
        rc := rc - d_v + 1;
        for i ∈ Q : ∃(v, i) ∈ W do
            if e_i then
                coda := insert(i);
            end
        end
    end
end
```

Algorithm 4: L'algoritmo greedy per il problema di *pricing*

L'algoritmo consiste in una modifica dell'algoritmo di visita in ampiezza; ad ogni iterazione (nel caso pessimo) avviene sempre l'inserimento del nodo nello heap che ha complessità $O(log(n))$, dove n rappresenta il numero di elementi contenuti nello heap. L'algoritmo, quando non vi sono variabili fissate dal branching, ha pertanto complessità $O(Nlog(N) + A)$.

3.5 Strategia di branching

In ogni sottoproblema, quando la soluzione ottima del *L-RMP* fornisce un bound duale migliore del bound primale e presenta valori frazionari nelle variabili originali, è necessario eseguire il branching sulle variabili originali. Ho scelto di eseguire un branching dicotomico sulle variabili corrispondenti ai nodi.

La scelta della variabile originale sulla quale eseguire il branching avviene secondo un criterio basato su priorità. L'ordine di priorità che ho utilizzato è dettato dalla difficoltà delle decisioni da prendere. La valutazione delle variabili originali su cui eseguire il branching avviene nel seguente ordine:

1. $z_b \; \forall i \in \mathcal{B}, \forall b \in \mathcal{R}_i$. Nella regione base $\mathcal{R}_i$ il nodo b è una radice
2. $x_{eb} \; \forall i \in \mathcal{B}, \forall b \in \mathcal{R}_i, \forall e \in \mathcal{R}_i$. Il nodo e, è un nodo base della regione $\mathcal{R}_i$ assegnato alla radice b
3. $x_{eb} \; \forall e \in \mathcal{E}, \forall i \in \mathcal{B}, \forall b \in \mathcal{R}_i$. Il nodo e, è un nodo esterno assegnato alla radice b della regione $\mathcal{R}_i$

Ispezionando le variabili a partire dall'insieme a priorità più alta (z_b), viene scelto di eseguire il branching sulla variabile frazionaria a maggiore priorità avente valore più prossimo a $\frac{1}{2}$. Il valore delle variabili originali è ottenuto sommando sulle colonne inserite, moltiplicate per il rispettivo λ_l.

L'ispezione nell'insieme delle variabili a priorità minore ($x_{eb} \; \forall i \in \mathcal{B}, \forall b \in \mathcal{R}_i, \forall e \in \mathcal{R}_i$ e $x_{eb} \; \forall e \in \mathcal{E}, \forall i \in \mathcal{B}, \forall b \in \mathcal{R}_i$) avviene unicamente quando l'insieme di variabili a priorità maggiore assume solamente valori interi.

Quando una variabile z_b è scelta dalla strategia di branching, vengono creati due sottoproblemi ponendo esplicitamente $z_b = 0$ nel primo e $z_b = 1$ nel secondo.

Nel sottoproblema dove $z_b = 1$, vengono poste a 0 le variabili $x_{ec} \; \forall e \in \mathcal{R}_i, \forall c \in \mathcal{R}_i : e = b, c \neq b$ (se un certo nodo base b della regione base $\mathcal{R}_i$ è vincolato ad essere radice, non può far parte di nessun altra arborescenza).

Quando una variabile $x_{eb} : e \in \mathcal{R}_i$ è scelta dalla strategia di branching creo due sottoproblemi: nel primo pongo $x_{eb} = 0$, nel secondo, invece di porre

esplicitamente $x_{eb} = 1$, pongo a 0 le variabili $x_{ec} : c \in \mathcal{R}_i, c \neq b$ (assegnare un certo nodo base e ad una possibile radice b della regione $\mathcal{R}_i$ è equivalente ad impedire a tutte le altre possibili radici $c \in \mathcal{R}_i : c \neq b$ di avere il nodo base e nella loro arborescenza).

In modo del tutto analogo, quando una variabile $x_{eb} : e \in \mathcal{E}$ è scelta dalla strategia di branching creo due sottoproblemi: nel primo pongo $x_{eb} = 0$, nel secondo, invece di porre esplicitamente $x_{eb} = 1$, pongo a 0 le variabili x_{ec} $\forall t \in \mathcal{B}, \forall c \in \mathcal{R}_t : c \neq b$ (assegnare un certo nodo esterno e ad una possibile radice b della regione $\mathcal{R}_i$ è equivalente ad impedire a tutte le altre possibili radici di avere il nodo esterno e nella loro arborescenza).

Ho scelto di non utilizzare il branching a 1 per le variabili x_{eb}, corrispondenti ai nodi base ed ai nodi esterni, al fine di semplificare l'algoritmo *greedy* e il possibile utilizzo di altre euristiche. In questo modo non vengono gestite variabili vincolate ad essere in soluzione. Il branching a 0 per le variabili x_{eb}, corrispondenti ai nodi base ed ai nodi esterni, è ottenuto eliminando tali nodi e gli archi coinvolti dal digrafo.

Il branching a 0 per le variabili z_b è invece ottenuto evitando di eseguire il problema di *pricing* che ha come radice il nodo base z_b.

3.6 Inizializzazione e gestione delle colonne

Per dare inizio al processo di *column generation* è necessario che nel *L-RMP* vi sia un insieme di colonne che costituiscano una soluzione di base ammissibile; questo problema può essere risolto inserendo nel *L-RMP* un opportuno insieme di colonne *dummy*. Ho utilizzato quindi una colonna *dummy* che soddisfa tutti i vincoli del *L-RMP* con associato un costo pari al numero di nodi presenti nel digrafo +1. Nel caso peggiore infatti, la soluzione ottima sarà l'intero insieme di nodi $\mathcal{N}$ del digrafo.

Durante l'inizializzazione vengono inserite nel *L-RMP* anche le colonne corrispondenti ai cosidetti *pattern vuoti*, è quindi inserita una colonna vuota per ogni possibile radice di un'arborescenza.

Inserendo di volta in volta nuove colonne, il numero di variabili nel *L-RMP* può diventare troppo elevato per poter essere gestibile efficientemente da un *LP solver*. Per contro, avere a disposizione un insieme di arborescenze "molto buone" prima di dare inizio alla generazione di nuove colonne, può accelerare la convergenza all'ottimo del rilassamento lineare, riducendo il numero di iterazioni di *column generation*.

In generale, politiche che prevedano un compromesso tra i due aspetti possono fornire le prestazioni migliori, anche se il corretto bilanciamento dei

tempi di calcolo dipende dell'efficienza relativa del *solver* utilizzato per la soluzione del rilassamento lineare e dell'algoritmo per il *pricing problem*.

SCIP, il framework utilizzato per l'implementazione dell'algoritmo di branch & price, applica automaticamente politiche molto efficienti basate sull'analisi del costo ridotto di ogni colonna in modo da mantenere un insieme di colonne di una certa qualità nel *L-RMP*.

In combinazione con le operazioni di branching, è necessario eliminare nei sottoproblemi le colonne non più ammissibili a causa dell'incompatibilità introdotta dai nuovi limiti sulle variabili.

Quando la scelta di branching impone che un certo nodo base b della regione base $\mathcal{R}_i$ non sia radice di un'arborescenza ($z_b = 0$), è necessario rimuovere dal sottoproblema tutte le colonne che hanno come radice il nodo b della regione $\mathcal{R}_i$. Nei sottoproblemi dove $z_b = 1$, viene rimosso il corrispondente *pattern vuoto* e vengono eliminate dal sottoproblema tutte le colonne nelle quali il nodo base b della regione base $\mathcal{R}_i$ non compare come radice.

In modo analogo, quando la scelta di branching impone che un certo nodo base e della regione base $\mathcal{R}_i$ non possa far parte di un'arborescenza con radice b ($x_{eb} = 0$), vengono eliminate dal sottoproblema tutte le colonne in cui il nodo e fa parte di un'arborescenza con radice b. Lo stesso discorso vale anche per i nodi esterni: quando la scelta di branching impone che un certo nodo esterno c della regione base $\mathcal{R}_i$ non possa far parte di un'arborescenza con radice b ($x_{cb} = 0$), vengono eliminate dal sottoproblema tutte le colonne in cui il nodo c fa parte di un'arborescenza con radice b.

3.7 Generazione delle istanze

Le istanze del problema *MKFTC* utilizzate per valutare le prestazioni dell'algoritmo di branch & price sono generate nel modo seguente:

- La griglia è quadrata, e viene popolata con valori di probabilità (somma pari a 1) generati da gaussiane bivariate; la cella della griglia che costituisce il centro della gaussiana è scelto in modo casuale uniforme
- Cercando di replicare le istanze reali, la generazione dei valori di probabilità è arrestata quando almeno il 70% delle celle ha associata una probabilità > 0
- Il numero e la dimensione delle regioni base è parametrizzato
- Una volta arrestata la generazione dei valori di probabilità ρ_i, viene scelta in modo casuale la posizione di ogni regione base. La posizione

è scelta in modo che, ad ogni cella base della regione base, corrisponda un valore di probabilità > 0. Le regioni base non possono sovrapporsi

- I valori di probabilità sono convertiti in pesi w_i mediante la relazione $w_i = \rho_i(\tau - s_i)$.

3.8 Risultati sperimentali

Nel capitolo 2 ho mostrato come la soluzione esatta di *MATC* sia più complessa quando ai nodi base vengono associati i pesi $w_i \in \mathbb{R}$. Nella risoluzione dei sottoproblemi di *pricing*, la situazione è ulteriormente complicata dai possibili valori negativi delle variabili duali associate a tutti i nodi.

Come conseguenza, i tempi richiesti dall'algoritmo di branch & price crescono molto rapidamente sia all'aumentare delle dimensioni di input, sia all'aumentare del numero delle celle di contorno.

Nelle tabelle 3.1, 3.2, 3.3 e 3.4 mostro la tendenza dei tempi di calcolo e del numero di nodi che formano la foresta al variare dei parametri. I dati sono il risultato dell'aggregazione dei risultati sperimentali (più esaurienti) che sono posti alla fine del capitolo nelle tabelle 3.5, 3.6, 3.7, 3.8, 3.9, 3.10, 3.11 e 3.12. Tutti i dati sono ottenuti eseguendo l'algoritmo di branch & price su un unico scenario corrispondente ad una griglia 15x15.

Nello scenario rimangono costanti le probabilità ρ_i associate ai nodi e le posizioni delle regioni base, mentre vengono fatti variare i parametri relativi ai valori di soglia τ, al numero di arborescenze k e al numero di regioni base.

Il significato dei campi delle tabelle è il seguente:

1. Aggregazione: è il parametro sul quale vengono aggregati i dati

2. Tempo: è il tempo medio, espresso in secondi, necessario per ottenere la soluzione ottima fermata al nodo radice per il livello di aggregazione considerato

3. TempoB&CT: è il tempo medio, espresso in secondi, necessario per ottenere una soluzione euristica fermata al nodo radice per il livello di aggregazione considerato. La risoluzione in modo euristico prevede l'utilizzo dell'algoritmo *greedy* e dell'algoritmo di branch & cut troncato al nodo radice, per risolvere i sottoproblemi di *pricing*.

4. NCO: è il numero di celle medio che costituisce la foresta per il livello di aggregazione considerato. Sono aggregati i valori delle soluzioni ottime.

5. NCE: è il numero di celle medio che costituisce la foresta per il livello di aggregazione considerato. Sono aggregati i valori delle soluzioni euristiche.

Nella tabella 3.1 sono mostrati i risultati aggregati per livello di soglia.

Aggregazione	**Tempo**	**TempoB&CT**	**NCO**	**NCE**
Livello di soglia 0.05	3444.35	61.43	27.14	27.29
Livello di soglia 0.10	2596.50	38.36	19.57	19.64
Livello di soglia 0.20	1637.79	14.64	15.36	15.36
Livello di soglia 0.40	754.07	12.31	12.71	12.71

Tabella 3.1: Aggregazione per livelli di soglia

All'aumentare del valore di soglia i tempi tendono a diminuire, ciò perché le arborescenze necessitano di "pochi" nodi esterni per soddisfare il vincolo di soglia. Come conseguenza le sovrapposizioni tra arborescenze diventano meno probabili e la soluzione richiede generalmente un tempo minore.

Come ci si aspettava, il numero di nodi (celle) che formano la foresta diminuisce all'aumentare del valore di soglia. Per valori di soglia alti (0.20 e 0.40) la soluzione euristica tende ad essere anche ottima più frequentemente.

La tabella 3.2 mostra i risultati aggregati per numero di regioni base; aumentando il numero di regioni base cresce il numero delle celle di contorno. Dall'analisi appare evidente come l'algoritmo di branch & price risulti sensibile al numero di celle di contorno; peggiorano maggiormente le prestazioni dell'algoritmo di branch & price con garanzia di ottimalità rispetto alle prestazioni dell'algoritmo branch & price utilizzato in modo euristico.

Aggregazione	**Tempo**	**TempoB&CT**	**NCO**	**NCE**
Una regione base 2x2	426.75	13.25	12.31	12.38
Due regioni base 2x2	2517.72	21.05	24.94	24.94

Tabella 3.2: Aggregazione per numero regioni base

La tabella 3.3 mostra la complessità introdotta dall'aumento della dimensione della regione base.

Aggregazione	**Tempo**	**TempoB&CT**	**NCO**	**NCE**
Una regione base 2x2	424.90	13.25	12.31	12.38
Una regioni base 3x3	1746.44	68.31	19.75	19.88

Tabella 3.3: Aggregazione per la dimensione delle regioni base

La tabella 3.4 mostra i risultati aggregati per classi "simili" di k. Al crescere della dimensione delle regioni base, per valori di k intermedi tra il numero delle regioni base e il numero delle celle di contorno, il numero di possibili combinazioni delle celle base che formano un'arborescenza è maggiore; pertanto peggiorano le prestazioni dell'algoritmo. Anche le soluzioni euristiche tendono ad essere più lontane dall'ottimo per valori di k intermedi.

Un elevato numero di combinazioni di celle base che formano un'arborescenza, può causare un aumento considerevole delle colonne di costo ridotto negativo.

La presenza di "tante" colonne di costo ridotto negativo richiede di attivare un maggior numero di volte il problema di *pricing* con un conseguente pesante aumento dei tempi di calcolo.

Un'altra conseguenza negativa derivante dall'elevato numero di colonne di costo ridotto negativo è il generale aumento delle operazioni di branching necessarie per raggiungere delle soluzioni ammissibili. Infatti, è in genere più difficile ricavare soluzioni intere quando "tante" colonne sono in soluzione con piccole frazioni.

Quando k assume valori estremi, i tempi di calcolo sono generalmente più brevi.

Aggregazione	**Tempo**	**TempoB&CT**	**NCO**	**NCE**
k = numero regioni base	2233.50	13.63	14.13	14.13
$k \approx$ numero regioni base	2269.78	60.11	14.58	14.58
$k \approx$ mediano	2939.94	63.13	20.56	21.43
$k \approx$ numero celle contorno	2212.65	58.31	21.50	21.61
k = numero celle contorno	1361.08	22.50	23.08	23.17

Tabella 3.4: Aggregazione per k

Facendo riferimento ai risultati delle tabelle 3.5, 3.6, 3.7, 3.8, 3.9, 3.10, 3.11 e 3.12 esposte a fine capitolo, si nota che l'utilizzo dell'algoritmo di branch & price che utilizza il solo algoritmo *greedy* per risolvere i sottoproblemi di *pricing*, risulta essere estremamente rapido. Di contro, tale algoritmo trova delle soluzioni quasi esclusivamente per i casi più "semplici" derivanti da elevati valori del parametro di soglia (0.20 e 0.40).

Quando i sottoproblemi di *pricing* vengono risolti con l'algoritmo *greedy* e con l'algoritmo di branch & cut troncato al nodo radice, l'algoritmo branch & price fornisce il miglior *trade off* tra bontà della soluzione e il tempo di calcolo. Infatti, nella maggior parte dei casi sono stati ottenuti dei risultati ottimi.

Come era possibile immaginare, il valore ottimo della funzione obiettivo si ottiene, in generale, per valori di k uguali o prossimi al numero delle celle

di contorno. In casi particolari, che possono nascere dalla vicinanza delle regioni base o dalla presenza dei pesi w_i positivi in regioni ristrette, il valore ottimo della funzione obiettivo è raggiunto con un valore di k minore rispetto al numero delle celle di contorno. Spesso tale valore è il massimo valore di k per il quale viene trovata una soluzione ammissibile.

Un caso dove il valore ottimo della funzione obiettivo non coincide con un valore di k pari al numero delle celle di contorno è rappresentato dalle figure 3.8 e 3.9.

Nelle figure 3.8 e 3.9 sono presenti due regioni base (sfondo rosso), il valore intero associato ad ogni nodo è il suo identificativo, il valore frazionario sottostante è invece il peso w_i associato al nodo. Ogni arborescenza è rappresentata con un contorno di uguale colore; la radice di un'arborescenza ha un contorno più marcato.

La soluzione con $k = 5$ utilizza un totale di 16 nodi ed ha quindi un valore di funzione obiettivo pari a $\frac{16}{5} = 3.2$, la soluzione con $k = 6$ utilizza invece 22 nodi ed ha quindi una funzione obiettivo di $\frac{22}{6} \approx 3.67$.

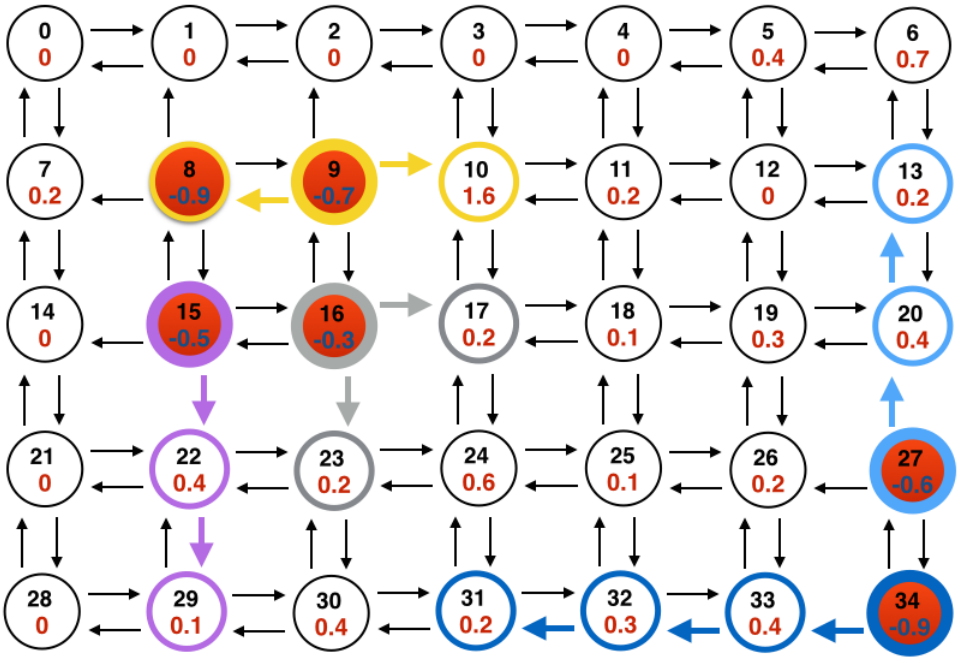

Figura 3.8: Soluzione ottima con $k = 5$

La tendenza del miglior valore della funzione obiettivo di essere in prossimità del numero delle celle di contorno, permette, nella pratica, di evitare la risoluzione per tutti i valori di k.

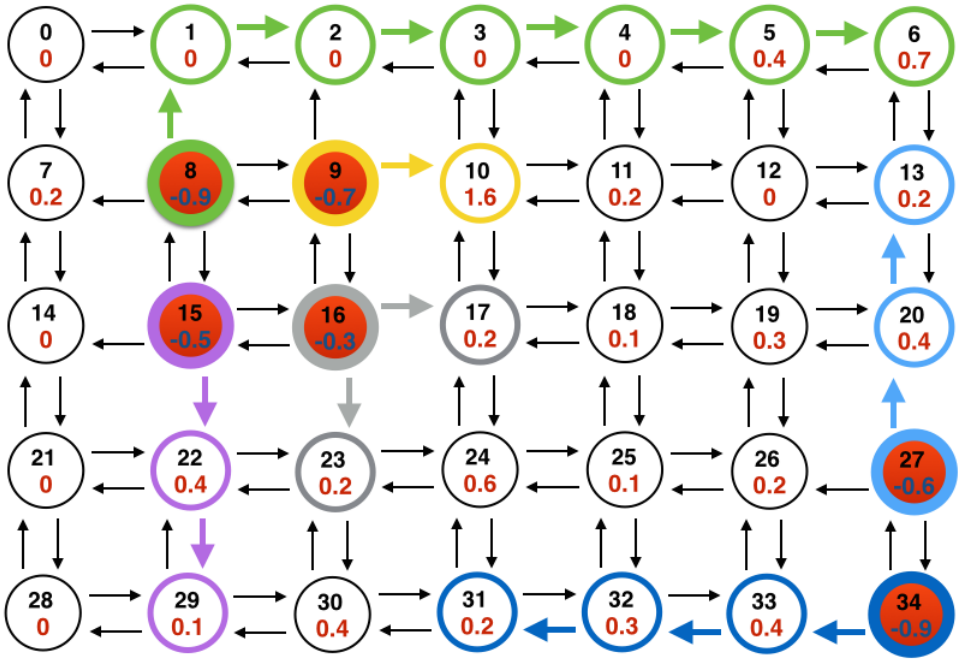

Figura 3.9: Soluzione ottima con $k = 6$

Partendo da un valore di k pari al numero di celle di contorno, il modo migliore per trovare una soluzione (probabilmente ottima), è quello di eseguire l'algoritmo branch & price decrementando iterativamente k fino a raggiungere una soluzione ammissibile.

In seguito, nelle tabelle 3.5, 3.6, 3.7, 3.8, 3.9, 3.10, 3.11 e 3.12 sono mostrati i risultati sperimentali in forma tabellare. Il significato dei vari campi delle tabelle è il seguente:

1. Istanza: è il nome dell'istanza; la coppia di interi separati dal carattere "x" indica la dimensione della griglia, il valore numerico finale rappresenta la soglia τ

2. TS: è il tipo di soluzione; O sta per soluzione ottima, in questo caso, il problema di *pricing* viene risolto in modo esatto quando necessario.

   B&CT è la soluzione ottenuta eseguendo *pricing* solamente con l'algoritmo *greedy* e con l'algoritmo di branch & cut troncato al nodo radice. G è la soluzione ottenuta eseguendo *pricing* con il solo algoritmo *greedy*.

3. K: è il parametro che definisce il numero di arborescenze da ricercare nel problema

4. TR: è il tempo, espresso in secondi, speso nel nodo radice dell'algoritmo di branch & price
5. LBR: è il valore di LB trovato nel nodo radice dell'algoritmo di branch & price
6. UBR: è il valore di UB trovato nel nodo radice dell'algoritmo di branch & price
7. BUB: è il miglior valore di UB trovato dall'algoritmo. La soluzione è ottima solamente in corrispondenza del tipo di soluzione "O".
8. OBJ: è il valore della funzione obiettivo: $\frac{UB}{K}$.
9. OS: è il valore della la soluzione ottima del problema.
10. TP: è il tempo, espresso in secondi, speso per eseguire *pricing*
11. T: è il tempo, espresso in secondi, necessario all'algoritmo di branch & price per risolvere il problema $MKFTC$ in maniera esatta o in maniera euristica a seconda del valore del campo TS

Infine, nella tabella 3.13, mostro dei risultati ottenuti utilizzando l'algoritmo di branch & price in modo euristico dove i sottoproblemi di *pricing* sono risolti con l'algoritmo *greedy* e con l'algoritmo di branch & cut troncato al nodo radice. Ho utilizzato un tempo limite di un'ora.

Il significato dei campi della tabella 3.13 è il seguente:

1. Istanza: è il nome dell'istanza; la coppia di interi separati dal carattere "x" indica la dimensione della griglia, il valore numerico finale rappresenta la soglia τ
2. NR: indica il numero di regioni base presenti nell'istanza
3. K: indica il valore utilizzato per il parametro k relativo al numero di arborescenze
4. DR: indica la dimensione delle regioni base
5. NCF: indica il numero di celle che forma la foresta nella soluzione trovata
6. OBJ: è il valore della funzione obiettivo ($\frac{NCF}{K}$)
7. T: è il tempo, espresso in secondi, utilizzato dall'algoritmo

Istanza	**TS**	**K**	**TR**	**LBR**	**UBR**	**BUB**	**OBJ**	**SO**	**TP**	**T**
1-dataKF1-15x15-0.05	O	1	812.14	15.00	15.00	15.00	15.00	15.00	799.32	812.14
1-dataKF1-15x15-0.05	B&CT	1	10.78	15.00	15.00	15.00	15.00	15.00	10.57	10.78
1-dataKF1-15x15-0.05	G	1	0.02	17.00	17.00	17.00	17.00	15.00	0.01	0.02
2-dataKF1-15x15-0.05	O	2	735.19	17.00	17.00	17.00	8.50	8.50	723.12	735.19
2-dataKF1-15x15-0.05	B&CT	2	9.34	17.00	23.00	17.00	8.50	8.50	43.99	44.79
2-dataKF1-15x15-0.05	G	2	0.02	23.00	23.00	23.00	11.50	8.50	0.01	0.02
3-dataKF1-15x15-0.05	O	3	912.56	20.50	/	21.00	7.00	7.00	2679.33	2812.54
3-dataKF1-15x15-0.05	B&CT	3	8.34	21.00	/	21.00	7.00	7.00	33.82	34.49
3-dataKF1-15x15-0.05	G	3	0.03	/	/	/	/	7.00	0.02	0.03
4-dataKF1-15x15-0.05	O	4	981.12	23.00	23.00	23.00	5.75	5.75	967.69	981.12
4-dataKF1-15x15-0.05	B&CT	4	57.50	24.00	24.00	24.00	6.00	5.75	56.36	57.50
4-dataKF1-15x15-0.05	G	4	0.03	/	/	/	/	5.75	0.02	0.03
5-dataKF1-15x15-0.10	O	1	234.89	10.00	10.00	10.00	10.00	10.00	227.19	234.89
5-dataKF1-15x15-0.10	B&CT	1	4.75	10.00	10.00	10.00	10.00	10.00	4.65	4.75
5-dataKF1-15x15-0.10	G	1	0.02	10.00	10.00	10.00	10.00	10.00	0.01	0.02
6-dataKF1-15x15-0.10	O	2	287.19	12.00	12.00	12.00	6.00	6.00	280.72	287.19
6-dataKF1-15x15-0.10	B&CT	2	34.53	12.00	12.00	12.00	6.00	6.00	33.85	34.53
6-dataKF1-15x15-0.10	G	2	0.02	13.00	13.00	13.00	6.50	6.00	0.01	0.02
7-dataKF1-15x15-0.10	O	3	837.15	14.00	/	14.00	4.67	4.67	828.19	837.15
7-dataKF1-15x15-0.10	B&CT	3	12.64	14.00	/	14.00	4.67	4.67	35.01	35.69
7-dataKF1-15x15-0.10	G	3	0.01	/	/	/	/	4.67	0.01	0.01
8-dataKF1-15x15-0.10	O	4	201.38	17.00	17.00	17.00	4.25	4.25	198.27	201.38
8-dataKF1-15x15-0.10	B&CT	4	20.79	17.00	17.00	17.00	4.25	4.25	20.39	20.79
8-dataKF1-15x15-0.10	G	4	0.02	/	/	/	/	4.25	0.01	0.02

Tabella 3.5: Risultati ottenuti con una regione base di dimensione 2x2 e per valori di τ di 0.05 e 0.10

Istanza	**TS**	**K**	**TR**	**LBR**	**UBR**	**BUB**	**OBJ**	**OS**	**TP**	**T**
9-dataKF1-15x15-0.20	O	1	198.12	8.00	8.00	8.00	8.00	8.00	195.74	198.12
9-dataKF1-15x15-0.20	B&CT	1	8.62	8.00	8.00	8.00	8.00	8.00	8.45	8.62
9-dataKF1-15x15-0.20	G	1	0.02	8.00	8.00	8.00	8.00	8.00	0.01	0.02
10-dataKF1-15x15-0.20	O	2	389.32	8.00	8.00	8.00	4.00	4.00	380.37	389.32
10-dataKF1-15x15-0.20	B&CT	2	10.05	8.00	8.00	8.00	4.00	4.00	9.83	10.05
10-dataKF1-15x15-0.20	G	2	0.02	8.00	8.00	8.00	4.00	4.00	0.01	0.02
11-dataKF1-15x15-0.20	O	3	107.49	10.00	10.00	10.00	3.33	3.33	104.48	107.49
11-dataKF1-15x15-0.20	B&CT	3	11.89	10.00	10.00	10.00	3.33	3.33	11.64	11.89
11-dataKF1-15x15-0.20	G	3	0.04	10.00	10.00	10.00	3.33	3.33	0.02	0.04
12-dataKF1-15x15-0.20	O	4	85.28	12.00	12.00	12.00	3.00	3.00	81.41	85.28
12-dataKF1-15x15-0.20	B&CT	4	4.74	12.00	12.00	12.00	3.00	3.00	4.63	4.74
12-dataKF1-15x15-0.20	G	4	0.01	12.00	12.00	12.00	3.00	3.00	0.01	0.01
13-dataKF1-15x15-0.40	O	1	389.37	6.00	6.00	6.00	6.00	6.00	385.54	389.37
13-dataKF1-15x15-0.40	B&CT	1	9.43	6.00	6.00	6.00	6.00	6.00	9.24	9.43
13-dataKF1-15x15-0.40	G	1	0.02	6.00	6.00	6.00	6.00	6.00	0.01	0.02
14-dataKF1-15x15-0.40	O	2	292.43	6.00	6.00	6.00	3.00	3.00	287.74	292.43
14-dataKF1-15x15-0.40	B&CT	2	5.32	6.00	6.00	6.00	3.00	3.00	5.13	5.32
14-dataKF1-15x15-0.40	G	2	0.04	6.00	6.00	6.00	3.00	3.00	0.02	0.04
15-dataKF1-15x15-0.40	O	3	298.65	8.00	8.00	8.00	2.67	2.67	291.57	298.65
15-dataKF1-15x15-0.40	B&CT	3	4.29	8.00	8.00	8.00	2.67	2.67	4.01	4.29
15-dataKF1-15x15-0.40	G	3	0.02	8.00	8.00	8.00	2.67	2.67	0.01	0.02
16-dataKF1-15x15-0.40	O	4	108.18	10.00	10.00	10.00	2.50	2.50	103.84	108.18
16-dataKF1-15x15-0.40	B&CT	4	2.63	10.00	10.00	10.00	2.50	2.50	2.41	2.63
16-dataKF1-15x15-0.40	G	4	0.02	10.00	10.00	10.00	2.50	2.50	0.01	0.02

Tabella 3.6: Risultati ottenuti con una regione base di dimensione 2x2 e per valori di τ di 0.2 e 0.4

Istanza	**TS**	**K**	**TR**	**LBR**	**UBR**	**BUB**	**OBJ**	**OS**	**TP**	**T**
1-dataKF1-15x15-0.05	O	3	5173.77	28.00	28.00	28.00	9.33	9.33	5074.75	5173.77
1-dataKF1-15x15-0.05	B&CT	3	53.01	28.00	28.00	28.00	9.33	9.33	52.17	53.01
1-dataKF1-15x15-0.05	G	3	0.03	/	/	/	/	/	0.02	0.03
2-dataKF1-15x15-0.05	O	4	5333.87	29.00	29.00	29.00	7.25	7.25	5202.68	5333.87
2-dataKF1-15x15-0.05	B&CT	4	43.32	29.00	29.00	29.00	7.25	7.25	42.59	43.32
2-dataKF1-15x15-0.05	G	4	0.02	/	/	/	/	/	0.01	0.02
3-dataKF1-15x15-0.05	O	5	5521.37	31.00	31.00	31.00	6.20	6.20	5391.49	5521.37
3-dataKF1-15x15-0.05	B&CT	5	43.80	31.00	31.00	31.00	6.20	6.20	43.10	43.80
3-dataKF1-15x15-0.05	G	5	0.02	/	/	/	/	/	0.01	0.02
4-dataKF1-15x15-0.05	O	6	7662.36	33.00	33.00	33.00	5.50	5.50	7512.12	7662.36
4-dataKF1-15x15-0.05	B&CT	6	47.21	33.00	33.00	33.00	5.50	5.50	46.46	47.21
4-dataKF1-15x15-0.05	G	6	0.02	/	/	/	/	/	0.02	0.03
5-dataKF1-15x15-0.05	O	7	5327.78	37.00	/	37.00	5.29	5.29	16329.91	16593.45
5-dataKF1-15x15-0.05	B&CT	7	37.38	37.00	/	37.00	5.29	5.29	78.67	80.01
5-dataKF1-15x15-0.05	G	7	0.02	/	/	/	/	/	0.01	0.02
6-dataKF1-15x15-0.05	O	8	5997.45	40.00	40.00	40.00	5.00	5.00	5802.98	5997.45
6-dataKF1-15x15-0.05	B&CT	8	58.17	40.00	40.00	40.00	5.00	5.00	57.21	58.17
6-dataKF1-15x15-0.05	G	8	0.02	/	/	/	/	/	0.01	0.02

Tabella 3.7: Risultati ottenuti con 2 regioni base di dimensione 2x2 e per τ pari a 0.05

Istanza	TS	K	TR	LBR	UBR	BUB	OBJ	OS	TP	T
7-dataKF1-15x15-0.10	O	3	5005.09	19.00	19.00	19.00	6.33	6.33	4897.25	5005.09
7-dataKF1-15x15-0.10	B&CT	3	11.72	19.00	19.00	19.00	6.33	6.33	11.48	11.72
7-dataKF1-15x15-0.10	G	3	0.03	/	/	/	/	/	0.02	0.03
8-dataKF1-15x15-0.10	O	4	4639.56	20.50	/	21.00	5.25	5.25	25848.57	26409.40
8-dataKF1-15x15-0.10	B&CT	4	10.96	20.50	/	21.00	5.25	5.25	22.19	22.61
8-dataKF1-15x15-0.10	G	4	0.03	/	/	/	/	/	0.02	0.03
9-dataKF1-15x15-0.10	O	5	4532.21	22.50	24.00	23.00	4.60	4.60	26398.56	27946.30
9-dataKF1-15x15-0.10	B&CT	5	11.45	23.00	24.00	23.00	4.60	4.60	22.57	22.99
9-dataKF1-15x15-0.10	G	5	0.02	/	/	/	/	/	0.01	0.02
10-dataKF1-15x15-0.10	O	6	3989.76	24.50	25.00	25.00	4.17	4.17	3811.32	3989.76
10-dataKF1-15x15-0.10	B&CT	6	10.98	24.50	25.00	25.00	4.17	4.17	28.90	29.41
10-dataKF1-15x15-0.10	G	6	0.02	/	/	/	/	/	0.01	0.02
11-dataKF1-15x15-0.10	O	7	3655.21	27.00	28.00	27.00	3.86	3.86	7945.73	8139.23
11-dataKF1-15x15-0.10	B&CT	7	13.67	27.00	28.00	27.00	3.86	3.86	34.30	34.93
11-dataKF1-15x15-0.10	G	7	0.02	/	/	/	/	/	0.01	0.02
12-dataKF1-15x15-0.10	O	8	2987.38	30.00	30.00	30.00	3.75	3.75	2871.91	2987.38
12-dataKF1-15x15-0.10	B&CT	8	36.75	30.00	30.00	30.00	3.75	3.75	36.16	36.75
12-dataKF1-15x15-0.10	G	8	0.02	/	/	/	/	/	0.01	0.02

Tabella 3.8: Risultati ottenuti con 2 regioni base di dimensione 2x2 e per τ pari a 0.10

Istanza	**TS**	**K**	**TR**	**LBR**	**UBR**	**BUB**	**OBJ**	**OS**	**TP**	**T**
13-dataKF1-15x15-0.20	O	3	9312.33	15.00	15.00	15.00	5.00	5.00	9158.34	9312.33
13-dataKF1-15x15-0.20	B&CT	3	4.45	15.00	15.00	15.00	5.00	5.00	4.34	4.45
13-dataKF1-15x15-0.20	G	3	0.04	15.00	15.00	15.00	5.00	5.00	0.02	0.03
14-dataKF1-15x15-0.20	O	4	2257.11	15.00	15.00	15.00	3.00	3.75	2227.92	2257.11
14-dataKF1-15x15-0.20	B&CT	4	4.20	15.00	15.00	15.00	3.75	3.75	4.10	4.20
14-dataKF1-15x15-0.20	G	4	0.04	15.00	15.00	15.00	3.75	3.75	0.02	0.04
15-dataKF1-15x15-0.20	O	5	3245.32	16.00	16.00	16.00	3.20	3.20	3201.92	3245.32
15-dataKF1-15x15-0.20	B&CT	5	9.48	16.00	16.00	16.00	3.20	3.20	9.25	9.48
15-dataKF1-15x15-0.20	G	5	0.04	16.00	16.00	16.00	3.20	3.20	0.02	0.04
16-dataKF1-15x15-0.20	O	6	1837.87	18.00	/	18.00	3.00	3.00	4521.88	4643.14
16-dataKF1-15x15-0.20	B&CT	6	5.45	18.00	/	18.00	3.00	3.00	12.62	12.94
16-dataKF1-15x15-0.20	G	6	0.07	18.00	/	18.00	3.00	3.00	0.04	0.07
17-dataKF1-15x15-0.20	O	7	589.12	20.00	20.00	20.00	2.86	2.86	577.63	589.12
17-dataKF1-15x15-0.20	B&CT	7	9.92	20.00	20.00	20.00	2.86	2.86	9.73	9.92
17-dataKF1-15x15-0.20	G	7	0.03	20.00	20.00	20.00	2.86	2.86	0.02	0.03
18-dataKF1-15x15-0.20	O	8	481.32	22.00	22.00	22.00	2.75	2.75	476.54	481.32
18-dataKF1-15x15-0.20	B&CT	8	6.70	22.00	22.00	22.00	2.75	2.75	6.55	6.70
18-dataKF1-15x15-0.20	G	8	0.03	/	/	/	/	/	0.02	0.03

Tabella 3.9: Risultati ottenuti con 2 regioni base di dimensione 2x2 e per τ pari a 0.20

Istanza	**TS**	**K**	**TR**	**LBR**	**UBR**	**BUB**	**OBJ**	**OS**	**TP**	**T**
19-dataKF1-15x15-0.40	O	3	781.14	12.00	12.00	12.00	4.00	4.00	777.02	781.14
19-dataKF1-15x15-0.40	B&CT	3	9.67	12.00	12.00	12.00	4.00	4.00	9.46	9.67
19-dataKF1-15x15-0.40	G	3	0.03	12.00	12.00	12.00	4.00	4.00	0.02	0.03
20-dataKF1-15x15-0.40	O	4	1353.89	12.00	12.00	12.00	3.00	3.00	1339.20	1353.89
20-dataKF1-15x15-0.40	B&CT	4	16.00	12.00	12.00	12.00	3.00	3.00	15.68	16.00
20-dataKF1-15x15-0.40	G	4	0.03	13.00	13.00	13.00	3.25	3.25	0.02	0.03
21-dataKF1-15x15-0.40	O	5	1023.87	13.00	13.00	13.00	2.60	2.60	1009.18	1023.87
21-dataKF1-15x15-0.40	B&CT	5	14.30	13.00	13.00	13.00	2.60	2.60	13.99	14.30
21-dataKF1-15x15-0.40	G	5	0.03	14.00	14.00	14.00	2.80	2.80	0.02	0.03
22-dataKF1-15x15-0.40	O	6	1078.24	14.00	14.00	14.00	2.33	2.33	1065.38	1078.24
22-dataKF1-15x15-0.40	B&CT	6	13.30	14.00	14.00	14.00	2.33	2.33	12.95	13.30
22-dataKF1-15x15-0.40	G	6	0.03	15.00	15.00	15.00	2.50	2.50	0.05	0.06
23-dataKF1-15x15-0.40	O	7	819.01	16.00	16.00	16.00	2.29	2.29	814.73	819.01
23-dataKF1-15x15-0.40	B&CT	7	4.38	16.00	16.00	16.00	2.29	2.29	4.25	4.38
23-dataKF1-15x15-0.40	G	7	0.03	16.00	16.00	16.00	2.29	2.29	0.02	0.03
24-dataKF1-15x15-0.40	O	8	691.09	18.00	18.00	18.00	2.25	2.25	686.93	691.09
24-dataKF1-15x15-0.40	B&CT	8	2.93	18.00	18.00	18.00	2.25	2.25	2.86	2.93
24-dataKF1-15x15-0.40	G	8	0.03	18.00	18.00	18.00	2.25	2.25	0.02	0.03

Tabella 3.10: Risultati ottenuti con 2 regioni base di dimensione 2x2 e per τ pari a 0.40

Istanza	**TS**	**K**	**TR**	**LBR**	**UBR**	**BUB**	**OBJ**	**OS**	**TP**	**T**
1-dataKF1-15x15-0.05	O	2	2389.32	18.00	18.00	18.00	9.00	9.00	2351.33	2389.32
1-dataKF1-15x15-0.05	B&CT	2	58.34	18.00	18.00	18.00	9.00	9.00	57.41	58.34
1-dataKF1-15x15-0.05	G	2	0.06	22.00	22.00	22.00	11.00	9.00	0.05	0.06
2-dataKF1-15x15-0.05	O	4	4191.18	23.50	/	24.00	6.00	6.00	10494.23	10663.78
2-dataKF1-15x15-0.05	B&CT	4	287.13	23.50	/	24.00	6.00	6.00	369.31	375.28
2-dataKF1-15x15-0.05	G	4	0.02	/	/	/	/	6.00	0.02	0.02
3-dataKF1-15x15-0.05	O	6	1743.69	29.00	29.00	29.00	4.83	4.83	1715.97	1743.69
3-dataKF1-15x15-0.05	B&CT	6	101.45	30.00	30.00	30.00	5.00	4.83	99.84	101.45
3-dataKF1-15x15-0.05	G	6	0.02	/	/	/	/	6.00	0.02	0.02
4-dataKF1-15x15-0.05	O	8	1481.78	35.00	35.00	35.00	4.38	4.37	1458.22	1481.78
4-dataKF1-15x15-0.05	B&CT	8	49.32	35.00	35.00	35.00	4.38	4.37	48.54	49.32
4-dataKF1-15x15-0.05	G	8	0.02	/		/	/	4.37	0.02	0.02
5-dataKF1-15x15-0.10	O	2	1881.19	12.00	12.00	12.00	6.00	6.00	1851.28	1881.19
5-dataKF1-15x15-0.10	B&CT	2	38.87	12.00	12.00	12.00	6.00	6.00	38.25	38.87
5-dataKF1-15x15-0.10	G	2	0.04	14.00	14.00	14.00	7.00	6.00	0.03	0.04
6-dataKF1-15x15-0.10	O	4	5919.32	16.75	18.00	17.00	4.25	4.25	9666.27	9822.45
6-dataKF1-15x15-0.10	B&CT	4	212.34	17.00	19.00	18.00	4.50	4.25	577.76	587.09
6-dataKF1-15x15-0.10	G	4	0.02	/	/	/	/	4.25	0.02	0.02
7-dataKF1-15x15-0.10	O	6	1366.25	19.50	20.00	20.00	3.33	3.33	3387.39	3442.12
7-dataKF1-15x15-0.10	B&CT	6	94.41	20.50	21.00	20.00	3.33	3.33	186.20	189.21
7-dataKF1-15x15-0.10	G	6	0.02	/	/	/	/	3.33	0.02	0.02
8-dataKF1-15x15-0.10	O	8	819.92	27.00	27.00	27.00	3.38	3.37	806.88	819.92
8-dataKF1-15x15-0.10	B&CT	8	31.44	27.00	27.00	27.00	3.38	3.37	30.94	31.44
8-dataKF1-15x15-0.10	G	8	0.03	/	/	/	/	3.37	0.02	0.03

Tabella 3.11: Risultati ottenuti con 1 regione base di dimensione 3x3 e per valori di τ di 0.05 e 0.10

Istanza	**TS**	**K**	**TR**	**LBR**	**UBR**	**BUB**	**OBJ**	**OS**	**TP**	**T**
9-dataKF1-15x15-0.20	O	2	818.67	13.00	13.00	13.00	6.50	6.50	805.65	818.67
9-dataKF1-15x15-0.20	B&CT	2	21.54	13.00	13.00	13.00	6.50	6.50	21.20	21.54
9-dataKF1-15x15-0.20	G	2	0.03	14.50	15.00	15.00	7.50	6.50	0.09	0.11
10-dataKF1-15x15-0.20	O	4	2019.34	15.00	15.00	15.00	3.75	3.75	1987.23	2019.34
10-dataKF1-15x15-0.20	B&CT	4	51.32	15.00	15.00	15.00	3.75	3.75	50.50	51.32
10-dataKF1-15x15-0.20	G	4	0.04	16.00	16.00	16.00	4.00	3.75	0.03	0.04
11-dataKF1-15x15-0.20	O	6	691.32	20.00	20.00	20.00	3.33	3.33	680.33	691.32
11-dataKF1-15x15-0.20	B&CT	6	48.14	20.00	20.00	20.00	3.33	3.33	47.37	48.14
11-dataKF1-15x15-0.20	G	6	0.04	/	/	/	/	3.33	0.03	0.04
12-dataKF1-15x15-0.20	O	8	901.32	23.00	23.00	23.00	2.88	2.88	886.99	901.32
12-dataKF1-15x15-0.20	B&CT	8	14.76	23.00	23.00	23.00	2.88	2.88	14.53	14.76
12-dataKF1-15x15-0.20	G	8	0.04	/	/	/	/	2.88	0.03	0.04
13-dataKF1-15x15-0.40	O	2	829.34	12.00	12.00	12.00	6.00	6.00	816.15	829.34
13-dataKF1-15x15-0.40	B&CT	2	17.75	12.00	12.00	12.00	6.00	6.00	17.47	17.75
13-dataKF1-15x15-0.40	G	2	0.05	12.00	12.00	12.00	6.00	6.00	0.04	0.05
14-dataKF1-15x15-0.40	O	4	1759.71	14.00	14.00	14.00	3.50	3.50	1731.73	1759.71
14-dataKF1-15x15-0.40	B&CT	4	48.87	14.00	14.00	14.00	3.50	3.50	48.09	48.87
14-dataKF1-15x15-0.40	G	4	0.05	15.00	15.00	15.00	3.75	3.50	0.04	0.05
15-dataKF1-15x15-0.40	O	6	536.72	17.00	17.00	17.00	2.83	2.83	528.19	536.72
15-dataKF1-15x15-0.40	B&CT	6	15.47	17.00	17.00	17.00	2.83	2.83	15.22	15.47
15-dataKF1-15x15-0.40	G	6	0.04	19.00	19.00	19.00	3.17	2.83	0.03	0.04
16-dataKF1-15x15-0.40	O	8	601.89	20.00	20.00	20.00	2.50	2.50	592.32	601.89
16-dataKF1-15x15-0.40	B&CT	8	9.89	20.00	20.00	20.00	2.50	2.50	9.73	9.89
16-dataKF1-15x15-0.40	G	8	0.04	/	/	/	/	2.50	0.03	0.04

Tabella 3.12: Risultati ottenuti con 1 regione base di dimensione 3x3 e per valori di τ di 0.20 e 0.40

Istanza	**NR**	**DR**	**K**	**NCF**	**OBJ**	**T**
1-data-EUR-20x20-0.05	3	2x2	12	59	4.92	238.29
1-data-EUR-20x20-0.10	3	2x2	12	49	4.08	231.78
1-data-EUR-20x20-0.20	3	2x2	12	38	3.17	187.39
1-data-EUR-20x20-0.40	3	2x2	12	28	2.33	90.49
2-data-EUR-20x20-0.05	5	2x2	20	/	/	1739.69
2-data-EUR-20x20-0.05	5	2x2	19	/	/	1928.48
2-data-EUR-20x20-0.05	5	2x2	18	93	5.17	2491.38
2-data-EUR-20x20-0.10	5	2x2	20	/	/	1439.81
2-data-EUR-20x20-0.10	5	2x2	19	79	4.16	2001.64
2-data-EUR-20x20-0.20	5	2x2	20	/	/	1249.53
2-data-EUR-20x20-0.20	5	2x2	19	61	3.21	1191.83
2-data-EUR-20x20-0.40	5	2x2	20	45	2.25	963.18
3-data-EUR-20x20-0.05	3	3x3	24	aborted	aborted	3600
3-data-EUR-20x20-0.10	3	3x3	24	aborted	aborted	3600
3-data-EUR-20x20-0.20	3	3x3	24	78	3.25	2301.15
3-data-EUR-20x20-0.40	3	3x3	24	54	2.25	3036.48
4-data-EUR-30x30-0.05	3	2x2	12	65	5.42	2996.19
4-data-EUR-30x30-0.10	3	2x2	12	52	4.33	2693.45
4-data-EUR-30x30-0.20	3	2x2	12	40	3.33	2227.31
4-data-EUR-30x30-0.40	3	2x2	12	27	2.25	1801.75
5-data-EUR-30x30-0.05	3	3x3	24	aborted	aborted	3600
5-data-EUR-30x30-0.10	3	3x3	24	aborted	aborted	3600
5-data-EUR-30x30-0.20	3	3x3	24	aborted	aborted	3600
5-data-EUR-30x30-0.40	3	3x3	24	52	2.17	3048.79

Tabella 3.13: Risultati euristici

Capitolo 4

Conclusioni e sviluppi futuri

I risultati ottenuti testimoniano l'elevata complessità del problema *MKFTC*. La complessità del problema *MKFTC* è in parte ereditata dal problema *MATC*.

Le difficoltà dell'algoritmo branch & cut per risolvere *MATC* dipendono fortemente dalla presenza dei pesi che possono assumere sia valori positivi che negativi, ciò è evidenziato dall'allargamento del *gap* tra LB e UB rispetto al caso in cui sono presenti solo pesi non negativi.

Nell'algoritmo branch & price, le difficoltà dell'algoritmo branch & cut per risolvere i sottoproblemi di *pricing* sono ulteriormente aumentate, perché la funzione obiettivo relativa al costo ridotto più negativo è influenzata dai valori duali dei vincoli del *L-RMP* che possono assumere valori sia positivi che negativi.

L'algoritmo branch & cut potrebbe essere migliorato ricercando un insieme di vincoli di connettività più efficaci, in modo da raggiungere gli stessi valori di LB con un insieme di vincoli di minore cardinalità. Abbiamo visto infatti che, all'aumentare del numero dei vincoli aggiunti dinamicamente, i tempi richiesti per risolvere il rilassamento lineare di *MATC* crescono rapidamente.

Un possibile miglioramento potrebbe quindi essere l'implementazione del *creep flow*[27], l'idea alla base del *creep flow* è quella di trovare il taglio composto dal minimo numero di archi. Per ricercare tale taglio, viene aggiunta una capacità aggiuntiva (ad esempio un certo $\epsilon = 10^{-6}$) agli archi in soluzione nel rilassamento prima di eseguire l'algoritmo del flusso massimo.

Un'altra possibile miglioria che si potrebbe apportare all'algoritmo branch & cut, sarebbe quella di separare l'insieme dei vincoli di connettività aggiunti dinamicamente tra i sottoproblemi con diverso nodo padre. I vincoli corri-

spondenti al taglio minimo, che hanno validità globale all'interno modello, vengono attualmente aggiunti per tutti i sottoproblemi.

Possibili sviluppi dell'algoritmo branch & price che risolve *MKFTC*, potrebbero riguardare l'aggiunta di euristiche e/o metaeuristiche come possono essere la ricerca locale o la ricerca tabù. L'algoritmo *greedy* proposto non risulta, infatti, molto efficace.

Ogni sottoproblema di *pricing* potrebbe essere eseguito in modo parallelo lanciando un *thread* per ogni sottoproblema, o forse meglio, risolvere i sottoproblemi di *pricing* mediante un gruppo di *thread*.

La complessità propria del problema *MKFTC* risiede nell'elevato numero di sottoproblemi di *pricing* per ogni regione base, e nell'esplosione combinatoria del numero di possibili arborescenze di una regione base, quando questa ha dimensioni elevate (già la dimensione 3x3 comporta un notevole aumento dei tempi richiesti).

Per limitare il numero dei sottoproblemi di *pricing*, si potrebbe risolvere un unico sottoproblema di *pricing* per ogni regione base, ciò sarebbe possibile eseguendo l'algoritmo branch & cut su un nodo fittizio che possiede un arco entrante in ogni cella base.

Un'altra possibile strada da percorrere per risolvere più efficientemente *MKFTC*, sarebbe quella di semplificare il problema da cui deriva. Una riduzione dei tempi necessari per risolvere *MKFTC*, potrebbe essere raggiunta approssimando il modello definito da *PROBE*.

Ad esempio, si potrebbe rinunciare alla combinazione delle celle base che forma una certa arborescenza. In questo modo, si potrebbe far collassare la regione base in un "super nodo" che mantiene lo stesso insieme di archi della regione base.

Il numero di arborescenze, in questo caso, potrebbe essere controllato da un vincolo nel *L-RMP* che obbliga l'utilizzo di k archi tra i super nodi. Ogni cella base di contorno ha infatti esattamente un arco che la collega ad una cella esterna.

Così facendo, sarebbe necessario un ulteriore vincolo che obbliga l'utilizzo di almeno un arco per ogni super nodo (obbliga ad avere almeno una arborescenza per ogni super nodo).

In questo nuovo contesto, il problema di *pricing* ne risulterebbe semplificato. Infatti, per ogni regione base esisterebbe un unico sottoproblema di *pricing* e non uno per ogni cella di contorno, inoltre, non esisterebbe più il problema derivante dalla combinazione delle celle base.

Bibliografia

[1] M. L. Damiani, E. Bertino, and C. Silvestri, *The PROBE Framework for the Personalized Cloaking of Private Locations. Transactions on Data Privacy, vol. (3)2, pp. 123–148, 2010*

[2] M. L. Damiani, C. Silvestri and E. Bertino, *Analyzing Semantic Location Cloaking in a Probabilistic Grid-Based Map, Proc. 18th ACM SIGSpatial Int'l Conf. Advances in Geographic Informa- tion Systems (ACM GIS), ACM Press, 2010, pp. 522–523*

[3] M. L. Damiani, C. Silvestri, and E. Bertino, *Fine-Grained Cloaking of Sensitive Positions in Location-Sharing Applications. IEEE Pervasive Computing, IEEE Pervasive Computing, vol. 10(4), pp. 64–72, 2011*

[4] J.KRARUP, *J.KRARUP, The generalized Steiner problem, unpublished note, 1978*

[5] A. Agrawal, P. Klein, and R. Ravi, *When trees collide: An approximation algorithm for the generalized steiner problem in networks. SIAM Journal on Computing, 24(3):440–456, 1995*

[6] M. Goemans and D. Williamson, *A general approximation technique for constrained forest problems. SIAM J. Comput., 24(2):296–317, 1995*

[7] H.N. GABOW, M.X.GOEMANS, AND D.P. Williamson, *An efficient approximation algorithm for the survivable network design problem, Proc. 3rd MPS Conference on Integer Programming and Combinatorial Optimization, 1993, pp. 57-74*

[8] D.P. WILLIAMSON, M.X. GOEMANS M. MIHAIL, and V.V. VAZIRANI, *A primal-dual approximation algorithm for generalized Steiner*

network problems, Proc. 25th ACM Symposium on Theory of Computing, 1993. pp. 708-717

[9] M. X. GOEMANS, A. V. GOLDBERG, S. PLOTKIN, D. SHMOYS, È. TARDOS, AND D. P. WILLIAMSON, *Improved approximation algorithms for network design problems, Proc. 5th ACM-SIAM Symposium on Discrete Algorithms, 1994, pp. 223-232*

[10] S. Chopra, E. Gorres, and M. R. Rao, *Solving a Steiner tree problem on a graph using branch and cut. ORSA J. Comput. 4 (1992) 320–335*

[11] I. Ljubić, R. Weiskircher, U. Pferschy, G. W. Klau, M. Fischetti, *An Algorithmic Framework for the Exact Solution of the Prize-Collecting Steiner Tree Problem*

[12] V.J.RAYWARD-SMITH, *Tile computation of nearly minimal Steiner trees in graphs, Internat. J. Math. Ed.Sci. Tech., 14 (1983), pp. 15-23*

[13] *A dual ascent approach for Steiner tree problems on a directed graph, Math. Programming, 28 (1984), pp. 271-287*

[14] P. BERMAN AND V.RAMAIYER, *Improved approximations for the Steiner tree problem, Proc.3rd Annual ACMSIAM Symposium on Discrete Algorithms, 1992, pp. 325-334*

[15] H. TAKAHASHI AND A.MATSUYAMA, *An approximate solution for the Steiner problem in graphs , Math. Japonica, 24 (1980), pp. 573-577*

[16] Y.S. Myung, C.H. Lee, D.W. Tcha, *On The Generalized Minimum Spanning Tree Problem, Networks 26, (1995), pp. 231-241*

[17] B. Golden, S. Raghavan, D. Stanojević (2007), *The prize-collecting generalized minimum spanning tree problem. Springer Science+Business Media, LLC 2007*

[18] CPLEX, *http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/*

[19] SCIP, *http://www.scip.zib.de*

[20] John E., Mitchell (2002), *Branch-and-Cut Algorithms for Combinatorial Optimization Problems. Handbook of Applied Optimization: 65–77*

[21] E. Lawler, *4.5. Combinatorial Implications of Max-Flow Min-Cut Theorem, 4.6. Linear Programming Interpretation of Max-Flow Min-Cut Theorem. Combinatorial Optimization: Networks and Matroids. Dover. pp. 117–120. ISBN 0-486-41453-1*

[22] Christos H. Papadimitriou, K. Steiglitz (1998), *6.1 The Max-Flow, Min-Cut Theorem. Combinatorial Optimization: Algorithms and Complexity. Dover. pp. 120–128. ISBN 0-486-40258-4*

[23] Vijay V. Vazirani (2004), *12. Introduction to LP-Duality. Approximation Algorithms. Springer. pp. 93–100. ISBN 3-540-65367-8*

[24] Cormen, Leiserson, Rivest, Stein, *Introduction to Algorithms, Chapter 16 "Greedy Algorithms" (2001)*

[25] J. Bang-Jensen, G. Gutin and A. Yeo, *When the greedy algorithm fails. Discrete Optimization 1 (2004), 121–127*

[26] Dantzig, G.B., P. Wolfe, *Wolfe, Decomposition principle for linear programs, Operations Research, 8, 101-111 (1960)*

[27] Koch,T.,Martin,A, *Solving Steiner tree problems in graphs to optimality. Networks,32,207–232(1998)*

Ringraziamenti

"Happiness is real only when shared." (Chris McCandless)

Senza dubbio, questa, è la citazione che preferisco. Credo di potermi considerare fortunato, sono molte le persone con cui posso condividere la gioia di questo momento, il raggiungimento di un importante traguardo di vita come la laurea magistrale.
Le stesse persone che, nel tempo, mi hanno aiutato a diventare la persona che sono oggi e che, ognuna a suo modo, mi hanno supportato durante questo percorso.
Di seguito voglio quindi ringraziare sia chi ha reso possibile lo svolgimento di questo lavoro di tesi sia coloro i quali in questi anni hanno sempre creduto in me, supportandomi ed incoraggiandomi:

- Innanzitutto il Prof. Giovanni Righini, relatore del lavoro, per la disponibilità e la cortesia dimostrata e per i preziosi consigli profusi sia in fase di scelta e definizione del progetto di tesi che durante la scrittura dell'elaborato.

- Al Prof. Alberto Ceselli, correlatore della tesi e punto di riferimento a cui rivolgere ogni dubbio, problematica o più semplicemente idea legata al progetto di tesi. Illuminante.

- Il più importante e sentito riconoscimento va ovviamente alla mia famiglia, i miei fratelli e soprattutto ai miei genitori, Giovanni e Maria, che con il loro incrollabile sostegno morale ed economico, mi hanno, oltre che permesso di raggiungere questo traguardo, costantemente supportato e consigliato.

- Debora, la mia fidanzata e compagna di tanti momenti, la quale, più di tutti, ha vissuto i miei/nostri attimi di gioia e di tristezza, di entusiasmo

ma anche di sconforto. Grazie per essere sempre al mio fianco. Grazie anche per le immagini!

- I miei compagni e amici di università: Mattia Terzi, Giuseppe Restivo, Andrea Taverna, Guido Lena Cota, Marco Casazza, Francesco Padovani, Adriana Novaes, Robert Anyumba, Andrea Mangiavini, Fabio Lazzaroni, Emanuele Rossi, i quali, in questo momento più di tutti possono comprendere il mio grado di soddisfazione.

- I miei amici più cari, quelli che conosco da sempre e quelli della 5^aA. Grazie perché ogni attimo vissuto con voi è un attimo di un altro mondo.