

Università degli Studi di Roma
“La Sapienza”



Tesi di Laurea

Pianificazione e Ottimizzazione di Interventi
Migliorativi su Reti di Comunicazione

Studente:

Luca La Rocca
matr. 09083159

Relatore:

prof. Alberto Marchetti Spaccamela
D.I.S. “La Sapienza”

Facoltà di Ingegneria
Corso di Laurea in Ingegneria Elettronica
Anno Accademico 1996/97

Indice

| | |
|---|-----------|
| Presentazione | 3 |
| 1 Introduzione alla Complessità Computazionale | 5 |
| 1.1 Complessità di Algoritmi | 5 |
| 1.1.1 Algoritmi Approssimati | 9 |
| 1.2 Complessità di Problemi | 10 |
| 2 Miglioramento di una Rete | 13 |
| 2.1 Modelli di Rete | 13 |
| 2.2 Modelli di Decisione | 17 |
| 3 Problemi di Cammino Minimo | 20 |
| 3.1 Introduzione al Cammino Minimo | 20 |
| 3.2 Diminuzione Discreta del Cammino Minimo | 23 |
| 3.2.1 Complessità Computazionale | 24 |
| 3.2.2 Risoluzione Esatta | 25 |
| 3.2.3 Risoluzione Approssimata | 29 |
| 3.3 Diminuzione Continua del Cammino Minimo | 32 |
| 3.3.1 Risoluzione Esatta | 33 |
| 3.3.2 Risoluzione Approssimata | 37 |
| 3.4 Accrescimento del Cammino Minimo | 40 |
| 3.5 Prospettive e Sviluppi | 42 |
| 4 Problemi di Flusso Massimo | 44 |
| 4.1 Introduzione al Flusso Massimo | 44 |
| 4.2 Diminuzione del Flusso Massimo | 46 |
| 4.2.1 Grafi Planari | 47 |
| 4.3 Accrescimento del Flusso Massimo | 50 |
| 4.4 Prospettive e Sviluppi | 54 |

| | | |
|----------|--|-----------|
| 5 | Problemi di Minimo Albero Ricoprente | 55 |
| 5.1 | Introduzione al Minimo Albero Ricoprente | 55 |
| 5.2 | Il Minimo Albero Ricoprente Vincolato | 58 |
| 5.3 | Diminuzione del Minimo Albero Ricoprente | 59 |
| 5.4 | Accrescimento del Minimo Albero Ricoprente | 69 |
| 5.5 | Prospettive e Sviluppi | 74 |
| 6 | Situazioni di Rischio o Incertezza | 76 |
| 6.1 | Nozione di Utilità | 77 |
| 6.2 | Problemi di Cammino Minimo | 78 |
| 6.3 | Problemi di Flusso Massimo | 82 |
| 6.4 | Problemi di Minimo Albero Ricoprente | 83 |
| | Bibliografia | 86 |

Presentazione

Quando ci si trova a gestire una rete di comunicazione le cui prestazioni non possano considerarsi soddisfacenti, al momento o in prospettiva futura, è necessario pianificare su di essa una serie di interventi migliorativi, tenendo conto non solo del beneficio che da questi si può trarre ma anche del loro costo: scopo di questa tesi è fornire strumenti efficaci ed efficienti per affrontare questa situazione.

Poiché ovviamente si desidera una pianificazione che sia la migliore possibile (secondo un qualche criterio) il lavoro rientra nell'ambito di quella disciplina che va sotto il nome di *ottimizzazione* e in particolare, per la natura degli oggetti matematici studiati, nell'ambito dell'ottimizzazione *combinatoria*. L'approccio è di tipo algoritmico (quindi prettamente informatico) e l'attenzione è rivolta alle garanzie che si possono dare sulle prestazioni degli algoritmi presentati, sia in termini di *qualità* della soluzione che in termini di *velocità* di esecuzione.

La trattazione è organizzata in sei capitoli, ciascuno preceduto da un breve sommario; i capitoli dal terzo al quinto prevedono anche una sezione conclusiva dedicata a inquadrarli nella giusta prospettiva "storica". Segue una breve lista dei contenuti di ogni capitolo, volta a dare una visione di insieme dell'opera.

- Il *primo* capitolo contiene dei richiami di *teoria della complessità computazionale*, strumento mediante il quale nei capitoli successivi vengono valutate le prestazioni degli algoritmi.
- Il *secondo* capitolo chiarisce i termini nei quali questo lavoro affronta il problema della pianificazione ottima di interventi migliorativi su reti di comunicazione, fornendo un *modello generale* per i problemi formulati nel seguito.
- I capitoli dal *terzo* al *quinto* sono dedicati in modo monografico a tre possibili *misure* per le prestazioni di una rete, in corrispondenza con tre classici problemi di ottimizzazione su grafi: cammino minimo, flusso massimo e minimo albero ricoprente.

- Il *sesto* capitolo considera le tre suddette misure in quelle situazioni nelle quali il problema risulta definito su *dati aleatori*: si tratta di un inevitabile punto di partenza per affrontare problemi di pianificazione stocastica.

I miei più sentiti ringraziamenti vanno a tutti coloro che hanno reso possibile la realizzazione di questa tesi: in particolare al prof. Alberto Marchetti Spaccamela per l'opportunità offertami e per la valida attività di indirizzo e al prof. Giampaolo Scalia Tomba per le stimolanti conversazioni.

Da un punto di vista più spicciolo (ma non meno importante) sono debitore da una parte nei confronti del mio amico e collega Raffaele Ramires per l'assistenza "software" e dall'altra nei confronti dei bibliotecari del DIS¹, dell'Istituto "Guido Castelnuovo", del Dipartimento di Statistica² (Francesco Di Stefano e Francesco Di Bernardino) e dello IASI³ (Franco Di Giacinto ed Elisabetta D'Alessandro) per la cortesia e la disponibilità mostrate verso la mia inesperienza.

Più in generale vorrei esprimere la mia gratitudine verso tutti coloro che in questi anni hanno contribuito alla mia crescita umana e culturale; in particolare desidero menzionare il mio amico e collega Alfonso Morriello e i miei genitori, preziosi per pazienza e discrezione.

¹Dipartimento di Informatica e Sistemistica

²Dipartimento di Statistica, Probabilità e Statistiche Applicate

³Istituto di Analisi dei Sistemi ed Informatica

Capitolo 1

Introduzione alla Complessità Computazionale

Questo capitolo presenta gli strumenti con i quali nei capitoli successivi vengono valutate l'efficienza degli algoritmi e la difficoltà dei problemi.

La prima sezione è dedicata a chiarire le ipotesi sotto le quali si analizzano le prestazioni degli algoritmi e i modelli allo scopo adottati; la seconda ha invece per oggetto la classificazione dei problemi e in particolare la definizione del grado NP-completo.

Un'introduzione teorica alla complessità computazionale di algoritmi e problemi può essere trovata in Ausiello et al. [9] mentre Ahuja et al. [2] presentano la materia da un punto di vista più concreto, specificamente per problemi definiti su grafi; per quanto riguarda la nozione di NP-completezza e gli algoritmi approssimati si rimanda invece a Garey e Johnson [27].

1.1 Complessità di Algoritmi

Nei capitoli successivi presenteremo i problemi di ottimo trattati in una forma intuitiva (di grande diffusione in letteratura) che ha il Problema 1.1 come prototipo.

Problema 1.1 *Generica Ottimizzazione*

$$\begin{array}{ll} \max & [min] \quad f(x) \\ x & \in \quad SOL \end{array}$$

Un *problema di ottimo* è quindi individuato da uno spazio di decisione *SOL* sul quale è definita una funzione numerica non negativa f che si vuole, a seconda dei casi, massimizzare o minimizzare.

Per essere più precisi un problema di ottimo consta di un insieme I di *istanze* per ognuna delle quali sono definiti f e SOL : un algoritmo A che lo risolva prenderà in ingresso una sua qualsiasi istanza $i \in I$ e fornirà in uscita una soluzione $x^* \in SOL(i)$ tale che sia $f_i(x^*) \geq [\leq] f_i(x) \forall x \in SOL(i)$ (ovvero una soluzione per la quale la *funzione obiettivo* sia massima [minima] relativamente all'istanza data).

Diamo velocemente concretezza ai concetti esposti con un esempio banale.

Problema 1.2 *Massimo di Due Numeri Naturali*

$$\begin{aligned} \max \quad & x \\ x \in \quad & \{a, b\} \subset N \end{aligned}$$

Si tratta di massimizzare la funzione identità su una coppia di numeri naturali. Al variare di a e b in N si realizzano le diverse istanze del Problema 1.2 (es. $i_1 = \{1, 2\}$, $i_2 = \{3, 7\}$; $I = \{y \in 2^N \mid |y| = 2\}$ dove 2^N è la famiglia formata dai sottoinsiemi di N).

L'istanza i deve essere fornita alla macchina che esegue l'algoritmo A codificata in modo opportuno: indicheremo con $\langle i \rangle$ tale codifica e assumeremo per i numeri una rappresentazione *binaria* (nell'esempio $\langle i_1 \rangle = \langle 1, 10 \rangle$ e $\langle i_2 \rangle = \langle 11, 111 \rangle$). Tale rappresentazione è quella di fatto utilizzata nei reali elaboratori ma non è l'unica possibile: in alternativa ha rilevanza teorica la rappresentazione *unaria* (nell'esempio $\langle i_1 \rangle = \langle |, || \rangle$ e $\langle i_2 \rangle = \langle |||, ||||| \rangle$).

Se indichiamo con $dim\langle i \rangle$ la lunghezza dell'input costituito dall'istanza i otteniamo, sempre per il semplice esempio di riferimento, $dim^b\langle i_1 \rangle = 4$ e $dim^b\langle i_2 \rangle = 6$ se si è scelta la codifica binaria oppure $dim^u\langle i_1 \rangle = 4$ e $dim^u\langle i_2 \rangle = 11$ se si è scelta la codifica unaria. Per il generico numero $n \in N$ si ha $dim^b\langle n \rangle = \lceil \log(n+1) \rceil$ e $dim^u\langle n \rangle = n$ per cui al crescere di n la rappresentazione unaria produce codifiche notevolmente più lunghe ed è praticamente inutilizzabile.

Un punto importante riguarda la scelta del *modello di calcolo*, ovvero della macchina M che deve eseguire l'algoritmo A e quindi della descrizione che di tale algoritmo si deve dare. Fatto salvo che il riferimento ultimo sono sempre gli elaboratori reali—e talvolta ha più importanza il comportamento empirico di un algoritmo che non la sua complessità teorica che ci accingiamo a definire—sono presenti in letteratura diversi modelli di calcolo, un'ampia rassegna dei quali può essere trovata in Ausiello et al. [9, cap. 1]. Ognuno di questi modelli prevede delle *celle* per la memorizzazione dei dati (iniziali e intermedi) e un *set di istruzioni* che la macchina M è in grado di eseguire (e con le quali si deve comporre A).

Misureremo, venendo al dunque, la complessità computazionale di un algoritmo A per la macchina M con la quantità di una data *risorsa* R che M

richiede per eseguirlo. Anche se questo approccio permette la massima generalità viene di fatto sempre considerata la risorsa *tempo*, intesa come somma dei tempi necessari a M per eseguire le singole istruzioni di A . Un'alternativa è di considerare la risorsa *spazio*, intesa come massimo numero di celle utilizzate nel corso dell'elaborazione, ma tale risorsa è nella pratica meno pregiata e perciò la complessità spaziale viene di solito ignorata.

Una volta stabiliti i tempi di esecuzione per le istruzioni di M , cosa che rientra nella definizione del modello di calcolo, risulta di conseguenza fissato il tempo T_i che M impiega per eseguire A su una data istanza i del problema assegnato. Tale tempo T_i varia però in generale al variare dell'istanza i e perciò si sceglie di classificare gli input secondo la loro dimensione, poiché è ragionevole attendersi una complessità crescente con essa, e per ogni fissata dimensione si considera l'istanza che richiede più tempo, assumendo così un punto di vista prudente. Quello che si ottiene è una complessità temporale $T(m) = \max_{dim(i)=m} T_i$ che è funzione della dimensione dei dati in ingresso e dipende evidentemente dal tipo di codifica utilizzata (binaria o unaria).

Continuando l'esempio del Problema 1.2 esso può essere immediatamente risolto da un automa M che disponga di un'istruzione per confrontare tra loro due numeri naturali a e b . Se $0.5 \min\{\lceil \log(a+1) \rceil, \lceil \log(b+1) \rceil\} \mu s$ è il tempo richiesto da tale istruzione (confronto bit a bit in rappresentazione binaria) allora risulta essere $T(m) = 0.5 \lfloor (m-1)/2 \rfloor$ la complessità di A (che consta di una sola istruzione) poiché il caso peggiore è quello in cui $a = b$.

I fattori costanti ed eventualmente i termini additivi presenti nell'espressione della complessità computazionale (peraltro dipendenti dall'unità di misura del tempo) non sono di grande interesse da un punto di vista teorico e perciò si introduce la cosiddetta *analisi asintotica*.

Date due funzioni numeriche $f(m)$ e $g(m)$ definite su N diremo che

$$f(m) \in O(g(m))$$

se esistono una costante positiva c e un intero m_0 tali che sia $f(m) \leq c g(m)$ per ogni $m \geq m_0$ (f cresce al più come g). D'altra parte diremo che

$$f(m) \in \Omega(g(m))$$

se esistono una costante positiva c e un intero m_0 tali che $f(m) \geq c g(m)$ per ogni $m \geq m_0$ (f cresce almeno quanto g). Scriveremo infine sinteticamente

$$f(m) \in \Theta(g(m))$$

quando si verifichino contemporaneamente le due eventualità sopra descritte (f cresce allo stesso modo di g).

Per l'esempio del Massimo tra Due Numeri Naturali possiamo scrivere $T(m) \in O(m)$ e rendere così conto sinteticamente delle prestazioni di A (singola istruzione di confronto). Potremmo anche scrivere $T(m) \in O(m^2)$ o $T(m) \in O(2^m)$ ma, poiché stiamo dando una delimitazione superiore, essa è chiaramente tanto più significativa quanto più è stringente.

Si osservi come l'analisi asintotica trascuri un numero finito (ma arbitrariamente grande) di dimensioni dell'input centrando l'attenzione sulla crescita della complessità per le istanze più grandi, quelle per le quali un comportamento efficiente è maggiormente importante.

Un caso particolarmente significativo è quello in cui si riesce a ottenere una limitazione di tipo polinomiale ($T(m) \in O(p(m))$) per la complessità di un algoritmo (nell'esempio $p(m) = m$). Disporre di un algoritmo polinomiale per un dato problema rende tale problema *trattabile*, nel senso che risulta possibile risolverne istanze di grandi dimensioni in tempo ragionevole.

Poiché $T(m)$ dipende dal tipo di codifica utilizzata anche la possibilità di individuare per essa una limitazione polinomiale dipende da tale codifica e perciò si possono distinguere per gli algoritmi tre tipi di polinomialità:

- *Forte Polinomialità*: algoritmi polinomiali sia in codifica binaria che in codifica unaria;
- *Debole Polinomialità*: algoritmi polinomiali in codifica binaria ma non in codifica unaria;
- *Pseudo-Polinomialità*: algoritmi polinomiali in codifica unaria ma non in codifica binaria.

Dal momento che in pratica si fa uso della codifica binaria le tre nozioni di polinomialità fornite sono elencate in ordine decrescente di importanza.

Per quanto riguarda il modello di calcolo, che finora è stato lasciato indefinito, diciamo ora che descriveremo gli algoritmi in linguaggio naturale. Questa scelta, che è molto comoda e permette un livello di astrazione elevato, necessita però di precisare quale sia il tempo richiesto dalle varie operazioni. A questo proposito si usa considerare di costo unitario le operazioni immediatamente disponibili in un linguaggio di programmazione ad alto livello come il Pascal (per una descrizione del quale si rimanda a Welsh ed Elder [54]). Un'analisi più accurata considererebbe di costo $\log n$ un'operazione sul numero n , senza per questo cambiare la sostanza dei risultati ottenuti.

Una nozione importante per l'analisi di algoritmi complessi è quella di *operazione dominante*: se il tempo $\tau(m)$ che viene speso eseguendo una certa

operazione è tale che $T(m) \in O(\tau(m))$ allora tale operazione si dice dominante. È chiaro che per determinare la complessità asintotica di un algoritmo è sufficiente individuarne un'operazione dominante.

1.1.1 Algoritmi Approssimati

A volte può succedere che non si riesca a sviluppare un algoritmo polinomiale per la risoluzione di un determinato problema (si veda a questo proposito quanto esposto nella sezione 1.2) e allora l'unico modo per affrontare istanze di grandi dimensioni è accettare un certo grado di approssimazione.

Un *algoritmo approssimato* per il Problema 1.1 riceve come input un'istanza i di tale problema e fornisce come output una soluzione $x^a \in SOL(i)$ che possa essere considerata quasi-ottima per l'istanza i . Di solito la nozione di quasi-ottimalità viene definita adottando un approccio, che si descrive di seguito, detto della *prestazione garantita*.

Se si definisce l'errore relativo di approssimazione come

$$e_r = \frac{|f(x^*) - f(x^a)|}{\max\{f(x^*), f(x^a)\}}$$

si può dire che un algoritmo è ϵ -approssimato quando garantisce che sia $e_r \leq \epsilon$ qualunque istanza i riceva in input. L'errore relativo varia tra zero (soluzione esatta) e uno (soluzione ammissibile qualsiasi) e perciò si prende $\epsilon \in [0, 1]$.

Per esempio un algoritmo $\frac{1}{3}$ -approssimato per un problema di minimo fornisce una soluzione x^a tale che sia $f(x^a) \leq \frac{3}{2}f(x^*)$; se tale approssimazione è accettabile nel contesto applicativo in cui l'algoritmo viene usato esso può tranquillamente sostituire un algoritmo esatto più lento (si rinuncia in qualche modo all'efficacia per ottenere maggiore efficienza).

In alcuni casi è possibile garantire un errore relativo massimo (piccolo) a piacere mediante una famiglia di algoritmi parametrici rispetto al valore di ϵ : chiameremo una tale famiglia *schema di approssimazione*. Uno schema di approssimazione che, fissato ϵ , abbia complessità polinomiale nella dimensione m dell'input viene detto PTAS (Polynomial Time Approximation Scheme): esempi di tali complessità possono essere $O(m^{1/\epsilon})$ oppure $O(m^3/\epsilon)$. La prima delle limitazioni sopra indicate cresce però esponenzialmente al decrescere di ϵ (cioè al crescere della precisione richiesta) mentre la seconda è polinomiale anche in $1/\epsilon$: distingueremo allora il secondo schema di approssimazione dal primo definendolo *pienamente polinomiale* (FPTAS=Fully Polynomial Time Approximation Scheme).

È chiaro che quando si cerca di risolvere un problema di ottimo in modo approssimato il migliore risultato che si può ottenere è un FPTAS; se non

risulta possibile sviluppare un FPTAS ci si può accontentare di un PTAS o, al limite, di un algoritmo ϵ -approssimato per uno specifico valore di ϵ .

Nella sezione 2.2 verrà introdotta un'altra nozione di approssimazione, specifica per problemi con vincoli di bilancio, che si basa sul concetto di quasi-ammissibilità invece che su quello di quasi-ottimalità.

1.2 Complessità di Problemi

Per caratterizzare la complessità del Problema 1.1 (Generica Ottimizzazione), piuttosto che quella di un particolare algoritmo che lo risolva, si usa considerare la sua cosiddetta *versione di decisione* (Problema 1.3).

Problema 1.3 *Generica Decisione*

Dato $k \geq 0$ esiste $x \in SOL$ tale che sia $f(x) \geq [\leq] k$?

La scelta di riferirsi a problemi in forma di decisione è motivata dal fatto che tale forma può essere utilizzata in diversi contesti, non solo nell'ambito dell'ottimizzazione, e inoltre la produzione dell'output (Si/No) è in questo caso di costo trascurabile e quindi l'attenzione è focalizzata sulla risoluzione vera e propria. D'altra parte è evidente che risolvere il problema di ottimo permette di rispondere al relativo problema di decisione e quindi che quest'ultimo non è più difficile del primo. Inoltre è spesso possibile risolvere il problema di ottimo mediante la risoluzione di un numero polinomiale di problemi di decisione (si veda per un esempio Ahuja et al. [2, sez. B.2]).

Un algoritmo per la risoluzione del Problema 1.3 (Generica Decisione) prende in ingresso una sua istanza i —che individua $k(i)$, f_i e $SOL(i)$ —e fornisce in uscita la risposta corrispondente $risp(i) \in \{Si, No\}$. Particolarmente adatta a descrivere un tale algoritmo risulta essere la Macchina di Turing (per il cui funzionamento si può vedere Ausiello et al. [9, pag. 51]) e perciò nel seguito ci riferiremo—come è prassi—a tale modello di calcolo. Vale comunque la pena osservare che la scelta della Macchina di Turing come modello di calcolo non è affatto limitativa perché secondo la Tesi di Church, oggi ampiamente accettata—cfr. Batini et al. [11, pag. 338], essa ha la massima *potenza espressiva* possibile, che è poi quella della maggior parte dei modelli ideati (Macchine a Registri e Pascal, per esempio). Inoltre è possibile realizzare interpreti per tali modelli alternativi che permettano a una Macchina di Turing di eseguirne le istruzioni in tempo polinomiale.

Diremo che un problema di decisione fa parte della *classe* P quando esiste una Macchina di Turing in grado di rispondere al suo quesito in tempo polinomiale (P sta, ovviamente, per Polinomiale).

Introduciamo ora la nozione di *algoritmo non deterministico*: si tratta di un algoritmo che prevede istruzioni in grado di “indovinare” l’elemento “giusto” di un insieme finito. Si possono poi immaginare tutte le scelte di questo tipo effettuate all’inizio e codificate in un *certificato* σ da affiancare all’istanza i per un’elaborazione deterministica.

Diremo che un problema di decisione fa parte della *classe NP* quando esiste una Macchina di Turing Non Deterministica in grado di rispondere al suo quesito in tempo polinomiale almeno nel caso in cui la risposta sia affermativa (*NP* sta, meno ovviamente, per Non deterministico Polinomiale).

Poiché esiste il caso banale di certificato nullo ($\sigma = \emptyset$) risulta evidentemente $P \subseteq NP$ mentre non è noto se sia vero il viceversa e, anzi, il resto di questa sezione è dedicato a chiarire perché si sia portati a credere che l’inclusione sia stretta (e quindi che gli algoritmi non deterministici siano in qualche modo più “potenti” di quelli deterministici che la mancanza di prescienza ci costringe a utilizzare nella pratica).

Diciamo che possiamo ridurre polinomialmente (secondo Karp) il problema Π al problema Q (e scriviamo $\Pi \leq Q$) se abbiamo una Macchina di Turing (deterministica) M la quale per ogni istanza i_π di Π che riceve in ingresso produce come output, in tempo polinomiale, un’istanza i_q di Q tale che $resp(i_q) = resp(i_\pi)$. Se $\Pi \leq Q$ ed è noto un algoritmo polinomiale per Q esso può essere combinato con M per risolvere anche Π in tempo polinomiale.

Un problema $Q \in NP$ si definisce *NP-completo* se $\forall \Pi \in NP$ è $\Pi \leq Q$: un problema NP-completo è al massimo grado di difficoltà possibile in *NP* (che contiene quasi tutti i problemi di interesse pratico) e se si trovasse un algoritmo polinomiale per un tale problema sarebbe chiaramente $P = NP$.

Nel 1971 Cook [18] ha introdotto la nozione di NP-completezza e ha dimostrato essere NP-completo un problema di logica noto col nome di *Soddisfacibilità*; successivamente è stato mostrato che sono NP-completi molti altri problemi, tra i quali la *Bisaccia 0-1* che verrà presentata nella sezione 2.2. Per una panoramica più vasta sui problemi NP-completi si rimanda a Garey e Johnson [27], con l’avvertenza che il numero di questi è in continua crescita perché per dimostrare che un problema di NP è NP-completo è sufficiente prendere un’altro problema la cui NP-completezza sia nota e ridurlo polinomialmente al nuovo problema (cfr. sez. 3.2).

La comunità scientifica non è ancora riuscita né a trovare un algoritmo polinomiale per un problema NP-completo (nel qual caso sarebbe $P = NP$) né a provare che un tale algoritmo non esista (ovvero che $P \neq NP$); visti i molti sforzi profusi nella ricerca di un tale algoritmo e la prevedibile difficoltà di una dimostrazione negativa molti oggi propendono per la seconda ipotesi.

La nozione di riducibilità polinomiale dipende dalla codifica adottata per i numeri: conseguentemente Q si dirà *debolmente* NP-completo se risulta NP-

completo con la usuale codifica binaria mentre potrà dirsi *fortemente* NP-completo solo se rimane NP-completo anche passando alla codifica unaria.

Quando la versione di decisione di un problema di ottimo è NP-completa esso si dice *NP-difficile*; l'unica via ragionevole per risolvere un problema NP-difficile in tempo polinomiale è sviluppare un algoritmo approssimato.

Capitolo 2

Miglioramento di una Rete

Questo capitolo è dedicato a definire i termini nei quali verrà studiato il problema di pianificare interventi migliorativi su una rete di comunicazione.

La prima sezione descrive come si possa modellare la rete e come possano essere misurate le sue prestazioni (relativamente all'uso che ne viene fatto).

La seconda sezione si occupa invece di modellare la possibilità di intervento sulla rete e in particolare le limitazioni cui essa è soggetta.

2.1 Modelli di Rete

La *rete* di comunicazione cui si farà riferimento nel seguito potrà essere formata da elaboratori collegati tramite fibre ottiche come da paesi interconnessi da strade o stazioni collegate da binari: in ogni caso al livello di astrazione prescelto la rappresenteremo come un *grafo etichettato*.

Un *grafo* $G = (V, A)$ è individuato da un insieme di nodi V (l'insieme degli elaboratori, dei paesi o delle stazioni) e da un insieme di archi A (le fibre ottiche, le strade o le tratte di binari); se il grafo è orientato (collegamenti monodirezionali) indicheremo l'arco dal nodo i al nodo j con la coppia ordinata (i, j) mentre se non c'è orientamento l'arco tra i e j sarà rappresentato dalla coppia non ordinata $ij = \{i, j\}$. La Figura 2.1 mostra un grafo di esempio (non orientato) nella classica rappresentazione grafica con cerchi per i nodi e linee per gli archi.

Un *multigrafo* è un grafo in cui possono essere presenti più *archi paralleli* tra due nodi e/o archi aventi i due estremi coincidenti (*cappi*). Spesso un problema di ottimo formulato su un multigrafo non è più difficile del suo analogo formulato su un grafo *semplice* ma richiede soltanto una notazione più complessa.

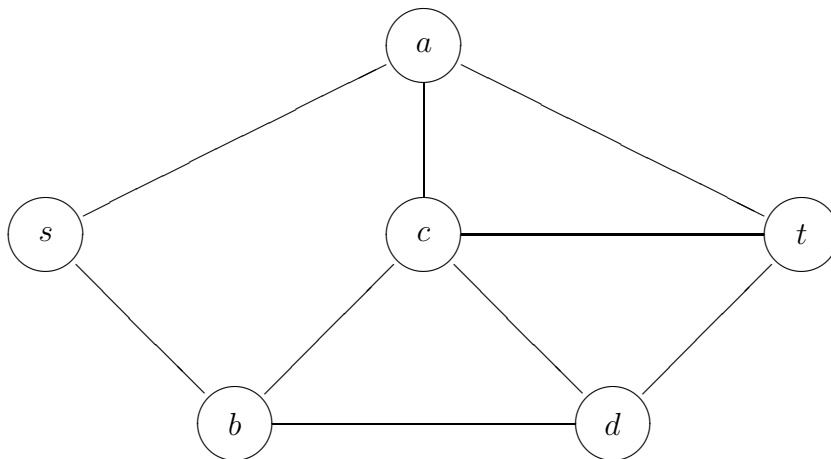


Figura 2.1: Una piccola parte della rete stradale toscana.

Per quanto riguarda le *etichette* si tratta in generale di funzioni a valori interi definite su A (o talvolta su V). Per la verità le etichette potrebbero tranquillamente essere a valori reali ma poiché ci occupiamo di algoritmi per computer i dati del problema devono essere rappresentati in modo finito e quindi vanno considerati i soli numeri razionali. L'approssimazione dei reali con i razionali è tradizionalmente studiata, assieme alla propagazione dell'errore sui dati, dal Calcolo Numerico (si veda per es. Gambolati [26, cap. 1]). Inoltre è generalmente possibile rendere interi i dati razionali moltiplicandoli per un numero abbastanza grande (il loro Minimo Comune Denominatore) come suggerito da Ahuja et al. [2, *integrality assumption* pag. 6]. La Tabella 2.1 elenca delle possibili etichette per gli archi del grafo di Figura 2.1: tempi di percorrenza e probabilità di funzionamento in caso di neve.

Data una rete $G = (V, A; t, p)$ come quella dell'esempio appena visto esistono diversi modi per rappresentarla su un computer e dalla scelta della rappresentazione dipende la complessità degli algoritmi che sulla rete operano. Per il tipo di problemi che noi considereremo (*problemi di flusso*, secondo Ahuja et al. [2, sez. 1.2]) risulta però in generale più conveniente la rappresentazione tramite *liste di adiacenza* e perciò ci limiteremo a considerare questa, rinviando ad Ahuja et al. [2, sez. 2.3] per una trattazione più completa. Rappresentare una rete tramite liste di adiacenza vuol dire semplicemente memorizzare le liste degli archi uscenti da ogni nodo, con ogni arco debitamente etichettato: così—Figura 2.2—si tiene conto sia della topologia

| ij | $t_{ij}(min)$ | $p_{ij}(\%)$ |
|------|---------------|--------------|
| sa | 14 | 75 |
| sb | 16 | 55 |
| ac | 15 | 75 |
| at | 24 | 90 |
| bc | 6 | 75 |
| bd | 13 | 85 |
| cd | 9 | 80 |
| ct | 10 | 85 |
| dt | 12 | 60 |

Tabella 2.1: Etichette di esempio per il grafo di Figura 2.1.

del grafo $G = (V, A)$ che dei valori numerici di t e p .

Per quanto riguarda la dimensione dell'input essa è proporzionale ai parametri $n = |V|$, $m = |A|$ e $\log D$ (dove D è il massimo valore assoluto di un'etichetta e il logaritmo tiene conto della rappresentazione binaria dei numeri nei calcolatori); è proprio in funzione di questi parametri che esprimeremo la complessità computazionale degli algoritmi, come è del resto prassi fare per semplicità e chiarezza (cfr. Ahuja et al. [2, *problem size* pag. 57]).

Per un'introduzione teorica ai grafi si rinvia a Cattaneo Gasparini [14, cap. 4] mentre un'approccio più informatico può essere trovato in Ahuja et al. [2, cap. 2]: qui si richiamano solo alcune nozioni considerate fondamentali per la trattazione svolta.

Un *cammino* P nel grafo $G = (V, A)$ è una successione (v_0, v_1, \dots, v_k) di nodi distinti in V tali che sia $\{v_i, v_{i+1}\} \in A$; in un grafo orientato questo significa che gli archi possono essere percorsi nel verso "sbagliato": se ciò non accade il cammino si dice *orientato*. Un *ciclo* (orientato) è un cammino (orientato) in cui $v_0 = v_k$: diremo *aciclico* un grafo che non contenga cicli.

D'altra parte si dirà che un grafo G è *connesso* se comunque si prendano due suoi nodi u e z esiste (almeno un) cammino $P = (u, \dots, z)$ in G che li congiunge; per i problemi di flusso su reti che verranno studiati si può, senza perdita di generalità, considerare G connesso. Un grafo connesso aciclico è detto *albero*. In un grafo connesso si ha $m \geq n - 1$ (l'uguaglianza caratterizza il caso dell'albero) mentre è vero per qualsiasi grafo che $m \in O(n^2)$ (se il grafo ha tutti i possibili archi si dice *completo*).

Un grafo $G' = (V', A')$ si dice *sotto-grafo* del grafo $G = (V, A)$ se si ha $V' \subseteq V$ e $A' \subseteq A$: scriveremo in tal caso $G' \leq G$.

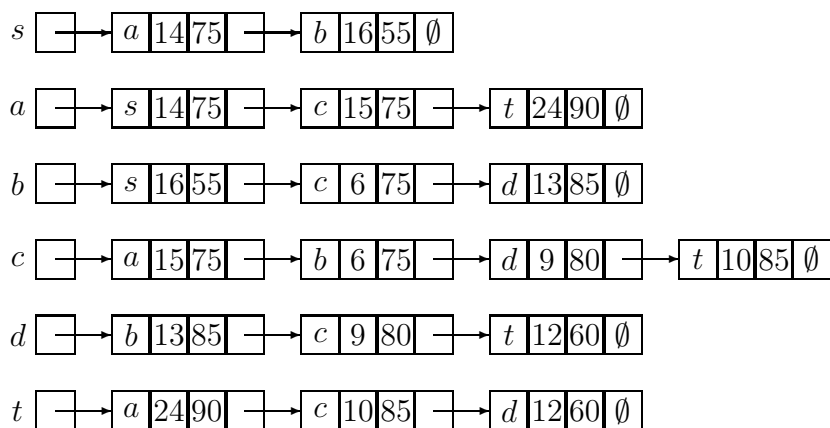


Figura 2.2: Rappresentazione tramite liste di adiacenza della rete esemplificata da Figura 2.1 e Tabella 2.1.

Una particolare classe di grafi, di uso frequente nelle applicazioni, è quella dei *grafi planari*: un grafo si dice planare quando può essere rappresentato su un piano euclideo in modo che i suoi archi si intersechino solo nei nodi. Per i grafi planari—come quello di Figura 2.1—si ha $m < 3n$ (si veda per es. Ahuja et al. [2, cap. 8]) e perciò tali grafi sono molto *sparsi*, ovvero hanno pochissimi archi rispetto al numero di nodi. Per questo gli algoritmi che risolvono problemi su grafi risultano in generale più efficienti quando questi sono planari; inoltre è talvolta possibile sfruttare in modo esplicito la topologia planare (un esempio viene trattato nella sezione 4.2).

Vediamo ora come la topologia e le etichette di una rete possano determinare le sue prestazioni in funzione di *chi* usa la rete e di *come* la usa. A rigore il *problema di ottimo* per la pianificazione di interventi migliorativi su una rete nasce solo dopo che si è definita una misura per le prestazioni di questa; perciò la figura del *decisore* degli investimenti, rispetto al quale l'ottimo viene cercato, sembra essere logicamente successiva a tale misura. In realtà anche nella scelta di una misura di efficienza per G è necessario, come vedremo subito, assumere un ben preciso punto di vista.

Con riferimento alla rete di Figura 2.1 e Tabella 2.1 supponiamo in un primo tempo vi sia un automobilista che vuole recarsi da s a t nel minore tempo possibile: egli può senza dubbio misurare l'efficienza di G per mezzo di un cammino minimo per tempo di percorrenza. Questo caso verrà studiato

nel Capitolo 3 dove si vedrà anche come il decisore possa avere interesse ad aumentare invece che diminuire tale misura.

In alternativa, sempre per la rete di Figura 2.1 e Tabella 2.1, il decisore può essere interessato a che tutte le località restino connesse anche in caso di neve e allora l'efficienza di G si può misurare con un albero ricoprente massimo per probabilità di funzionamento. Questo caso viene trattato nel Capitolo 5 mentre nel Capitolo 4 si affronta l'ulteriore caso del Flusso Massimo.

È chiaro che le due misure presentate presuppongono diversi punti di vista, se non altro perché nel primo caso i nodi s e t godono di una posizione "privilegiata" mentre nel secondo tutti i nodi hanno pari importanza.

2.2 Modelli di Decisione

Supponiamo data una rete G con una misura $mis(G)$ della sua efficienza, secondo quanto visto nella sezione precedente, e modelliamo da un punto di vista generale la possibilità di intervenire su di essa affermando che può trovarsi in diversi *stati*: l'efficienza di G nello stato Σ sarà $mis(G, \Sigma)$.

Originariamente la rete si trova nello stato Σ_0 , le cui prestazioni non sono soddisfacenti, ma tramite interventi migliorativi ancora da definire è possibile far variare Σ in un insieme di stati ammissibili che chiameremo *spazio degli stati*. Descriveremo tale spazio degli stati per mezzo di *variabili di decisione* che saranno *continue* o *discrete* a seconda dei casi e in ogni caso logicamente distinte da quelle che individuano $mis(G)$ (per esempio quali archi appartengano a un cammino minimo). L'introduzione del concetto di stato serve proprio a sottolineare questa distinzione logica, oltre che a fornire un modello molto generale.

Portare G nello stato Σ , cioè effettuare gli interventi migliorativi corrispondenti, è un'operazione che presenta un costo $c(\Sigma)$ ($c(\Sigma_0) = 0$ poiché si suppone che la rete si trovi già nel suo stato iniziale). Solo considerando le coppie $(mis(G, \Sigma), c(\Sigma))$ si può allora individuare uno stato Σ^{OT} che, secondo un qualche criterio, sia ottimo dal punto di vista della spesa come da quello delle prestazioni.

Come criterio di ottimalità adotteremo senza eccezioni la massima efficienza compatibile con un *budget* B supposto a disposizione del decisore e perciò i problemi affrontati avranno tutti la forma che segue.

Problema 2.1 *Miglioramento Ottimo di una Rete*

$$\begin{array}{ll} \max [\min] & mis(G, \Sigma) \\ & c(\Sigma) \leq B \end{array}$$

La formulazione è chiaramente influenzata dal fatto che alta efficienza corrisponda a $mis(G)$ grande [piccola].

Questo non è l'unico criterio possibile, ovviamente, ma appare senz'altro di grande realismo poiché gli investimenti avvengono quasi sempre sotto un *vincolo di bilancio*. Tale vincolo caratterizza il noto problema della *Bisaccia*: un viaggiatore deve scegliere fra n oggetti quali mettere nella sua borsa di capienza B basandosi sul profitto p_i che egli trae dall'avere con se l' i -esimo oggetto e sull'ingombro c_i del medesimo.

Problema 2.2 *Bisaccia 0-1*

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^n p_i y_i \\ & \sum_{i=1}^n c_i y_i \leq B \\ & y_i \in \{0, 1\} \end{aligned}$$

In questa prima versione della *Bisaccia* gli oggetti sono indivisibili e questo rende il problema (debolmente) NP-difficile; se invece gli oggetti sono divisibili a piacere è possibile formulare il problema direttamente in termini di ingombri unitari $\gamma_i = c_i/p_i$.

Problema 2.3 *Bisaccia Continua*

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^n z_i \\ & \sum_{i=1}^n \gamma_i z_i \leq B \\ & 0 \leq z_i \leq p_i \end{aligned}$$

Questa seconda versione della *Bisaccia* è risolvibile in tempo polinomiale con un algoritmo di tipo *greedy* che scelga gli oggetti in ordine crescente di ingombro unitario (sino all'esaurimento dello spazio B). Algoritmi risolutivi per le due versioni della *Bisaccia* possono essere trovati, per esempio, in Ausiello et al. [9].

Per il Problema 2.1 (Miglioramento Ottimo di una Rete) è possibile considerare un approccio algoritmico approssimato diverso da quello presentato nella sezione 1.1; tale approccio, che può definirsi ad *ammissibilità garantita*, ha senso per un qualsiasi problema di ottimo NP-difficile con un vincolo di bilancio. Definito l'errore relativo di ammissibilità

$$e'_r = \max \left\{ 0, \frac{c(\Sigma^A) - B}{c(\Sigma^A)} \right\}$$

una soluzione Σ^A si dirà ottimale ϵ -ammissibile se $e'_r \leq \epsilon$ e inoltre

$$mis(G, \Sigma^A) \geq [\leq] mis(G, \Sigma^{OT}).$$

È anche possibile la combinazione dei due approcci in un algoritmo che fornisca una soluzione Σ^A ϵ -ammissibile e δ -approssimata, ovvero tale che $c(\Sigma^A) \leq B/(1 - \epsilon)$ e $mis(G, \Sigma^A) \geq mis(G, \Sigma^{OT})/(1 - \delta)$ (analogamente per il caso della minimizzazione—cfr. cap. 5).

Una possibile alternativa al vincolo di bilancio può essere la massimizzazione (o minimizzazione) del rapporto

$$\frac{mis(G, \Sigma)}{c(\Sigma)}$$

tra le prestazioni e il loro costo. Cunningham [19] fa uso di questo criterio al fine di sconnettere i nodi di una rete (e la sua definizione di *strenght*(G) tornerà utile nella sezione 5.4) mentre Chandrasekaran [15] studia il problema di trovare un albero ricoprente che sia minimo rispetto al rapporto tra due funzioni degli archi. Da un punto di vista teorico l'uso del rapporto prestazioni/costo è riduttivo rispetto all'uso di un vincolo di bilancio perché non tiene conto di interventi che possono avere costo smisurato o profitto insufficiente (questa è esattamente la ragione per cui un algoritmo *greedy* non risolve la *Bisaccia 0-1*).

Per quanto riguarda infine la caratterizzazione degli interventi migliorativi si fa l'ipotesi che non sia possibile modificare la topologia della rete ma si possa solo modificarne le etichette. Anche sotto questo aspetto esistono in letteratura approcci diversi: per esempio Wilkow [55] ha studiato, tra gli altri, il progetto di topologie invulnerabili mentre Ball et al. [10] hanno affrontato il problema di accrescere il cammino minimo di una rete eliminando alcuni dei suoi archi (similmente Frederickson e Solis-Oba [22] nel caso del minimo albero ricoprente).

Capitolo 3

Problemi di Cammino Minimo

In questo capitolo si considera il caso in cui l'efficienza della rete venga misurata, in positivo o in negativo, dalla lunghezza del cammino più breve tra un nodo sorgente e un nodo destinazione.

Nella prima sezione viene presentato il problema classico di calcolare il cammino minimo tra due nodi di una rete e successivamente vengono distinte una modalità di intervento discreta e una modalità di intervento continua.

La seconda e la terza sezione sono dedicate al problema di impiegare ottimamente i fondi a disposizione per diminuire la lunghezza del cammino minimo: nella seconda sezione il decisore sceglie per ogni arco tra un numero discreto di alternative mentre nella terza sezione le lunghezze degli archi possono variare con continuità da un valore massimo a un valore minimo.

La quarta sezione si occupa invece del caso in cui il decisore sia interessato ad aumentare la lunghezza del cammino minimo: in questo caso vengono considerate la modalità di intervento continua e una particolare modalità di intervento discreta.

Nella quinta sezione viene infine fatta una sintesi dei risultati presentati nel corso del capitolo, sottolineandone gli aspetti di originalità ed evidenziando i problemi che essi lasciano aperti.

3.1 Introduzione al Cammino Minimo

Data una rete orientata $G = (V, A; l)$, dove $l : A \rightarrow N \cup \{0\}$ è una funzione lunghezza per i suoi archi, definiamo la lunghezza di un cammino orientato $P = (v_o, v_1, \dots, v_k)$ in G come $l(P) = \sum_{i=1}^k l_{v_{i-1}v_i}$; fissati poi due nodi "speciali" s e t cerchiamo un cammino di lunghezza minima tra quelli che cominciano in s e finiscono in t . Indicheremo con l^* tale lunghezza minima e sarà quindi $mis(G) = l^*$ la misura di efficienza adottata in questo capitolo.

In un contesto più generale il problema del cammino minimo può essere formulato come problema di programmazione lineare. Esistono infatti due formulazioni—l'una la duale dell'altra—che si riportano qui di seguito.

Problema 3.1 *Flusso Unitario a Costo Minimo*

$$\begin{aligned} & \text{minimize} && \sum_{(i,j) \in A} l_{ij} x_{ij} \\ & \sum_{j:(v,j) \in A} x_{vj} - \sum_{i:(i,v) \in A} x_{iv} = && \begin{cases} +1 & v = s \\ 0 & v \in V - \{s, t\} \\ -1 & v = t \end{cases} \\ & && x_{ij} \geq 0 \end{aligned}$$

Per giustificare questa prima formulazione, nella quale le lunghezze assumono il significato di costi, si osservi che in base ai vincoli le variabili di decisione x_{ij} , una per ogni arco, individuano in G un *flusso* di valore unitario da s a t (cfr. cap. 4). La Proprietà di Integralità Primale per i problemi di flusso a costo minimo (Ahuja et al. [2, teo. 11.5]) garantisce l'esistenza di una soluzione ottima intera: in essa l'unità di flusso emessa da s non può dividersi lungo percorsi diversi e perciò individua un unico cammino—di lunghezza minima—fino a t .

Problema 3.2 *Massima Tensione*

$$\begin{aligned} & \text{maximize} && d(t) - d(s) \\ & d(j) - d(i) \leq && l_{ij} \quad \forall (i, j) \in A \end{aligned}$$

Questa seconda formulazione può essere giustificata rigorosamente come duale della prima (Ahuja et al. [2, sez. 9.4]) e anche qui, grazie alla Proprietà di Integralità Duale per problemi di flusso a costo minimo (Ahuja et al. [2, teo. 11.4]), è garantita l'esistenza di una soluzione ottima intera. Per una tale soluzione la differenza $d(t) - d(s)$ è la minima lunghezza di un cammino da s a t in G : avrà tale lunghezza un qualsiasi cammino formato da archi $(i, j) \in A$ per i quali sia $l_{ij} = d(j) - d(i)$.

È anche possibile pensare il problema del cammino minimo definito su un multigrafo ma poiché i cappi e gli archi paralleli più lunghi risultano del tutto inutili non si ottiene una situazione più generale.

Se il grafo del quale interessa il cammino minimo non è orientato esso può essere sostituito dal grafo che si ottiene rimpiazzando ogni suo arco $ij \in A$ di lunghezza l_{ij} con la coppia di archi orientati (i, j) e (j, i) entrambi di lunghezza l_{ij} .

Su un grafo non orientato è anche possibile interpretare in modo suggestivo la formulazione come problema di Massima Tensione. Se infatti si realizza fisicamente ogni arco ij con una cordicella anelastica lunga l_{ij} e si allontanano tra loro i nodi s e t quanto più è possibile allora si tenderanno alcune cordicelle mentre altre resteranno lente: un qualsiasi cammino tra s e t che consti solo di cordicelle tese (e solo un cammino che goda di questa proprietà) avrà lunghezza minima.

Oltre a questo metodo per così dire “analogico”, proposto da Klee [37] in una forma adatta anche ai grafi orientati, esistono diversi algoritmi in grado di risolvere il problema classico del cammino minimo. Di tutti il più famoso è senza dubbio quello di Dijkstra [21] la cui complessità temporale $O(n^2)$ è ottima per grafi *densi* ($m = \Omega(n^2)$) poiché qualsiasi algoritmo deve esaminare tutti gli archi. Quando lo si implementa con l’ausilio di uno *heap* di Fibonacci (struttura dati descritta per es. da Ahuja et al. [2, sez. A.4]) la sua complessità diviene $O(m + n \log n)$ (si veda Ahuja et al. [2, sez. 4.7]); questo risultato è il migliore conosciuto per un algoritmo fortemente polinomiale e verrà indicato nel seguito con $\Pi(n, m)$.

Se si prendono in considerazione anche algoritmi debolmente polinomiali la migliore limitazione che si conosca della complessità temporale diventa $\Pi(n, m, L) = \min\{m + n \log n, m \log \log L, m + n\sqrt{\log L}\}$ ($L = \max_{ij \in A} l_{ij}$). Del primo termine si è già detto, per il secondo si può vedere Johnson [36] mentre il terzo è stato ottenuto da Ahuja et al. [3] mediante una struttura dati chiamata *radix heap*; quale dei tre termini sia più stringente dipende dai dati del problema.

In alternativa è possibile ricorrere al metodo del simplesso nella sua versione specializzata per i problemi di flusso su reti (Ahuja et al. [2, cap. 11]). Tale metodo risulta teoricamente inferiore dal punto di vista del caso peggiore ma ha in pratica un comportamento eccellente (e peraltro funziona anche nel caso più generale in cui le lunghezze possano essere negative—caso che in questa sede non viene trattato).

È comunque da sottolineare che non aumenta la complessità computazionale del caso peggiore se si vogliono i cammini minimi da un nodo s a tutti gli altri nodi di G (perché da ognuno di essi, in generale, si può arrivare a t).

Le possibili applicazioni per il modello del cammino minimo sono sterminate e un buon compendio di quelle presenti in letteratura può essere trovato in Ahuja et al. [2]. L’esempio che segue vuole solo chiarire in che modo la lunghezza del cammino minimo possa misurare l’efficienza di una rete.

Si consideri una rete in cui il nodo s detiene una informazione riservata (esso può rappresentare, per esempio, un ufficio giudiziario o un gruppo di ricercatori di una società *hi-tech*) e il nodo t è interessato ad acquisire tale informazione (perché è un’agenzia di stampa o una ditta concorrente); gli altri

nodi della rete rappresentano altri soggetti tramite i quali l'informazione può arrivare da s a t ed è presente un arco da i a j con etichetta l_{ij} quando i comunica a j le informazioni che riceve dopo un tempo l_{ij} .

Nella situazione descritta l^* è chiaramente il tempo per il quale l'informazione in questione rimane segreta e quindi se si assume il punto di vista di s la rete è tanto più efficiente quanto più tale tempo è lungo, mentre dal punto di vista di t la rete è tanto più efficiente quanto più tale tempo è breve.

Distinguiamo ora due possibili modalità di intervento sulla rete:

1. Per ogni arco è dato un numero finito di possibilità (nessun intervento, intervento di tipo I, intervento di tipo II, ...). Chiameremo *discreta* questa modalità caratterizzata da uno spazio degli stati finito.
2. La lunghezza l_{ij} di ogni arco può essere ridotta in modo continuo dal valore iniziale Λ_{ij} sino a un valore minimo λ_{ij} . Chiameremo *continua* questa modalità caratterizzata da uno spazio degli stati infinito.

Il caso in cui per le lunghezze siano leciti i soli valori interi in $[\lambda_{ij}, \Lambda_{ij}]$ si può far rientrare nella prima modalità. Entrambe le modalità verranno riproposte nei due capitoli successivi per le misure di efficienza ivi considerate.

3.2 Diminuzione Discreta del Cammino Minimo

Sia data una rete orientata $G = (V, A; l^0, l^1, \dots, l^{S-1}; c^0, c^1, \dots, c^{S-1})$ nella quale $l_{ij}^k \in N \cup \{0\}$ è la lunghezza dell'arco $(i, j) \in A$ quando questo si trovi nello stato k e $c_{ij}^k \in N \cup \{0\}$ è il costo da sostenere per portarlo in tale stato.

Ordiniamo per chiarezza gli l_{ij}^k in ordine decrescente con k per ogni arco (i, j) e di conseguenza i c_{ij}^k cresceranno con k (poiché se $l_{ij}^h \leq l_{ij}^k$ e $c_{ij}^h \leq c_{ij}^k$ lo stato k può essere ignorato in quanto lo stato h è non meno vantaggioso sia come lunghezza che come costo); in particolare $k = 0$ individuerà lo stato iniziale e avremo perciò $c_{ij}^0 = 0 \quad \forall (i, j) \in A$.

Introduciamo poi le variabili di decisione $y_{ij} \in \{0, 1, \dots, S_{ij} - 1\}$ (dove $S_{ij} \leq S$ è il numero di stati effettivamente disponibili per l'arco (i, j)) con $y_{ij} = k$ a significare che si pone l'arco (i, j) nello stato k : l'insieme $\{y_{ij}\}_{(i,j) \in A}$ di tali variabili individuerà lo stato complessivo Σ della rete G .

Il vincolo di bilancio, che caratterizza gli stati ammissibili, si scrive allora $\sum_{(i,j) \in A} c_{ij}^{y_{ij}} \leq B$ e l'obiettivo è trovare un stato ammissibile Σ^{OT} tale che $l^*(\Sigma^{OT})$ sia minima. Possiamo allora formulare il problema introdotto sul modello del Problema 3.1 (Flusso Unitario a Costo Minimo).

Problema 3.3 *Diminuzione Discreta del Cammino Minimo*

$$\begin{aligned}
 & \text{minimize} && \sum_{(i,j) \in A} l_{ij}^{y_{ij}} x_{ij} \\
 & \sum_{j:(v,j) \in A} x_{vj} - \sum_{i:(i,v) \in A} x_{iv} = && \begin{cases} +1 & v = s \\ 0 & v \in V - \{s, t\} \\ -1 & v = t \end{cases} \\
 & \sum_{(i,j) \in A} c_{ij}^{y_{ij}} \leq && B \\
 & x_{ij} \in && \{0, 1\} \\
 & y_{ij} \in && \{0, 1, \dots, S_{ij} - 1\}
 \end{aligned}$$

Se la rete G non è orientata ogni suo arco ij rappresenta la coppia di archi orientati (i, j) e (j, i) : tali archi non possono assumere stati diversi tra loro e quindi per essi si dovrebbe considerare una sola variabile di stato $y_{ij} = y_{ji}$. In pratica conviene però richiedere che sia $y_{ij} = 0$ o $y_{ji} = 0$ in modo che non vi siano ripercussioni sul vincolo di bilancio (resta sottintesa l'uguaglianza al valore più elevato); questo è possibile perché le lunghezze sono non negative e quindi $x_{ij} = 0$ o $x_{ji} = 0$ senza perdita di generalità.

Se si interpretano gli archi come strade sopraelevate, le loro lunghezze come il tempo necessario a ripristinarle dopo un sisma e gli interventi migliorativi come lavori per rinforzare le strutture si ottiene un problema suggerito da Augusti e Ciampoli [8] in un contesto fortemente applicativo.

3.2.1 Complessità Computazionale

Consideriamo una versione semplificata del Problema 3.3, caratterizzata dal fatto che per ogni arco è possibile lasciarlo inalterato oppure intervenire in un solo modo ($S_{ij} = S = 2 \ \forall (i, j) \in A$).

Problema 3.4 *Diminuzione Discreta Semplificata del Cammino Minimo*

$$\begin{aligned}
 & \text{minimize} && \sum_{(i,j) \in A} [l_{ij}^0(1 - y_{ij}) + l_{ij}^1 y_{ij}] x_{ij} \\
 & \sum_{j:(v,j) \in A} x_{vj} - \sum_{i:(i,v) \in A} x_{iv} = && \begin{cases} +1 & v = s \\ 0 & v \in V - \{s, t\} \\ -1 & v = t \end{cases} \\
 & \sum_{(i,j) \in A} c_{ij}^1 y_{ij} \leq && B \\
 & x_{ij} \in && \{0, 1\} \\
 & y_{ij} \in && \{0, 1\}
 \end{aligned}$$

L'obiettivo consiste, da un punto di vista relativo rispetto allo stato iniziale, nel cercare la massima diminuzione possibile per il cammino minimo e quindi siamo di fronte a qualcosa di molto simile al classico problema combinatorio della *Bisaccia 0-1*: dobbiamo infatti scegliere quali interventi mettere nella bisaccia (cioè eseguire) sulla base dei loro costi c_{ij}^1 e del profitto che se ne trae per la diminuzione del cammino minimo.

Tuttavia il profitto che deriva da un singolo intervento non può in generale essere quantificato a priori con un numero p_{ij} perché dipende dagli altri interventi che gli si affiancheranno. Perciò il Problema 3.4 (e di conseguenza il Problema 3.3) può essere visto come una generalizzazione della *Bisaccia 0-1* e se questo non è di aiuto per la sua risoluzione permette tuttavia di caratterizzarne la complessità computazionale.

Teorema 3.1 *Il Problema 3.3 (Diminuzione Discreta del Cammino Minimo) è debolmente NP-difficile.*

Dimostrazione.

Dato un problema di *Bisaccia 0-1* (per la cui formulazione si rimanda alla sezione 2.2) si costruisca la seguente rete a topologia “rettilenea”:

$$\begin{aligned} V &= \{s = 0, 1, \dots, n = t\} \\ A &= \{(i-1, i) \mid i = 1, 2, \dots, n\} \\ l_{(i-1)i}^0 &= p_i \\ l_{(i-1)i}^1 &= 0 \\ c_{(i-1)i}^1 &= c_i \end{aligned}$$

Il Problema 3.3 definito su tale rete equivale alla *Bisaccia 0-1* di partenza poiché in questa particolare topologia vi è un unico cammino da s a t al quale appartengono tutti gli archi e perciò il profitto di ogni singolo intervento è quantificabile a priori come $l_{(i-1)i}^0 - l_{(i-1)i}^1 = p_i$. Dalla debole NP-difficoltà della *Bisaccia 0-1* segue perciò quella del Problema 3.3. \diamond

La dimostrazione appena svolta è interessante soprattutto perché in qualche modo evidenzia la struttura fondamentale dei problemi di natura discreta trattati in questo e nei due successivi capitoli.

3.2.2 Risoluzione Esatta

Come conseguenza del Teorema 3.1 non possiamo trovare un algoritmo polinomiale che risolva in modo esatto il Problema 3.3 (a meno che non sia $P = NP$): al più possiamo sperare in un algoritmo pseudo-polinomiale.

I due algoritmi mostrati nel seguito sono basati su tecniche di programmazione dinamica e sono effettivamente pseudo-polinomiali. Essi si ispirano a un lavoro di Phillips [46] sull'interdizione del flusso per reti planari: ciò è reso possibile dall'equivalenza che sussiste per tali reti tra problemi di flusso massimo e problemi di cammino minimo (Teorema 4.4).

Osserviamo preliminarmente che il Problema 3.3 (Diminuzione Discreta del Cammino Minimo) può anche essere formulato come ricerca di un cammino minimo in G da s a t con il vincolo che tale cammino non costi più di B (dove il costo di un cammino è la somma dei costi da sostenere per portare gli archi percorsi allo stato nel quale vengono percorsi). Facendo riferimento a questa formulazione costruiamo la seguente *rete di ricerca*:

$$\begin{aligned} G_r &= (V_r, A_r; l_r) \\ V_r &= V \times \{0, 1, \dots, B\} \\ A_r &= \{((i, c), (j, d)) \mid (i, j) \in A, \exists k \in \{0, 1, \dots, S_{ij} - 1\} : d = c + c_{ij}^k\} \\ l_r(ic, jd) &= l_{ij}^k, k : d = c + c_{ij}^k \end{aligned}$$

La rete di ricerca G_r è formata da coppie nodo-costo: essere in una di queste coppie (i, c) significa, con riferimento alla rete originaria G , essere nel nodo i avendo speso c ; andare da (i, c) a (j, d) vuol dire altresì andare da i a j spendendo la differenza $(d - c)$ che serve per potere utilizzare un arco di lunghezza $l_r(ic, jd)$. È allora chiaro che un cammino minimo in G_r da $(s, 0)$ a (t, c) corrisponde a un cammino minimo in G da s a t che abbia esattamente costo c . Possiamo allora dare l'Algoritmo 3.1 e il successivo Teorema 3.2 relativo alle sue prestazioni.

Algoritmo 3.1 Diminuzione Discreta del Cammino Minimo I

1. *Costruisci a partire da G la rete di ricerca G_r .*
2. *Trova i cammini minimi $P(c)$ per $c \in \{0, 1, \dots, B\}$ che vanno in G_r dal nodo $(s, 0)$ ai nodi (t, c) .*
3. *Ripeti per c che va da 1 a B :
se $l(P(c-1)) \leq l(P(c))$ poni $P(c) := P(c-1)$.*
4. *Determina Σ^{OT} portando gli archi di $P(B)$ allo stato nel quale vengono attraversati e lasciando tutti gli altri allo stato zero.*

Teorema 3.2 *L'Algoritmo 3.1 fornisce una soluzione ottima per il Problema 3.3 in tempo $O(\Pi(nB, mSB, L))$.*

Dimostrazione.

L'ottimalità della soluzione fornita segue dalle considerazioni fatte in precedenza: dopo l'esecuzione del passo 3 ogni cammino $P(c)$ è ottimo se il *budget* è pari a c . L'algoritmo funziona anche per grafi non orientati poiché ogni arco ij compare in $P(B)$ al più una volta e in un solo verso: sarà quindi sempre $y_{ij} = 0$ o $y_{ji} = 0$ come richiesto.

Per quanto riguarda la complessità computazionale è sufficiente osservare che la rete di ricerca G_r è formata da $n(B + 1)$ nodi e al più $mS(B + 1)$ archi: perciò la sua rappresentazione tramite liste di adiacenza può essere costruita in $O(mSB)$ a partire da quella di G e i cammini minimi da $(s, 0)$ a tutti gli altri nodi di G_r possono essere calcolati in $O(\Pi(nB, mSB, L))$ (cfr. sez. 3.1). Rendere ottimo il cammino $P(B)$ richiede un tempo $O(B)$ mentre la determinazione dello stato Σ^{OT} è $O(m)$ per cui risulta dominante la ricerca dei cammini minimi (passo 2). \diamond

Vediamo ora una variante dell'Algoritmo 3.1 nella quale, ai fini della programmazione dinamica, lunghezze e costi si scambiano di ruolo. Si tratta principalmente della definizione di una rete di ricerca alternativa:

$$\begin{aligned} G'_r &= (V'_r, A'_r; c'_r) \\ V'_r &= V \times \{0, 1, \dots, \Delta\} \\ A'_r &= \{((i, l), (j, p)) \mid (i, j) \in A, \exists k \in \{0, 1, \dots, S_{ij} - 1\} : p = l + l_{ij}^k\} \\ c_r(il, jp) &= c_{ij}^k, k : p = l + l_{ij}^k \\ \Delta &\in N \cup \{0\} \end{aligned}$$

La rete di ricerca G'_r è formata da coppie nodo-lunghezza: essere in una di queste coppie (i, l) significa, con riferimento alla rete originaria G , essere nel nodo i avendo percorso un cammino lungo l ; andare da (i, l) a (j, p) vuol dire altresì andare da i a j percorrendo l'arco (i, j) di lunghezza $(p - l)$, considerato quindi in uno stato di costo $c'_r(il, jp)$.

Si vede allora che un cammino minimo in G'_r da $(s, 0)$ a (t, l) corrisponde a un cammino in G da s a t che abbia esattamente lunghezza l e costo minimo: il valore più piccolo di l per il quale tale costo minimo non è maggiore del *budget* B corrisponde al valore ottimo $l^*(\Sigma^{OT})$.

Il parametro Δ deve essere scelto almeno pari a $l^*(\Sigma^{OT})$ e potrebbe perciò essere preso, per semplicità, pari a $l^*(\Sigma_0)$ (valore calcolabile in $O(\Pi(n, m, L))$) o, ancora più semplicemente, pari a $(n - 1)L$ (valore noto). È però chiaro che quanto più Δ è piccolo tanto più sono ridotte le dimensioni di G'_r e di conseguenza ci vuole meno tempo per esplorarla: per questo l'Algoritmo 3.2 adotta una strategia più sofisticata che riduce la sua complessità computazionale (Teorema 3.3).

Algoritmo 3.2 Diminuzione Discreta del Cammino Minimo II

1. Poni $\Delta := 0$.
2. Costruisci a partire da G la rete di ricerca G'_r .
3. Trova i cammini minimi in G'_r dal nodo $(s, 0)$ ai nodi (t, l) per $l \in \{0, 1, \dots, \Delta\}$.
4. Se tutti questi cammini minimi hanno costo maggiore di B poni $\Delta := \max\{1, 2\Delta\}$ e ritorna al punto 2.
5. Seleziona il cammino P di lunghezza minima tra quelli che hanno costo non maggiore di B .
6. Determina Σ^{OT} portando gli archi di P allo stato nel quale vengono attraversati e lasciando tutti gli altri allo stato zero.

Teorema 3.3 L'Algoritmo 3.2 fornisce una soluzione ottima per il Problema 3.3 in tempo $O(\Pi(n\delta, mS\delta, C))$ ($C = \max_{(i,j) \in A} c_{ij}^{S_{ij}}$; $\delta = l^*(\Sigma^{OT}) + 1$).

Dimostrazione.

L'ottimalità della soluzione fornita segue dalle considerazioni svolte in precedenza; il raddoppio di Δ al punto 4 garantisce che tale parametro sia, prima o poi, non minore di $l^*(\Sigma^{OT})$.

Per quanto riguarda invece la complessità computazionale si può osservare che, fissato il valore di Δ , la rete di ricerca G'_r consta di $n(\Delta + 1)$ nodi e al più $mS(\Delta + 1)$ archi, mentre le sue etichette assumono valori in $\{0, 1, \dots, C\}$; perciò i cammini minimi da $(s, 0)$ a tutti gli altri nodi di G'_r possono essere calcolati in $O(\Pi(n(\Delta + 1), mS(\Delta + 1), C))$ (cfr. sez. 3.1).

D'altra parte l'algoritmo genera una successione di valori $\{\Delta_i\}_{i=0, \dots, h}$ per ognuno dei quali costruisce G'_r ; il tempo totale che esso richiede è allora

$$O\left(\sum_{i=0}^h \Pi(n(\Delta_i + 1), mS(\Delta_i + 1), C)\right).$$

Poiché però per $\mu, \nu \in N$ si ha

$$\Pi(n\mu, mS\mu, C) + \Pi(n\nu, mS\nu, C) \leq \Pi(n(\mu + \nu), mS(\mu + \nu), C)$$

tale tempo totale è anche $O(\Pi(nH, mSH, C))$ con $H = \sum_{i=0}^h (\Delta_i + 1)$.

Inoltre qualunque sia h (cioè in qualsiasi momento l'algoritmo termini) si ha $H < 3(\Delta_h + 1)$ e, per il meccanismo di raddoppio, $\Delta_h + 1 < 2\delta$; la dimostrazione si completa osservando che per $\mu \in N$ si ha

$$\Pi(n\mu, mS\mu, C) \in O(\Pi(n, mS, C)).$$

◇

Si osservi come da un punto di vista pratico non è necessario generare in anticipo la rete di ricerca (quale che sia) ma se ne possono introdurre i nodi via via che l'algoritmo per i cammini minimi li richiede, continuando a utilizzare le liste di adiacenza di G ; in questo modo si evita anche di considerare nodi e archi irraggiungibili da $(s, 0)$.

3.2.3 Risoluzione Approssimata

Se nella risoluzione del Problema 3.3 l'efficienza assume un'importanza primaria e si possono avere valori comunque elevati di B e δ i due algoritmi appena proposti risultano, in quanto *pseudo*-polinomiali, troppo lenti. D'altra parte si è già detto che non è ragionevole sperare in un algoritmo polinomiale a meno che, aggiungiamo ora, non si sia disposti ad accettare una certa "degradazione" della soluzione.

Innanzitutto è possibile convertire l'Algoritmo 3.1 in uno schema di approssimazione pienamente polinomiale (FPTAS) del tipo ad ammissibilità garantita (cfr. sez. 2.2). A tal fine si consideri un *budget* $\hat{B}(t) = \lfloor B/2^t \rfloor$ e una *rete scalata* $G_c(t)$ con la stessa topologia di G e costi $\hat{c}_{ij}^k(t) = \lfloor c_{ij}^k/2^t \rfloor$: il problema che si ottiene può essere risolto efficientemente e la soluzione trovata può essere accettata anche per il problema originario. L'Algoritmo 3.3 concretizza questa intuizione come risulta dal successivo Teorema 3.4.

Algoritmo 3.3 FPTAS-I per la Diminuzione Discreta del Cammino Minimo

1. *Poni* $t := \max\{0, \lfloor \log(\epsilon B/n) \rfloor\}$.
2. *Considera il Problema 3.3 (Diminuzione Discreta del Cammino Minimo) sulla rete $G_c(t)$ con budget $\hat{B}(t)$.*
3. *Trova una soluzione Σ^A al problema del punto 2 mediante l'Algoritmo 3.1.*
4. *Fornisci come output Σ^A .*

Teorema 3.4 *L'Algoritmo 3.3 trova $\forall \epsilon \in (0, 1)$ una soluzione Σ^A ottimale ϵ -ammissibile ($l^*(\Sigma^A) \leq l^*(\Sigma^{OT})$ & $c(\Sigma^A) \leq B/(1 - \epsilon)$) per il Problema 3.3 in tempo $O(\Pi(n^2/\epsilon, mnS/\epsilon, L))$.*

Dimostrazione.

Se $t = 0$ si ha di fatto l'Algoritmo 3.1 e quindi la soluzione Σ^A è addirittura ottima mentre la complessità computazionale è $O(\Pi(n^2/\epsilon, mnS/\epsilon, L))$ perché $\epsilon B/n < 2$; possiamo perciò assumere $t = \lfloor \log(\epsilon B/n) \rfloor$.

Se Σ è uno stato ammissibile per il problema originario esso è uno stato ammissibile anche per il problema scalato ($\hat{c}(\Sigma) \leq \lfloor c(\Sigma)/2^t \rfloor$) e di conseguenza $l^*(\Sigma^A) \leq l^*(\Sigma^{OT})$. D'altra parte Σ^A è uno stato ammissibile per il problema scalato e quindi

$$c(\Sigma^A) \leq \hat{c}(\Sigma^A) \cdot 2^t + n \cdot 2^t \leq B + \epsilon B \leq \frac{B}{1 - \epsilon}$$

poiché l'Algoritmo 3.1 spende al più su $n - 1$ archi.

La limitazione per la complessità discende dall'essere $2^{t+1} > \epsilon B/n$ e quindi $\hat{B}(t) + 1 < 3n/\epsilon$ ($B > 0$ ma $\hat{B}(t) \geq 0$); per $\Pi(n, m, L) = \Pi(n, m)$ (cfr. sez. 3.1) si ottiene una limitazione *fortemente* polinomiale. \diamond

In alternativa a quanto visto sopra è possibile convertire l'Algoritmo 3.2 in uno schema di approssimazione pienamente polinomiale (FPTAS) del tipo a prestazione garantita (cfr. sez. 1.1). Per fare ciò si considera la *rete scalata* $G_t(t)$ avente topologia identica a G e lunghezze $\hat{l}_{ij}^k(t) = \lfloor l_{ij}^k/2^t \rfloor$. Poiché però δ non è noto a priori (a differenza di B) l'Algoritmo 3.4 deve cercare per tentativi il valore "giusto" di t : il successo della strategia adottata è testimoniato dal successivo Teorema 3.5.

Algoritmo 3.4 FPTAS-II per la Diminuzione Discreta del Cammino Minimo

1. *Poni $t := \lceil \log(L + 1) \rceil$.*
2. *Costruisci a partire da G la rete scalata $G_t(t)$.*
3. *Calcola mediante l'Algoritmo 3.2 una soluzione $\Sigma(t)$ ottima per il Problema 3.3 definito su $G_t(t)$.*
4. *Se $t > 0$ & $\hat{l}^*(\Sigma(t)) < n/\epsilon$ poni $t := t - 1$ e torna al punto 2.*
5. *Fornisci come Σ^A l'ultima soluzione $\Sigma(t)$ trovata al punto 3.*

Teorema 3.5 *L'Algoritmo 3.4 trova $\forall \epsilon \in (0, 1)$ una soluzione Σ^A ammissibile ϵ -approssimata ($c(\Sigma^A) \leq B$ & $l^*(\Sigma^A) \leq l^*(\Sigma^{OT})/(1 - \epsilon)$) per il Problema 3.3 in tempo $O(\Pi(n(\log L + n/\epsilon), mS(\log L + n/\epsilon), C))$.*

Dimostrazione.

L'algoritmo genera per il parametro t (che corrisponde al numero di *bit* meno significativi trascurati) una successione $\{t_i\}_{i=0}^h$ di valori in corrispondenza dei quali costruisce la rete scalata $G_l(t)$.

Se $t_h = 0$ la risoluzione è esatta, altrimenti

$$\frac{l^*(\Sigma^{OT})}{2^{t_h}} \geq \hat{l}^*(\Sigma(t_h)) \geq \frac{n}{\epsilon}$$

poiché Σ^{OT} è ammissibile nella rete scalata; parimenti (i costi non dipendono dal parametro t) lo stato $\Sigma(t_h)$ è ammissibile nella rete originaria ed essendo un cammino formato da al più $n - 1$ archi

$$l^*(\Sigma(t_h)) \leq 2^{t_h} \hat{l}^*(\Sigma(t_h)) + (n - 1)(2^{t_h} - 1)$$

per cui

$$l^*(\Sigma(t_h)) \leq 2^{t_h} \hat{l}^*(\Sigma^{OT}) + n \cdot 2^{t_h} \leq l^*(\Sigma^{OT}) + n \cdot 2^{t_h}$$

onde si conclude che

$$l^*(\Sigma(t_h)) \leq l^*(\Sigma^{OT}) (1 + \epsilon) \leq \frac{l^*(\Sigma^{OT})}{1 - \epsilon}$$

ovvero in ogni caso l'approssimazione ottenuta è quella desiderata.

Per quanto riguarda il tempo richiesto si ha sempre $t_0 \geq 1$ e $\hat{l}^*(\Sigma(t_0)) = 0$ per cui esso è $O(\sum_{i=0}^h \Pi(n\hat{\delta}_i, mS\hat{\delta}_i, C))$ con $h > 0$ e anche (cfr. teo. 3.3) $O(\Pi(nH, mSH, C))$ con $H = \sum_{i=0}^h \hat{\delta}_i$.

Se t_j è l'ultimo valore di t per il quale $\hat{l}^*(\Sigma(t)) = 0$ allora chiaramente $\sum_{i=0}^j \hat{\delta}_i \in O(\log L)$. Si può poi osservare che

$$2 \cdot \hat{l}^*(\Sigma(t_i)) \leq \hat{l}^*(\Sigma(t_{i+1})) \leq 2 \cdot \hat{l}^*(\Sigma(t_i)) + n - 1$$

come conseguenza dell'aggiunta di un *bit* "in coda" a quelli già considerati. Perciò da una parte $\hat{\delta}_{i+1} \geq 2 \cdot \hat{\delta}_i - 1$ e quindi $\sum_{i=j+1}^h \hat{\delta}_i \leq 3 \cdot \hat{\delta}_h$ e dall'altra

$$\hat{\delta}_h \leq 2 \cdot \hat{\delta}_{h-1} + n \leq n \left(1 + \frac{2}{\epsilon}\right) \leq 3 \cdot \frac{n}{\epsilon}$$

perché all'iterazione precedente l'algoritmo non si è fermato. Si ottiene così la limitazione di complessità polinomiale nella dimensione dell'input e nell'inverso di ϵ che costituisce la tesi. \diamond

3.3 Diminuzione Continua del Cammino Minimo

Sia data la rete orientata $G = (V, A; \Lambda, \lambda; \gamma)$ nella quale:

- $\Lambda_{ij} \in N \cup \{0\}$ è la lunghezza originaria dell'arco (i, j) ;
- $\lambda_{ij} \in N \cup \{0\}$ è il valore minimo al quale si può ridurre la lunghezza dell'arco (i, j) ($\lambda_{ij} \leq \Lambda_{ij}$);
- $\gamma_{ij} \in N$ è il costo da sostenere per ridurre di un'unità la lunghezza dell'arco (i, j) .

Per questa rete definiamo uno spazio degli stati continuo tramite le variabili di decisione $y_{ij} \in [0, \alpha_{ij}]$ (con $\alpha_{ij} = \Lambda_{ij} - \lambda_{ij}$ massima riduzione possibile per l'arco (i, j)); se in un dato stato di G è $y_{ij} = a$ vuol dire che $l_{ij} = \Lambda_{ij} - a$ in tale stato. Si continua a richiedere che per reti non orientate sia $y_{ij} = 0$ o $y_{ji} = 0$ intendendo in realtà lo stato di ij individuato da $\max\{y_{ij}, y_{ji}\}$.

Il vincolo di bilancio per gli stati ammissibili si scrive semplicemente $\sum_{(i,j) \in A} \gamma_{ij} y_{ij} \leq B$ e l'obiettivo è, similmente al caso discreto, trovare uno stato Σ^{OT} tale che sia $l^*(\Sigma^{OT}) \leq l^*(\Sigma)$ per ogni altro stato Σ ammissibile. Anche questo caso continuo viene formulato prendendo a modello il Problema 3.1 (Flusso Unitario a Costo Minimo).

Problema 3.5 *Diminuzione Continua del Cammino Minimo*

$$\begin{aligned}
 & \text{minimize} && \sum_{(i,j) \in A} (\Lambda_{ij} - y_{ij}) x_{ij} \\
 & \sum_{j:(v,j) \in A} x_{vj} - \sum_{i:(i,v) \in A} x_{iv} = && \begin{cases} +1 & v = s \\ 0 & v \in V - \{s, t\} \\ -1 & v = t \end{cases} \\
 & \sum_{(i,j) \in A} \gamma_{ij} y_{ij} \leq && B \\
 & 0 \leq x_{ij} \leq && 1 \\
 & 0 \leq y_{ij} \leq && \alpha_{ij}
 \end{aligned}$$

Se definiamo $c_{ij} = \gamma_{ij} \alpha_{ij}$ costo di completamento dell'arco (i, j) possiamo introdurre delle variabili di decisione alternative $z_{ij} = \gamma_{ij} y_{ij}$ che assumono valori in $[0, c_{ij}]$ e risultano utili in quanto, come vedremo, possono essere considerate intere.

Per quanto riguarda la complessità computazionale il Problema 3.5 non è più facile della sua versione discreta (Problema 3.3).

Teorema 3.6 *Il Problema 3.5 (Diminuzione Continua del Cammino Minimo) è debolmente NP-difficile.*

Dimostrazione.

Questo risultato è una diretta conseguenza del Teorema 4.4 (equivalenza tra problemi di flusso massimo e problemi di cammino minimo su reti planari) e del Teorema 4.3 (debole NP-difficoltà per l'interdizione del flusso su reti planari). \diamond

3.3.1 Risoluzione Esatta

Per risolvere esattamente il Problema 3.5 (Diminuzione Continua del Cammino Minimo) cominciamo prendendo spunto da un'idea di Phillips [46]: se conoscessimo *a priori* un cammino P minimo per G in Σ^{OT} potremmo senz'altro ricavare Σ^{OT} . Sarebbe infatti sufficiente limitarsi a spendere il *budget* B sugli archi di P , come di fatto avveniva anche nel caso discreto, e questo ridurrebbe il Problema 3.5 a un semplice problema di *Bisaccia Continua* (Problema 2.3) con costi unitari γ_{ij} per $(i, j) \in P$. Potremmo allora adottare una strategia di tipo *greedy* intervenendo sugli archi di P in ordine crescente di costo unitario (fino all'esaurimento del *budget* o alla massima riduzione possibile per P).

Questo modo di determinare Σ^{OT} non è utilizzabile in pratica, perché P non è noto a priori, ma evidenzia come basti considerare due soli casi:

1. Tutti gli archi di P sono o lasciati inalterati ($y_{ij} = z_{ij} = 0$) oppure *completati* ($y_{ij} = \alpha_{ij}$, $z_{ij} = c_{ij}$); chiameremo questo tipo di soluzione *tutto/niente*.
2. Su un solo arco (q, r) di P l'intervento viene *parzializzato*, quello durante il miglioramento del quale, nel corso della strategia *greedy*, si esaurirebbe il *budget* B .

In ogni caso è verificata l'interezza delle variabili z_{ij} poiché intero è il *budget*, intera la spesa per completare un arco e, per differenza, intera la spesa sull'arco qr (la riduzione y_{qr} della sua lunghezza è invece in generale razionale).

La migliore soluzione di tipo *tutto/niente* per il Problema 3.5 può essere individuata risolvendo il Problema 3.4 con $l_{ij}^0 = \Lambda_{ij}$, $l_{ij}^1 = \lambda_{ij}$ e $c_{ij}^1 = c_{ij}$. Esso è una versione semplificata del Problema 3.3 (Diminuzione Discreta del Cammino Minimo) e pertanto si possono utilizzare gli algoritmi della sezione 3.2 con il vantaggio di avere $S = 2$.

Per quanto riguarda invece le soluzioni di cui al punto 2 le si possono costruire a partire dal loro arco parzializzato: se infatti si ipotizza che sia (q, r) tale arco allora P andrà di necessità completato con un cammino *tutto/niente* da s a i e con un'altro da j a t . Conviene non distinguere fra “prima di i ” e “dopo di j ” e considerare un unico cammino *tutto/niente* da j a i vincolato a passare per un arco “fittizio” (t, s) di lunghezza (gratuita) nulla. Costruiamo allora la seguente *rete ausiliaria*:

$$\begin{aligned}
G_\rho &= (V_\rho, A_\rho; l^0, l^1; c^1) \\
V_\rho &= V \times \{0, 1\} \\
A_\rho &= \{((i, p), (j, p)) \mid (i, j) \in A, p \in \{0, 1\}\} \cup \{((t, 0), (s, 1))\} \\
l^0(ip, jp) &= \Lambda_{ij} \\
l^0(t0, s1) &= 0 \quad (S_{t0s1} = 1) \\
l^1(ip, jp) &= \lambda_{ij} \\
c^1(ip, jp) &= c_{ij}
\end{aligned}$$

Tale rete ausiliaria può essere pensata a due piani sovrapposti, ciascuno avente la topologia di G , con la possibilità di passare dal primo al secondo piano solamente “ritornando” da t a s ; un cammino da $(j, 0)$ a $(i, 1)$ può quindi essere visto come un cammino da j a i con l'obbligo di “cambiare piano” e quindi la necessità di passare per l'arco fittizio (t, s) .

Supponiamo che sia $z_{qr} = b \in \{1, \dots, K\}$ ($K = \min\{c_{qr} - 1, B\}$) la spesa sull'arco parzializzato (q, r) : il cammino ottimo P può allora completarsi con un cammino minimo da $(r, 0)$ a $(q, 1)$ di costo al più $B - b$. L'Algoritmo 3.5 sfrutta questa osservazione per risolvere il Problema 3.5, come affermato dal successivo Teorema 3.7.

Algoritmo 3.5 Diminuzione Continua del Cammino Minimo I

1. *Calcola mediante l'Algoritmo 3.1 la migliore soluzione P' del tipo tutto/niente.*
2. *Calcola per ogni arco $(i, j) \in A$ (con i seguenti passi) la migliore soluzione $P(ij)$ nell'ipotesi che sia parzializzato tale arco (i, j) :*
 - (a) *Calcola in G_ρ mediante l'Algoritmo 3.1 i cammini minimi tutto/niente $P[ij, c]$ da $(j, 0)$ a $(i, 1)$ che abbiano al più costo $c \in \{0, \dots, B - 1\}$.*
 - (b) *Per ogni $b \in \{1, \dots, K\}$ ($K = \min\{c_{qr} - 1, B\}$) poni $P_{ij}(b) := P[ij, B - b] \cup \{(i, j)\} - \{(t, s)\}$ ($z_{ij} := b$).*

- (c) Prendi $P(ij)$ tale che sia $l(P_{ij}(b))$ minima.
3. Seleziona P'' di lunghezza minima tra i $P(ij)$.
 4. Prendi P tra P' e P'' di modo che abbia lunghezza minima.
 5. Determina Σ^{OT} a partire da P .

Teorema 3.7 *L'Algoritmo 3.5 fornisce una soluzione ottima per il Problema 3.5 in tempo $O(m\Pi(nB, mB, L))$.*

Dimostrazione.

Sulla base della discussione svolta esiste una soluzione ottima *tutto/niente* oppure con al più un arco parzializzato: l'algoritmo trova necessariamente una tale soluzione al punto 1 o al punto 2.

A proposito del punto 2 si osservi che $P_{ij}(b)$ può in generale contenere un *ciclo* ma, in tal caso, esso passerà per (i, j) e avrà quindi lunghezza e costo positivi ((i, j) è parzializzato); perciò a $P_{ij}(b)$ sarà preferito il cammino *tutto/niente* che da esso si ottiene eliminando il ciclo. Ne segue che P è senz'altro un *cammino* (cioè passa per ogni arco di G al più una volta) e quindi l'algoritmo è valido anche su grafi non orientati—per i quali si richiede $z_{ij} = 0$ o $z_{ji} = 0$.

Il tempo richiesto è asintoticamente determinato dal punto 2 ed è il tempo preso da m esecuzioni dell'Algoritmo 3.1 su G_ρ (con $S = 2$). Si sfrutta il fatto che tale algoritmo può fornire, senza aumentare in complessità, i cammini minimi per tutti i valori del *budget* non maggiori di B . \diamond

Per la ricerca della migliore soluzione con un arco parzializzato è anche possibile un'altro approccio: se (q, r) è tale arco e il cammino da r a q tramite (t, s) è lungo esattamente l allora tale cammino può essere scelto, senza perdita di generalità, di costo minimo c_{min} ; ne risulta $z_{qr} = B - c_{min}$ poiché in una soluzione parzializzata il *budget* viene speso per intero.

Se si considerano i valori di l in $\{0, 1, \dots, \Delta\}$ si ottiene per ogni arco (i, j) una lista di al massimo $\Delta + 1$ soluzioni $P_{ij}(l)$ di lunghezza $\lambda = l + \Lambda_{ij} - (z_{ij}/\gamma_{ij})$ (se $z_{ij} \leq 0$ allora l non è una lunghezza possibile per il cammino da j a i tramite (t, s) mentre se $z_{ij} \geq c_{ij}$ c'è una soluzione *tutto/niente* migliore).

La lista sopra descritta contiene tutte le soluzioni potenzialmente ottime di lunghezza non superiore a Δ e anche, in generale, qualche soluzione di lunghezza maggiore di Δ . Perciò se in tale lista si trova un cammino di lunghezza minima e non maggiore di Δ esso è certamente ottimo (tra quelli con l'arco (i, j) parzializzato). Diamo allora l'Algoritmo 3.6 e il Teorema 3.8 relativo alle sue prestazioni.

Algoritmo 3.6 Diminuzione Continua del Cammino Minimo II

1. Poni $\Delta := 0$.
2. Poni $LISTA := \emptyset$.
3. Calcola in G mediante l'Algoritmo 3.2 i cammini di tipo tutto/niente meno costosi da s a t che abbiano esattamente lunghezza $l \in \{0, 1, \dots, \Delta\}$: se ce ne sono alcuni di costo non maggiore di B metti in $LISTA$ il più breve di questi.
4. Per ogni arco $(i, j) \in A$ (mediante i seguenti passi) aggiungi a $LISTA$ le soluzioni potenzialmente ottime per le quali il cammino da j a i tramite (t, s) è lungo $l \in \{0, 1, \dots, \Delta\}$:
 - (a) Calcola mediante l'Algoritmo 3.2 i cammini tutto/niente meno costosi da $(j, 0)$ a $(i, 1)$ in G_ρ che abbiano esattamente lunghezza $l \in \{0, 1, \dots, \Delta\}$.
 - (b) Per ognuno di questi cammini $P[ij, l]$, avente costo $c_{min}[ij, l]$, poni $z_{ij} := B - c_{min}[ij, l]$.
 - (c) Se $z_{ij} \notin \{1, \dots, c_{ij} - 1\}$ allora scarta $P[ij, l]$ altrimenti metti in $LISTA$ il cammino $P_{ij}(l) = P[ij, l] \cup \{(i, j)\} - \{(t, s)\}$ di lunghezza $\lambda = l + \Lambda_{ij} - (z_{ij}/\gamma_{ij})$.
5. Elimina da $LISTA$ le soluzioni più lunghe di Δ e se poi $LISTA = \emptyset$ allora poni $\Delta := \max\{1, 2\Delta\}$ e torna al punto 2.
6. Prendi P di lunghezza minima in $LISTA$ e determina Σ^{OT} .

Teorema 3.8 L'Algoritmo 3.6 fornisce una soluzione ottima per il Problema 3.5 in tempo $O(m\Pi(n\delta, m\delta, C))$ ($C = \max_{(i,j) \in A} c_{ij}$; $\delta = l^*(\Sigma^{OT}) + 1$).

Dimostrazione.

Per quanto riguarda la correttezza è chiaro, sulla base della discussione svolta, che l'algoritmo individua Σ^{OT} non appena risulti essere $\Delta \geq l^*(\Sigma^{OT})$.

La complessità della singola iterazione è invece determinata dalle $m + 1$ ripetizioni dell'Algoritmo 3.2 ed è quindi $O(m\Pi(n(\Delta + 1), m(\Delta + 1), C))$; in particolare la scansione di $LISTA$ è $O(m(\Delta + 1))$ e quindi è trascurabile.

L'algoritmo genera per Δ una successione di valori $\{\Delta_i\}_{i=0}^h$ e dunque (cfr. teo. 3.3) il tempo richiesto complessivamente è $O(m\Pi(nH, mH, C))$ con $H = \sum_{i=0}^h \Delta_i$. Ma $H < 3(\Delta_h + 1)$ e inoltre, per il meccanismo di raddoppio, $\Delta_h + 1 < 2\delta$: ne segue la tesi (in questo caso δ è in generale razionale). \diamond

3.3.2 Risoluzione Approssimata

Anche nel caso continuo che stiamo ora esaminando è possibile convertire gli algoritmi pseudo-polinomiali descritti in schemi di approssimazione pienamente polinomiali (FPTAS).

L'Algoritmo 3.7 fa assumere alle variabili z_{ij} degli archi parzializzati solo valori multipli di 2^t e completa le soluzioni corrispondenti cercando cammini minimi *tutto/niente* in una rete ausiliaria scalata $G_\rho(t)$. Tale rete $G_\rho(t)$ viene costruita con le lunghezze originarie λ_{ij} e Λ_{ij} e con i costi di completamento scalati $\hat{c}_{ij}(t) = \lfloor c_{ij}/2^t \rfloor$. Considerando un *budget* $\hat{B}(t) = \lfloor B/2^t \rfloor$ si riesce allora a realizzare (Teorema 3.9) uno schema di approssimazione ad ammissibilità garantita (Algoritmo 3.7).

Algoritmo 3.7 FPTAS-I per la Diminuzione Continua del Cammino Minimo

1. *Calcola in G mediante l'Algoritmo 3.3 (FPTAS-I per la Diminuzione Discreta del Cammino Minimo) una soluzione P' ottimale ϵ -ammissibile tra quelle del tipo tutto/niente.*
2. *Poni $t := \max\{0, \lfloor \log(\epsilon B/n) \rfloor\}$.*
3. *Calcola per ogni arco $(i, j) \in A$ (con i seguenti passi) una soluzione $P(ij)$ ottimale ϵ -ammissibile nell'ipotesi che sia parzializzato tale arco (i, j) :*
 - (a) *Calcola in $G_\rho(t)$ mediante l'Algoritmo 3.1 i cammini ottimi tutto/niente $P[ij, \hat{c}]$ per andare da $(j, 0)$ a $(i, 1)$ con un budget $\hat{c} \in \{0, \dots, \hat{B}(t)\}$.*
 - (b) *Per ogni $b \in \{1, \dots, K\}$ ($K = \min\{c_{ij} - 1, B\}$) che sia multiplo di 2^t (e in ogni caso per $b = K$) poni $P_{ij}(b) := P[ij, \hat{c}(b)] \cup \{(i, j)\} - \{(t, s)\}$ ($z_{ij} := b$) con $\hat{c}(b) = \lfloor (B - b + 2^t - 1)/2^t \rfloor$.*
 - (c) *Prendi $P(ij)$ tale che sia $l(P_{ij}(b))$ minima.*

4. Seleziona P'' di lunghezza minima tra i $P(ij)$.
5. Prendi P tra P' e P'' di modo che abbia lunghezza minima.
6. Determina Σ^A a partire da P .

Teorema 3.9 *L'Algoritmo 3.7 fornisce $\forall \epsilon \in (0, 1)$ una soluzione Σ^A ottimale ϵ -ammissibile ($l^*(\Sigma^A) \leq l^*(\Sigma^{OT})$ & $c(\Sigma^A) \leq B/(1 - \epsilon)$) per il Problema 3.5 in tempo $O(m\Pi(n^2/\epsilon, mn/\epsilon, L))$.*

Dimostrazione.

Per quanto riguarda la correttezza è sufficiente dimostrare che tutte le soluzioni generate sono ϵ -ammissibili e che almeno una di esse è ottimale.

La soluzione P' del punto 1 è chiaramente ϵ -ammissibile; se $t = 0$ le soluzioni $P_{ij}(b)$ del punto 3(b) sono addirittura ammissibili. D'altra parte se $t = \lfloor \log(\epsilon B/n) \rfloor$ allora (cfr. teo. 3.4) si ha

$$c(P_{ij}(b)) = b + c(P[ij, \hat{c}(b)]) \leq b + (B - b + 2^t - 1) + (n - 2) \cdot 2^t \leq \frac{B}{1 - \epsilon}$$

e quindi le soluzioni del punto 3(b) sono ϵ -ammissibili.

Se Σ^{OT} è del tipo tutto niente allora P' è ottimale; altrimenti se qr è il suo arco parzializzato $P_{qr}(b)$ risulta ottimale per $b = \min\{\lceil z_{qr}^{OT}/2^t \rceil \cdot 2^t, K\}$. Infatti da una parte $z_{qr} = b \geq z_{qr}^{OT}$ e dall'altra $l(P[qr, \hat{c}(b)]) \leq l(P^{OT} - \{qr\})$ perché $\hat{c}(P^{OT} - \{qr\}) \leq \hat{c}(b)$ (cfr. teo. 3.4).

Per quanto riguarda invece la complessità computazionale essa è asintoticamente determinata dal punto 3, ossia da m ripetizioni dell'Algoritmo 3.1: $S = 2$ e t viene scelto in modo tale che $\hat{B}(t) + 1 < 3n/\epsilon$. Si ottiene un risultato di *forte* polinomialità per $\Pi(n, m, L) = \Pi(n, m)$ (cfr. sez. 3.1). \diamond

Al fine di realizzare un FPTAS a prestazione garantita per il Problema 3.5 conviene considerare una rete scalata $G(t)$ avente la stessa topologia di G , lunghezze originarie $\hat{\Lambda}_{ij} = \lfloor \Lambda_{ij}/2^t \rfloor$, lunghezze minime $\hat{\lambda}_{ij} = \lceil \lambda_{ij}/2^t \rceil$ e costi unitari $\hat{\gamma}_{ij} = 2^t \cdot \gamma_{ij}$. I costi di completamento $\hat{c}_{ij} = (\hat{\Lambda}_{ij} - \hat{\lambda}_{ij}) \hat{\gamma}_{ij}$ per gli archi di $G(t)$ sono tali che $c_{ij} - 2^{t+1} \cdot \gamma_{ij} \leq \hat{c}_{ij} \leq c_{ij}$ e quindi uno stato $\hat{\Sigma} = \{\hat{z}_{ij}\}$ ammissibile per $G(t)$ risulta ammissibile anche per G mentre uno stato $\Sigma = \{z_{ij}\}$ ammissibile per G può essere trasformato in uno stato $\hat{\Sigma} = \{\hat{z}_{ij}\}$ ammissibile per $G(t)$ ponendo $\hat{z}_{ij} := \min\{z_{ij}, \hat{c}_{ij}\}$. Si ricorda che la variabile di stato z_{ij} determina la lunghezza dell'arco relativo $l_{ij} = \Lambda_{ij} - z_{ij}/\gamma_{ij}$ e si ricorda inoltre che, come nel caso discreto, il valore ottimo $l^*(\Sigma^{OT})$ non è in generale noto a priori. L'Algoritmo 3.8, le cui prestazioni sono descritte dal successivo Teorema 3.10, deve pertanto realizzare quel compromesso tra precisione e tempo di esecuzione che costituisce l'essenza di ogni (F)PTAS procedendo per tentativi sul valore del parametro t .

Algoritmo 3.8 FPTAS-II per la Diminuzione Continua del Cammino Minimo

1. Poni $t := \lceil \log(L + 1) \rceil$.
2. Costruisci la rete scalata $G(t)$.
3. Trova mediante l'Algoritmo 3.6 una soluzione $\Sigma(t)$ ottima per il Problema 3.5 definito su $G(t)$.
4. Se $t > 0$ & $\hat{l}^*(\Sigma(t)) < n(2 + 3/\epsilon)$ poni $t := t - 1$ e torna al punto 2.
5. Fornisci come output Σ^A l'ultima soluzione $\Sigma(t)$ trovata al punto 3.

Teorema 3.10 L'Algoritmo 3.8 trova $\forall \epsilon \in (0, 1)$ una soluzione Σ^A ammissibile ϵ -approssimata ($c(\Sigma^A) \leq B$ & $l^*(\Sigma^A) \leq l^*(\Sigma^{OT})/(1 - \epsilon)$) per il Problema 3.5 in tempo $O(m \Pi(n^2 \log L/\epsilon, nm \log L/\epsilon, C))$.

Dimostrazione.

Si consideri la successione $\{t_i\}_{i=0}^h$ dei valori di t in corrispondenza dei quali viene costruita la rete scalata $G(t)$. Se $t_h = 0$ la risoluzione è esatta, altrimenti

$$\frac{l^*(\Sigma^{OT})}{2^{t_h}} \geq \hat{l}^*(\Sigma(t_h)) - 2n \geq 3n/\epsilon$$

perché esiste $\hat{\Sigma}^{OT}$ ammissibile per $G(t)$; d'altra parte $\Sigma(t_h)$ è uno stato ammissibile per la rete G , oltre che uno stato ottimo per la rete $G(t)$, quindi

$$l^*(\Sigma(t_h)) \leq 2^{t_h} \hat{l}^*(\Sigma(t_h)) + n \cdot 2^{t_h} \leq 2^{t_h} \hat{l}^*(\hat{\Sigma}^{OT}) + n \cdot 2^{t_h}$$

e di conseguenza

$$l^*(\Sigma(t_h)) \leq l^*(\Sigma^{OT}) + 3n \cdot 2^{t_h} \leq \frac{l^*(\Sigma^{OT})}{1 - \epsilon}$$

ossia in ogni caso l'approssimazione ottenuta è quella desiderata.

Il tempo impiegato è chiaramente $O(\sum_{i=0}^h \Pi(n\hat{\delta}_i, mS\hat{\delta}_i, C))$ e quindi anche (cfr. teo. 3.3) $O(\Pi(nH, mSH, C))$ con $H = \sum_{i=0}^h \hat{\delta}_i$. Inoltre $h \in O(\log L)$ e $\hat{\delta}_i \in O(n/\epsilon) \forall i \in \{0, \dots, h\}$ per cui $H \in O(n \log L/\epsilon)$. \diamond

3.4 Accrescimento del Cammino Minimo

Sia data la rete orientata $G = (V, A; \lambda, \Lambda; \gamma)$ nella quale:

- $\lambda_{ij} \in N \cup \{0\}$ è la lunghezza originaria dell'arco (i, j) ;
- $\Lambda_{ij} \in N \cup \{0\}$ è il valore massimo che può raggiungere la lunghezza dell'arco (i, j) ($\Lambda_{ij} \geq \lambda_{ij}$);
- $\gamma_{ij} \in N$ è il costo da sostenere per aumentare di un'unità la lunghezza dell'arco (i, j) .

In tale rete le variabili di decisione $y_{ij} \in [0, \beta_{ij}]$ ($\beta_{ij} = \Lambda_{ij} - \lambda_{ij}$ massimo accrescimento possibile per l'arco $(i, j) \in A$) definiscono uno spazio degli stati continuo molto simile a quello visto nella sezione 3.3 (Diminuzione Continua del Cammino Minimo). Il vincolo di bilancio per gli stati ammissibili ha sempre la forma $\sum_{(i,j) \in A} \gamma_{ij} y_{ij} \leq B$ mentre l'obiettivo diventa la ricerca di uno stato Σ^{OT} nel quale $l^*(\Sigma^{OT})$ sia massima rispetto a tutti gli stati ammissibili. Conviene formulare il problema in esame con riferimento al Problema 3.2 (Massima Tensione) e si ottiene così il Problema 3.6.

Problema 3.6 *Accrescimento Continuo del Cammino Minimo*

$$\begin{aligned}
 & \text{maximize} && d(t) - d(s) \\
 & d(j) - d(i) - y_{ij} &\leq & \lambda_{ij} \quad \forall (i, j) \in A \\
 & \sum_{(i,j) \in A} \gamma_{ij} y_{ij} &\leq & B \\
 & 0 &\leq & y_{ij} \leq \beta_{ij}
 \end{aligned}$$

Così formulato (cfr. Fulkerson e Harding [23]) il problema di accrescere il cammino minimo è un problema di programmazione lineare e perciò da una parte esiste l'algoritmo debolmente polinomiale dell'ellissoide (si veda per es. Grötschel et al. [30, cap. 3]) e dall'altra è disponibile il metodo del simpleso (si veda per es. Ahuja et al. [2, app. C]) il quale gode di un eccellente comportamento pratico. Fulkerson e Harding [23] propongono anche (tramite il duale del Problema 3.6) una formulazione come flusso a costo minimo parametrizzato del caso in cui la lunghezza di ogni arco possa essere aumentata senza alcuna limitazione superiore ($\Lambda_{ij} = \infty \forall (i, j) \in A$): essi risolvono il problema in questione in tempo pseudo-polinomiale mediante una variante dell'algoritmo Primal-Dual (presentato per es. da Ahuja et al. [2, sez. 9.8]).

Per quanto riguarda il caso di rete non orientata se in una soluzione ammissibile è $d(j) \leq d(i)$ allora si può senz'altro porre in essa $y_{ij} = 0$ ottenendo

così una soluzione ammissibile non peggiore: la peculiarità di questo caso è pertanto solo apparente.

Golden [29] ha studiato un problema strettamente collegato a quello di Fulkerson e Harding [23], ovvero al Problema 3.6 con $\Lambda_{ij} = \infty$: esso può formularsi come segue secondo il modello del Problema 3.2 (Massima Tensione).

Problema 3.7 *Accrescimento Vincolato di Costo Minimo*

$$\begin{aligned} & \text{minimize} && \sum_{(i,j) \in A} \gamma_{ij} y_{ij} \\ & d(t) - d(s) & \geq & \tau \\ & d(j) - d(i) - y_{ij} & \leq & \lambda_{ij} \quad \forall (i,j) \in A \\ & y_{ij} & \geq & 0 \end{aligned}$$

Nel Problema 3.7 lunghezze e costi si scambiano in qualche modo di ruolo, rispetto al Problema 3.6, poiché il vincolo di bilancio è sostituito da un vincolo sulla lunghezza del cammino minimo mentre il costo dell'accrescimento assume il ruolo di funzione obiettivo: si minimizza cioè il costo da sostenere affinché il cammino minimo della rete abbia lunghezza almeno τ .

Golden [29] mostra come il duale del Problema 3.7 sia un problema di flusso a costo minimo e di conseguenza come sia possibile una sua risoluzione efficiente. Ahuja et al. [2] introducono il problema del flusso a costo minimo (cap. 9), espongono algoritmi polinomiali per la sua risoluzione (cap. 10) e infine mostrano come le soluzioni ottime primali e duali possano essere considerate intere (cap. 11). Una interessante conseguenza di quest'ultimo fatto è che mediante una ricerca binaria sul valore τ del cammino minimo accresciuto è possibile risolvere efficientemente anche il seguente caso particolare di accrescimento discreto (cfr. sez. 4.3).

Problema 3.8 *Accrescimento Intero del Cammino Minimo*

$$\begin{aligned} & \text{maximize} && d(t) - d(s) \\ & d(j) - d(i) - y_{ij} & \leq & \lambda_{ij} \quad \forall (i,j) \in A \\ & \sum_{(i,j) \in A} \gamma_{ij} y_{ij} & \leq & B \\ & y_{ij} & \in & N \cup \{0\} \end{aligned}$$

Ricordiamo infine che Ball et al. [10] hanno studiato il problema degli archi più *essenziali* per il cammino minimo di una rete. Si tratta dell'individuazione di un insieme di archi la cui rimozione comporti, compatibilmente

| | Diminuzione | Accrescimento |
|-------------------|---|--|
| Modalità Discreta | deb. NP-difficile (teo. 3.1) FPTAS ϵ -ammis. (alg. 3.3) FPTAS ϵ -appros. (alg. 3.4) | deb. NP-difficile, ma in un caso particolare alg. pol. (prob. 3.8) |
| Modalità Continua | deb. NP-difficile (teo. 3.6) FPTAS ϵ -ammis. (alg. 3.7) FPTAS ϵ -appros. (alg. 3.8) | Prog. Lin. (prob. 3.6) |

Tabella 3.1: Quadro riassuntivo dei problemi di cammino minimo.

con un vincolo di bilancio, un aumento massimo per $mis(G) = l^*$. Il problema è NP-difficile e perciò Ball et al. [10] risolvono (in tempo polinomiale) una sua variante più semplice che fornisce qualche indicazione sulla soluzione ottima del problema originario.

3.5 Prospettive e Sviluppi

Nel corso del capitolo sono stati studiati alcuni problemi di ottimizzazione su grafi che sorgono quando si voglia intervenire sugli archi di una rete per modificare, sotto un vincolo di bilancio, la lunghezza del cammino minimo tra due suoi nodi. Tali problemi possono essere classificati in funzione del loro *obiettivo* (diminuzione o accrescimento) e in funzione della *modalità di intervento* che prevedono (discreta o continua). La Tabella 3.1 rispecchia questa classificazione e riassume i risultati ottenuti: il seguito della sezione è dedicato a una loro corretta collocazione nell'ambito della letteratura specializzata.

Il Problema 3.3 (Diminuzione Discreta del Cammino Minimo) non sembra essere stato studiato in precedenza, sebbene lo si possa vedere come un problema di Cammino Minimo Vincolato definito su un multigrafo (cfr. cap. 5). In un problema di Cammino Minimo Vincolato ogni arco è caratterizzato da una lunghezza l_{ij} e da un costo c_{ij} e l'obiettivo è trovare un cammino di lunghezza minima tra quelli aventi costo non maggiore di un *budget* B assegnato. Il problema del Cammino Minimo Vincolato è un problema ben noto in letteratura, come ben nota è la sua NP-difficoltà (per la quale si può vedere per es. Ahuja et al. [2, sez. B.4]); esso è stato studiato da Hassin [31] che ha proposto due FPTAS a prestazione garantita per il caso di rete priva di cicli orientati. Anche il Problema 3.5 (Diminuzione Continua del Cammino Minimo) non risulta studiato in precedenza, nemmeno in casi particolari; d'altra parte esiste un grosso debito nei confronti di Phillips [46] per le tecniche con le quali viene affrontata la diminuzione del cammino minimo sia nel

caso discreto che in quello continuo. Dal punto di vista della complessità computazionale l'approccio ad ammissibilità garantita (la cui applicazione a problemi di cammino minimo è originale) risulta più efficiente di quello a prestazione garantita per via del fatto che $l^*(\Sigma^{OT})$ (a differenza di B) non è noto a priori: se sia possibile ridurre o eliminare questo *gap* è senz'altro un possibile spunto per ulteriori ricerche.

Il Problema 3.6 (Accrescimento Continuo del Cammino Minimo) è un problema di fatto formulato e risolto nell'ambito della programmazione lineare da Fulkerson e Harding [23] e pertanto viene considerato principalmente per ragioni di completezza. Una forma generale di Accrescimento Discreto del Cammino Minimo (corrispondente a quella vista per la diminuzione) non è stata oggetto di studio neanche in questa sede: per essa vale certamente un risultato di debole NP-completezza, che si può dimostrare analogamente al Teorema 3.1, e pertanto un'eventuale ricerca andrà indirizzata verso schemi di approssimazione polinomiale (come del resto si è fatto per la diminuzione). Si è invece visto che nel caso particolare del Problema 3.8 (Accrescimento Intero del Cammino Minimo) è possibile una risoluzione in tempo polinomiale basata sul lavoro di Golden [29] e su tecniche di ricerca binaria; al contrario la Diminuzione Intera del Cammino Minimo viene risolta dagli algoritmi proposti per il Problema 3.3 (Diminuzione Discreta del Cammino Minimo) in tempo pseudo-polinomiale, perché in questo caso la dimensione dell'input è proporzionale a $\log S$ e non a S .

Per quanto riguarda infine alcune possibili varianti dei problemi di questo capitolo c'è per esempio la possibilità di intervenire sui nodi invece che sugli archi, similmente a quanto fatto da Krumke et al. [38] per la diminuzione del minimo albero ricoprente (cfr. sez. 5.3). Un'altra possibilità è considerare i cammini minimi da un nodo a tutti gli altri: Golden [29] ha cercato interventi di costo minimo che diano lunghezze non superiori a quelle assegnate mentre Berman [12] ha cercato di minimizzare sotto un vincolo di bilancio la somma di tutte le lunghezze. Può anche essere interessante considerare più vincoli di bilancio (su diverse risorse) ovvero riferirsi a un modello di *Bisaccia* più complicato.

Capitolo 4

Problemi di Flusso Massimo

Questo capitolo si occupa del caso in cui l'efficienza della rete sia misurata dal massimo flusso che è possibile inviare attraverso di essa da un nodo sorgente a un nodo destinazione.

Similmente a quanto visto per i problemi di cammino minimo la prima sezione introduce il problema classico di calcolare il flusso massimo tra due nodi di una rete mentre l'ultima riassume il capitolo con lo sguardo rivolto ai possibili sviluppi della ricerca.

La seconda sezione è invece dedicata alla diminuzione ottima del flusso massimo sotto un vincolo di bilancio; tale argomento si rivela adatto a introdurre l'equivalenza tra problemi di flusso massimo e di cammino minimo limitatamente a una certa classe di reti planari.

La terza sezione tratta infine il problema opposto di accrescere il flusso massimo sotto un vincolo di bilancio; in tale contesto risulta utile introdurre il problema del Flusso Intero a Costo Convesso Minimo (Problema 4.5).

4.1 Introduzione al Flusso Massimo

Data una rete orientata $G = (V, A; u)$, dove $u : A \rightarrow N \cup \{0\}$ è una funzione capacità per i suoi archi, si intenderà che l'arco (i, j) possa sostenere un flusso $x_{ij} \in \{0, 1, \dots, u_{ij}\}$. L'insieme dei flussi sostenuti dai vari archi di A individua in G un *preflusso* e questo definisce per ogni nodo $i \in V$ la quantità

$$e(i) = \sum_{j:(j,i) \in A} x_{ji} - \sum_{j:(i,j) \in A} x_{ij}$$

che chiameremo *eccesso* del nodo nel preflusso. Fissati poi un nodo sorgente s e un nodo destinazione t parleremo di *flusso* da s a t se valgono le condizioni

$$e(i) = 0 \quad i \in V - \{s, t\} \quad \& \quad e(t) = -e(s) = \nu \geq 0$$

dove ν viene detto *valore* del flusso.

Il problema del flusso massimo consiste nel cercare un flusso da s a t che abbia il valore più elevato possibile (se indichiamo con ν^* tale valore sarà $\text{mis}(G) = \nu^*$ in questo capitolo) e può essere formulato, similmente al problema del cammino minimo, nell'ambito della programmazione lineare: anche in questo caso è garantita l'esistenza di una soluzione ottima intera (si veda per es. Ahuja et al. [2, teo. 6.5]).

Problema 4.1 *Flusso Massimo*

$$\begin{aligned} & \text{maximize } \nu \\ & \sum_{j:(v,j) \in A} x_{vj} - \sum_{i:(i,v) \in A} x_{iv} = \begin{cases} +\nu & v = s \\ 0 & v \in V - \{s, t\} \\ -\nu & v = t \end{cases} \\ & 0 \leq x_{ij} \leq u_{ij} \end{aligned}$$

Il caso di rete non orientata, analogamente a quanto visto per il cammino minimo, non presenta sostanziale diversità perché sostituendo ogni arco ij di capacità u_{ij} con la coppia di archi orientati (i, j) e (j, i) entrambi di capacità u_{ij} si ottiene una rete orientata "equivalente" ($|x_{ij} - x_{ji}|$ sarà il flusso *netto* in uno dei due versi di ij).

Inoltre se G è un multigrafo semplicemente si complicano la notazione e la rappresentazione sull'elaboratore, potendosi peraltro ignorare i cappi e semplificare gli archi paralleli con un unico arco avente per capacità la somma delle capacità in parallelo.

Partizionando l'insieme V dei nodi di G in due insiemi S e $\bar{S} = V - S$ e considerando i soli archi (i, j) che vanno da S a \bar{S} si ottiene un *taglio* $T = \{(i, j) \in A \mid i \in S \ \& \ j \in \bar{S}\}$ di G che si dice *st-taglio* se $s \in S$ e $t \in \bar{S}$. La capacità di un taglio T si definisce come $u(T) = \sum_{(i,j) \in T} u_{ij}$ e il problema del Taglio Minimo consiste nella ricerca di un *st-taglio* avente capacità la più piccola possibile (u^*).

Il seguente importante e famoso teorema mostra il legame di tipo duale che esiste tra il problema del flusso massimo e quello del taglio minimo.

Teorema 4.1 *Il valore del massimo flusso che può essere inviato da s a t è pari alla minima capacità di un st -taglio ($\nu^* = u^*$).*

Dimostrazione.

Si veda per esempio Ahuja et al. [2, cap. 6]. ◇

Di conseguenza possiamo anche considerare $\text{mis}(G) = u^*$ in questo capitolo e senz'altro lo faremo quando si tratterà di diminuire $\text{mis}(G)$ (sez. 4.2).

Per quanto riguarda la risoluzione del Problema 4.1, cioè per quanto riguarda il calcolo del flusso massimo, sono presenti in letteratura diversi algoritmi dei quali Ahuja et al. [2, cap. 7] presentano una rassegna storica. Qui si dirà soltanto che mentre una prima generazione di algoritmi evolveva dal flusso nullo ($x_{ij} = 0 \forall (i, j) \in A$) al flusso massimo attraverso la ricerca di *cammini aumentanti* una seconda generazione, da considerarsi superiore sia teoricamente che praticamente, realizza tale evoluzione mediante una successione di *preflussi*. Alon [5] fornisce un algoritmo avente complessità $O(mn)$ per $m \in \Omega(n^{5/3} \log n)$ e $O(mn \log n)$ altrimenti: esso è il migliore che si conosca nell'ambito di quelli fortemente polinomiali tranne per il caso dei grafi *densi* ($m \in \Omega(n^2)$) nel quale esiste un algoritmo dovuto a Cheriyan, Hagerup e Mehlhorn [17] avente complessità $O(n^3 / \log n)$. Tra gli algoritmi debolmente polinomiali si segnala la complessità $O(mn \log(n\sqrt{\log U}/m + 2))$ ottenuta da Ahuja, Orlin e Tarjan [4] mentre è sempre possibile, trattandosi di programmazione lineare, avvalersi del buon comportamento empirico del metodo del simplesso (si veda a tale proposito Ahuja et al. [2, cap. 11]). Indicheremo nel seguito con $\Phi(n, m, U)$ la migliore limitazione conosciuta per il problema del flusso massimo.

Una semplice applicazione per il modello del flusso massimo è una rete di elaboratori connessi da canali aventi banda passante u_{ij} (Kb/s) destinata a trasmettere informazioni da s a t : in questo caso il flusso è un flusso informativo che s massimizzerà per comunicare con t in modo efficiente. Anche per il problema del flusso massimo (come per quello del cammino minimo) la letteratura propone una grande varietà di applicazioni, per una buona panoramica delle quali si può vedere per esempio Ahuja et al. [2].

Faremo l'ipotesi, secondo l'impostazione scelta, di avere a disposizione un *budget* B spendibile per diminuire (sez. 4.2) o aumentare (sez. 4.3) le capacità u_{ij} degli archi di G , essendo l'obiettivo determinato dal punto di vista del decisore (nell'esempio dal suo interesse a ostacolare piuttosto che a favorire il flusso di dati da s a t). Analogamente a quanto visto per i problemi di cammino minimo distingueremo una modalità di intervento *continua* e una modalità di intervento *discreta* (nell'ambito della quale individueremo una particolare modalità di intervento *intera*).

4.2 Diminuzione del Flusso Massimo

Si consideri la rete orientata $G = (V, A; \Upsilon, v; \gamma)$ nella quale:

- $\Upsilon_{ij} \in N \cup \{0\}$ è la capacità originaria dell'arco (i, j) ;

- $v_{ij} \in N \cup \{0\}$ è il valore al di sotto del quale non si può ridurre la capacità dell'arco (i, j) ($v_{ij} \leq \Upsilon_{ij}$);
- $\gamma_{ij} \in N$ è il costo da sostenere per diminuire di un'unità la capacità dell'arco (i, j) .

Per questa rete le variabili di decisione $y_{ij} \in [0, \alpha_{ij}]$ ($\alpha_{ij} = \Upsilon_{ij} - v_{ij}$ massima riduzione possibile per l'arco (i, j)) definiscono uno spazio degli stati continuo nel quale $u_{ij} = \Upsilon_{ij} - y_{ij}$. Per reti non orientate si richiederà $y_{ij} = 0$ o $y_{ji} = 0$ (cfr. cap. 3) coerentemente col fatto che non è limitativo cercare il flusso massimo con la condizione $x_{ij} = 0$ o $x_{ji} = 0$. Possiamo allora formulare il problema della Diminuzione Continua del Flusso Massimo (si tratta in verità di una formulazione abbastanza implicita).

Problema 4.2 *Diminuzione Continua del Flusso Massimo*

$$\begin{aligned}
 & \text{minimize} && \nu^*(\Sigma) \\
 & && \Sigma = \{y_{ij}\}_{(i,j) \in A} \\
 & \sum_{(i,j) \in A} \gamma_{ij} y_{ij} && \leq B \\
 & 0 && \leq y_{ij} \leq \alpha_{ij}
 \end{aligned}$$

La difficoltà di formulazione riflette, se si vuole, la difficoltà di risoluzione che si può desumere dal seguente teorema.

Teorema 4.2 *Il Problema 4.2 (Diminuzione Continua del Flusso Massimo) è fortemente NP-difficile.*

Dimostrazione.

Phillips [46] presenta questo risultato mostrando anche come piccole variazioni del *budget* B possano pregiudicare in maniera drammatica l'ottimalità di una soluzione. \diamond

Di fronte a questo risultato seguiamo le orme di Phillips [46] rinunciando a trattare il Problema 4.2 per grafi qualsiasi e limitandoci invece a considerarlo definito su grafi planari.

4.2.1 Grafi Planari

Si è già detto (sez. 2.1) che un grafo G è planare quando lo si può disegnare sul piano senza "incrociare" i suoi archi. Una volta che è stato rappresentato in tale modo esso individua univocamente delle *regioni* di piano: due punti

| | |
|-------|----------------|
| v_d | <i>regione</i> |
| s_d | <i>sbd</i> |
| a_d | <i>bcd</i> |
| b_d | <i>sabc</i> |
| c_d | <i>cdt</i> |
| d_d | <i>act</i> |
| t_d | <i>sat</i> |

Tabella 4.1: Corrispondenza di dualità tra regioni della Figura 2.1 e nodi della Figura 4.1.

appartengono alla stessa regione se e solo se possono essere congiunti da una curva continua che non intersechi alcun arco di G ; in particolare è definita un'unica regione *esterna* di dimensioni infinite. Diremo *st-planare* un grafo i cui nodi s e t giacciono lungo il confine di tale regione esterna (per esempio è *st-planare* il grafo di Figura 2.1 a pag. 14). L'interdizione del flusso (Problema 4.2) su reti planari è comunque un problema di una certa difficoltà, come testimonia il seguente risultato dovuto a Phillips [46].

Teorema 4.3 *Il Problema 4.2 (Diminuzione Continua del Flusso Massimo) definito su un grafo planare è debolmente NP-difficile.*

È però possibile mostrare che la difficoltà del Problema 4.2 si riduce effettivamente nel caso, particolare ma importante, di reti non orientate *st-planari*. A tal fine introduciamo la nozione di *grafo duale* G_d per un grafo planare non orientato G : esso si ottiene associando un vertice a ogni regione e assegnando ogni arco di G alla coppia di regioni che esso separa. La Figura 4.1 mostra il duale del grafo di Figura 2.1 la cui regione esterna è stata idealmente divisa in “sopra” e “sotto” da un ipotetico arco *st* esterno a tutti gli altri (per maggiore chiarezza si riporta in Tabella 4.1 la corrispondenza tra regioni primali e nodi duali). La ragione per cui si divide in due la regione esterna è da ricercarsi nel fatto che in questo modo a ogni *st*-taglio in G corrisponde un cammino da s_d a t_d in G_d : risulta allora evidente che invece di risolvere un problema di taglio minimo (o equivalentemente di flusso massimo) su G è possibile risolvere un problema di cammino minimo su G_d .

Da un punto di vista tecnico rimangono da chiarire due punti: la ricerca di una rappresentazione planare per un grafo assegnato (o al contrario la verifica della sua non planarità) e la costruzione, a partire da questa, del grafo duale. Entrambe queste funzioni possono essere realizzate in tempo $O(n)$ (cioè lineare): si veda al proposito quanto scritto da Hopcroft e Tarjan [32]. Si osservi inoltre che il grafo duale è in generale un multigrafo ma che questo

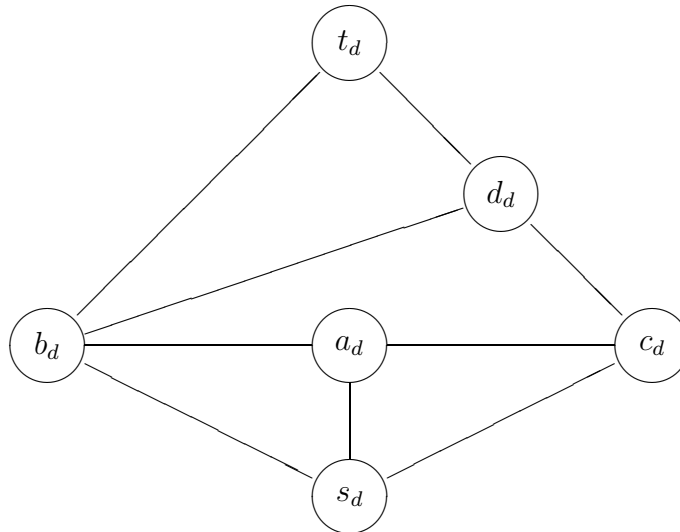


Figura 4.1: Grafo duale di quello mostrato in Figura 2.1.

non ha rilevanza teorica perché sia i suoi nodi che i suoi archi sono $O(n)$. Il ragionamento svolto (che può essere invertito per trasformare un problema di cammino minimo in un problema di flusso massimo) porta all'importante risultato che segue.

Teorema 4.4 *Il problema del flusso massimo su reti st -planari non orientate è equivalente a quello del cammino minimo.*

Come conseguenza del Teorema 4.4 per diminuire o accrescere il flusso massimo su reti non orientate st -planari si possono utilizzare le tecniche sviluppate nel Capitolo 3. In particolare l'esistenza di algoritmi pseudopolinomiali per il Problema 3.5 (Diminuzione Continua del Cammino Minimo) implica che (se $P \neq NP$) il Problema 4.2 (Diminuzione Continua del Flusso Massimo) definito su reti non orientate st -planari *non è fortemente* NP-completo (cfr. Papadimitriou e Steiglitz [44, teo. 16.4]). Per quanto riguarda l'accrescimento del flusso massimo vedremo comunque nella prossima sezione un'approccio diretto valido per reti qualsiasi.

Se si formula il problema della Diminuzione Discreta del Flusso Massimo sul modello del Problema 3.3 (Diminuzione Discreta del Cammino Minimo) esso risulta debolmente NP-completo come conseguenza del Teorema 3.1 e dell'equivalenza stabilita dal Teorema 4.4.

4.3 Accrescimento del Flusso Massimo

È data la rete orientata $G = (V, A; v, \Upsilon; \gamma)$ nella quale:

- $v_{ij} \in N \cup \{0\}$ è la capacità originaria dell'arco (i, j) ;
- $\Upsilon_{ij} \in N \cup \{0\}$ è il valore oltre il quale non è possibile aumentare la capacità dell'arco (i, j) ($\Upsilon_{ij} \geq v_{ij}$);
- $\gamma_{ij} \in N$ è il costo da sostenere per aumentare di un'unità la capacità dell'arco (i, j) .

Per tale rete definiamo lo spazio degli stati continuo individuato dalle variabili di decisione $y_{ij} \in [0, \beta_{ij}]$ ($\beta_{ij} = \Upsilon_{ij} - v_{ij}$ massima crescita possibile per la capacità dell'arco (i, j)). Similmente al caso della diminuzione (sez. 4.2) sarà $u_{ij} = v_{ij} + y_{ij}$ e si richiederà, per reti non orientate, $y_{ij} = 0$ o $y_{ji} = 0$. È allora possibile formulare, a partire dal Problema 4.1 (Flusso Massimo), il seguente problema di Accrescimento Continuo del Flusso Massimo.

Problema 4.3 *Accrescimento Continuo del Flusso Massimo*

$$\begin{aligned}
 & \text{maximize } \nu \\
 & \sum_{j:(v,j) \in A} x_{vj} - \sum_{i:(i,v) \in A} x_{iv} = \begin{cases} +\nu & v = s \\ 0 & v \in V - \{s, t\} \\ -\nu & v = t \end{cases} \\
 & x_{ij} - y_{ij} \leq v_{ij} \\
 & \sum_{(i,j) \in A} \gamma_{ij} y_{ij} \leq B \\
 & 0 \leq y_{ij} \leq \beta_{ij} \\
 & x_{ij} \geq 0
 \end{aligned}$$

Come il Problema 3.6 (Accrescimento Continuo del Cammino Minimo) il Problema 4.3 è un problema di programmazione lineare e perciò da una parte esiste l'algoritmo dell'ellissoide (si veda per es. Grötschel et al. [30, cap. 3]) che testimonia la possibilità di risoluzione in tempo debolmente polinomiale e dall'altra è disponibile il metodo del simplesso (si veda per es. Ahuja et al. [2, app. C]) che permette una risoluzione efficiente da un punto di vista empirico.

Una possibile alternativa è definire per la rete $G = (V, A; v, \Upsilon; \gamma)$ uno spazio degli stati discreto mediante le variabili discrete $y_{ij} \in \{0, 1, \dots, \beta_{ij}\}$. Si ottiene così l'analogo per il flusso massimo del Problema 3.8 (Accrescimento Intero del Cammino Minimo) e l'analogia si spinge al punto che anche

in questo caso è possibile una risoluzione in tempo polinomiale basata su tecniche di ricerca binaria. Tale risoluzione, proposta da Ahuja et al. [2, es. 14.18], affronta preliminarmente il problema del Flusso Intero a Costo Convesso Minimo (formulato nel seguito come Problema 4.5): il resto di questa sezione è dedicato a descriverla in un certo dettaglio.

Problema 4.4 *Accrescimento Intero del Flusso Massimo*

$$\begin{aligned}
 & \text{maximize } \nu \\
 & \sum_{j:(v,j) \in A} x_{vj} - \sum_{i:(i,v) \in A} x_{iv} = \begin{cases} +\nu & v = s \\ 0 & v \in V - \{s, t\} \\ -\nu & v = t \end{cases} \\
 & x_{ij} - y_{ij} \leq v_{ij} \\
 & \sum_{(i,j) \in A} \gamma_{ij} y_{ij} \leq B \\
 & y_{ij} \in \{0, 1, \dots, \beta_{ij}\} \\
 & x_{ij} \in N \cup \{0\}
 \end{aligned}$$

Data una rete orientata $G = (V, A; u, c)$ in cui $c_{ij}(x_{ij})$ è il costo da sostenere affinché l'arco $(i, j) \in A$ conduca un flusso $x_{ij} \in \{0, 1, \dots, u_{ij}\}$ il problema del Flusso Intero a Costo Convesso Minimo consiste nell'instaurare un flusso da s a t di valore $\nu \in N$ assegnato e costo complessivo minimo nell'ipotesi che i costi siano funzioni convesse—oltre che non negative e non decrescenti—dei flussi.

Problema 4.5 *Flusso Intero a Costo Convesso Minimo*

$$\begin{aligned}
 & \text{minimize } \sum_{(i,j) \in A} c_{ij}(x_{ij}) \\
 & \sum_{j:(v,j) \in A} x_{vj} - \sum_{i:(i,v) \in A} x_{iv} = \begin{cases} +\nu & v = s \\ 0 & v \in V - \{s, t\} \\ -\nu & v = t \end{cases} \\
 & x_{ij} \in \{0, 1, \dots, u_{ij}\} \\
 & c_{ij}(\theta x'_{ij} + (1 - \theta) x''_{ij}) \leq \theta c_{ij}(x'_{ij}) + (1 - \theta) c_{ij}(x''_{ij}) \quad \theta \in [0, 1]
 \end{aligned}$$

Il Problema 3.1 di pagina 21 (Flusso Unitario a Costo Minimo) può essere visto come un caso particolare del Problema 4.5 in cui i costi sono lineari ($c_{ij}(x_{ij}) = l_{ij} x_{ij}$), il flusso è di valore unitario ($\nu = 1$) e le capacità sono infinite ($u_{ij} = \infty \quad \forall (i, j) \in A$).

La scelta di considerare flussi interi, cioè valori discreti per le variabili x_{ij} , segue un approccio non convenzionale (Ahuja et al. [2, cap. 14]) che

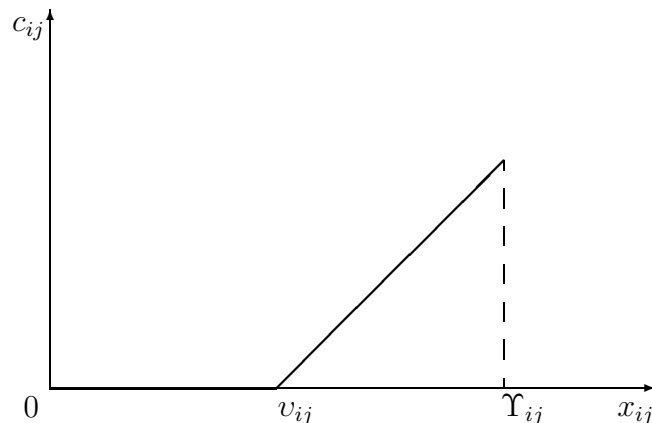


Figura 4.2: Funzione di costo adottata per il Problema 4.5.

è funzionale ad affrontare il Problema 4.4 (Accrescimento Intero del Flusso Massimo). Con tale scelta i costi possono, senza perdita di generalità, essere considerati lineari a tratti (cfr. Figura 4.2).

Ahuja et al. [2, sez. 14.5] presentano un algoritmo (nel seguito indicato con \tilde{A}) che risolve il Problema 4.5 (Flusso Intero a Costo Convesso Minimo) in tempo $O((m \log U) \Pi(n, m, C))$ dove $U = \max_{(i,j) \in A} u_{ij}$ e $\Pi(n, m, C)$ è (cfr. sez. 3.1) la migliore limitazione nota per il calcolo di un cammino minimo in una rete con lunghezze non negative e non maggiori di $C = \max_{ij \in A} c_{ij}(u_{ij})$. L'algoritmo \tilde{A} è—debolmente—polinomiale nella dimensione dell'input sia che i costi siano dati in forma chiusa ($\dim\langle c_{ij}(x_{ij}) \rangle \in O(1)$) sia che essi siano dati per punti ($\dim\langle c_{ij}(x_{ij}) \rangle \in O(C)$).

In particolare adotteremo per il Problema 4.5 le funzioni di costo

$$c_{ij}(x_{ij}) = \begin{cases} 0 & x_{ij} \in \{0, 1, \dots, v_{ij}\} \\ \gamma_{ij}(x_{ij} - v_{ij}) & x_{ij} \in \{v_{ij} + 1, v_{ij} + 2, \dots, \Upsilon_{ij}\} \end{cases}$$

e si vede immediatamente (Figura 4.2) che si tratta di funzioni convesse e che esse corrispondono alla situazione del Problema 4.4: un flusso di valore ν e costo $\Gamma \leq B$ è una limitazione inferiore alla soluzione ottima per l'Accrescimento Intero del Flusso Massimo. Organizzando una ricerca binaria sul valore (intero) del flusso è allora possibile utilizzare l'algoritmo \tilde{A} per risolvere il Problema 4.4 (Accrescimento Intero del Flusso Massimo).

Algoritmo 4.1 Accrescimento Intero del Flusso Massimo

1. Poni $\nu_m := \nu^*(\Sigma_0)$ (flusso massimo originario:
 $B = 0 \Rightarrow u_{ij} = v_{ij} \forall (i, j) \in A$).
2. Poni $\nu_M := \nu^*(\Sigma_\infty)$ (flusso massimo senza vincolo
di bilancio: $B = \infty \Rightarrow u_{ij} = \Upsilon_{ij} \forall (i, j) \in A$).
3. Ripeti i seguenti passi
 - (a) Poni $\nu := \lfloor (\nu_m + \nu_M)/2 \rfloor$;
 - (b) Utilizza l'algoritmo \tilde{A} per determinare
il minimo costo Γ di un flusso avente valore ν ;
 - (c) Se $\Gamma \leq B$ poni $\nu^{OPT} := \nu$ e $\nu_m := \nu$
altrimenti poni $\nu_M := \nu$;finché non è $\nu_m = \nu_M$.
4. A partire dal flusso a costo minimo di valore ν^{OPT}
determina Σ^{OPT} ponendo $y_{ij} := \max\{0, x_{ij} - v_{ij}\}$.

Il risultato che segue riguarda l'efficacia e l'efficienza dell'algoritmo descritto.

Teorema 4.5 *L'Algoritmo 4.1 fornisce una soluzione ottima per il Problema 4.4 in tempo $O(m \log U \log nU \Pi(n, m, C))$.*

Dimostrazione.

La ricerca binaria si può fare perché Γ è funzione non decrescente di ν (costi non negativi e non decrescenti); l'intervallo di valori considerato per il flusso garantisce la correttezza dell'algoritmo. Per reti non orientate occorre considerare flussi netti ($x_{ij} \geq x_{ji} \Rightarrow x_{ij} := x_{ij} - x_{ji} \ \& \ x_{ji} := 0$) di modo che sia $y_{ij} = 0$ o $y_{ji} = 0$ come richiesto dal caso in questione.

La complessità del punto 3 corrisponde a $O(\log mU)$ esecuzioni dell'algoritmo \tilde{A} ($mU < n^2U$ è un'ovvia limitazione superiore per il valore del flusso) e di conseguenza gli altri punti possono essere trascurati. \diamond

Accrescere sotto un vincolo di bilancio il flusso massimo sostenibile da una rete è quindi un'operazione che può essere pianificata in modo ottimo mediante un algoritmo efficiente sia nel caso di flusso continuo (Problema 4.3) che in quello di flusso intero (Problema 4.4). Se invece si formula l'Accrescimento Discreto del Flusso Massimo nella sua forma più generale (S_{ij} stati per ogni arco) esso risulta, quando formulato su reti non orientate st -planari, equivalente all'Accrescimento Discreto del Cammino Minimo (per il Teorema 4.4) e di conseguenza (cfr. sez. 3.5) si ottiene un problema debolmente NP-difficile.

| | Diminuzione | Accrescimento |
|-------------------|---|--|
| Modalità Discreta | deb. NP-difficile (sez. 4.2) per reti <i>st</i> -planari non or. equiv. cam. min. (teo. 4.4) | deb. NP-difficile, ma caso Intero (prob. 4.4) risol. pol. (alg. 4.1) |
| Modalità Continua | fort. NP-difficile (teo. 4.2) per reti <i>st</i> -planari non or. equiv. cam. min. (teo. 4.4) | Prog. Lin. (prob. 4.3) |

Tabella 4.2: Quadro riassuntivo dei problemi di flusso massimo.

4.4 Prospettive e Sviluppi

Questo capitolo è stato dedicato ad alcuni problemi di ottimizzazione su grafi che sorgono quando si voglia modificare il valore del flusso massimo di una rete intervenendo, sotto un vincolo di bilancio, sugli archi della medesima. Come nel caso del cammino minimo è possibile classificare tali problemi in funzione del loro *obiettivo* (diminuzione o accrescimento) e in funzione della *modalità di intervento* prevista (discreta o continua). La Tabella 4.2, a cui il seguito della sezione fa da commento, raccoglie i risultati del capitolo proprio secondo questa classificazione.

Per quanto riguarda i problemi di diminuzione del flusso massimo ci siamo sostanzialmente limitati a ricondurli, nel caso particolare ma significativo di reti non orientate *st*-planari, ai corrispondenti problemi di cammino minimo. La formulazione del Problema 4.2 (Diminuzione Continua del Flusso Massimo) è dovuta a Phillips [46] come del resto i relativi risultati di forte e debole NP-difficoltà rispettivamente per il caso generale (Teorema 4.2) e per quello planare (Teorema 4.3). Phillips [46] fornisce anche un algoritmo per reti non orientate *st*-planari che si basa sulla nozione di grafo duale senza che sia resa esplicita l'equivalenza tra flusso massimo e cammino minimo nelle reti suddette: la risoluzione dei problemi di diminuzione del cammino minimo nel Capitolo 3 è ispirata proprio a questo algoritmo. Eventuali ricerche future potranno essere volte a risolvere il Problema 4.2 e la Diminuzione Discreta del Cammino Minimo nel caso più generale di reti qualsiasi.

Il Problema 4.3 (Accrescimento Continuo del Flusso Massimo) non risulta formulato in precedenza ma poiché è un problema di programmazione lineare la sua risoluzione non richiede particolare sforzo. Il Problema 4.4 (Accrescimento Intero del Flusso Massimo) è invece stato suggerito da Ahuja et al. [2] sotto forma di esercizio, assieme alle tecniche su cui si basa l'Algoritmo 4.1 che lo risolve. La forma generale dell'Accrescimento Discreto del Flusso Massimo è d'altra parte tutta da studiare alla luce della sua debole NP-difficoltà.

Capitolo 5

Problemi di Minimo Albero Ricoprente

Questo capitolo tratta il caso in cui l'efficienza della rete possa essere misurata dal minimo peso di un albero che connetta tutti i suoi nodi.

Similmente a quanto visto nei due capitoli precedenti per i problemi di cammino minimo e di flusso massimo nella prima sezione viene introdotto il classico problema di calcolare il minimo albero ricoprente per una rete assegnata mentre nell'ultima vengono riassunti i risultati ottenuti sottolineandone gli aspetti di originalità.

Nella seconda sezione si espone invece una variante del minimo albero ricoprente in cui sono presenti due funzioni peso distinte: poiché non è possibile in generale minimizzarle entrambe si realizza con una delle due un vincolo di bilancio sotto il quale minimizzare l'altra.

La terza e la quarta sezione studiano infine la possibilità di diminuire e, rispettivamente, aumentare in modo ottimo la misura di efficienza prescelta secondo modalità analoghe a quelle già viste per i casi del cammino minimo e del flusso massimo.

5.1 Introduzione al Minimo Albero Ricoprente

Data una rete *non* orientata $G = (V, A; w)$, dove $w : A \rightarrow N \cup \{0\}$ è una funzione peso per i suoi archi, definiamo peso di un albero $T = (V_T, A_T) \leq G$ (si veda sez. 2.1) il valore $w(T) = \sum_{ij \in A_T} w_{ij}$ e cerchiamo un albero che abbia peso minimo tra tutti quelli per i quali è $V_T = V$ (ossia tra tutti quelli che ricoprono G). Indicheremo con w^* tale peso minimo e sarà di conseguenza

$mis(G) = w^*$ la misura di efficienza adottata in questo capitolo. Osserviamo che si può equivalentemente parlare di massimo albero ricoprente effettuando la sostituzione $w_{ij} := W - w_{ij}$ dove $W = \max_{ij \in A} w_{ij}$ (ogni albero ricoprente consta esattamente di $n - 1$ archi—cfr. sez. 2.1). Inoltre è possibile definire il problema anche su un multigrafo ma, essendo i *cappi* inutili e potendosi trascurare gli archi paralleli di peso maggiore, non si guadagna in generalità.

Il problema del minimo albero ricoprente può anche essere formulato in termini di programmazione lineare semplicemente definendo per ogni $U \subseteq V$ $A(U) = \{ij \in A \mid i \in U \ \& \ j \in U\}$ ($A(V) = A$) e interpretando $ij \in A_T$ per $x_{ij} = 1$, $ij \notin A_T$ per $x_{ij} = 0$.

Problema 5.1 *Minimo Albero Ricoprente*

$$\begin{aligned} \text{minimize} \quad & \sum_{ij \in A} w_{ij} x_{ij} \\ & \sum_{ij \in A} x_{ij} = n - 1 \\ & \sum_{ij \in A(U)} x_{ij} \leq |U| - 1 \quad \forall U \subset V \\ & x_{ij} \geq 0 \end{aligned}$$

Questa formulazione è molto interessante perché il poliedro convesso delle soluzioni ammissibili ha vertici interi (cfr. per es. Ahuja et al. [2, teo. 13.10]) ma è di scarsa utilità pratica dal momento che mentre per richiedere la connessione di T è sufficiente un vincolo (il primo) per richiederne l'aciclicità se ne usano un numero esponenziale (uno per ogni sottoinsieme di V).

Dato un albero ricoprente $T = (V, A_T)$ di $G = (V, A)$ risulta automaticamente definito il suo *co-albero* $\bar{T} = (V, A_{\bar{T}})$ con $A_{\bar{T}} = A - A_T$. Si dice *ciclo fondamentale* di un arco ij del co-albero l'unico ciclo $CF(ij)$ che si ottiene aggiungendo tale arco all'albero. Analogamente si dice *taglio fondamentale* di un arco ij dell'albero il taglio $TF(ij)$ definito in G (cfr. sez. 4.1) dalle due componenti connesse che si ottengono rimuovendo ij dall'albero. Nel seguito si indicherà brevemente con T sia un albero ricoprente sia l'insieme dei suoi archi, restando sottinteso che l'insieme dei nodi è V ; analogamente per il co-albero corrispondente \bar{T} . È allora possibile caratterizzare in modo utile e semplice i minimi alberi ricoprenti di una rete (per le dimostrazioni si può vedere per es. Ahuja et al. [2, cap. 13]).

Teorema 5.1 (Condizioni di Ottimalità sui Cicli) *Un albero ricoprente $T \leq G$ è minimo se e solo se ogni arco ij del co-albero \bar{T} ha peso massimo nel suo ciclo fondamentale ($w_{ij} \geq w_{hk} \ \forall hk \in CF(ij)$).*

Teorema 5.2 (Condizioni di Ottimalità sui Tagli) *Un albero ricoprente $T \leq G$ è minimo se e solo se ogni arco $ij \in T$ ha peso minimo nel suo taglio fondamentale ($w_{ij} \leq w_{hk} \forall hk \in TF(ij)$).*

Il modo più semplice per risolvere il Problema 5.1 è cominciare con $T = \emptyset$ e considerare gli archi di G in ordine crescente di peso: si pone $T := T \cup \{ij\}$ se e solo se non si crea un ciclo in T . La correttezza di questo approccio, generalmente attribuito a Kruskal [40], è garantita essenzialmente dal Teorema 5.1 (Condizioni di Ottimalità sui Cicli) mentre per quanto riguarda la complessità si deve ricordare a parte che ordinare m numeri richiede $\Theta(m \log m)$ confronti (si veda per es. Ausiello et al. [9, cap. 4]). Se gli archi vengono forniti già ordinati il migliore algoritmo conosciuto per il Problema 5.1 è un'elaborazione dovuta a Tarjan [52] dell'algoritmo di Kruskal avente complessità $O(m \alpha(n, m))$ ($\alpha(n, m)$ è una funzione inversa di Ackermann che cresce tanto lentamente da aversi $\alpha(n, m) \leq 6$ per valori pratici di n e m); altrimenti, nel caso generale di archi non ordinati, il migliore algoritmo conosciuto ha complessità $m \log \beta(m, n)$ ($\beta(m, n) = \min\{i \mid \log^i(m/n) \leq 1\}$ cosicché $\beta(m, n) \leq 6$ per valori pratici di m e n) ed è stato sviluppato da Gabow et al. [24] a partire da quello che è noto come algoritmo di Prim [48]. Indicheremo nel seguito parametricamente con $\Psi(n, m)$ una generica limitazione di complessità per il problema del minimo albero ricoprente, rinviando ad Ahuja et al. [2, cap. 13] per una panoramica sugli algoritmi disponibili in letteratura (tutti, di fatto, *fortemente* polinomiali).

Una rassegna delle molteplici applicazioni per il problema del minimo albero ricoprente può essere trovata in Ahuja et al. [2] ma vale la pena esplicitare un esempio che evidenzia come w^* possa misurare le prestazioni di una rete. Sia G una rete di comunicazione soggetta ad “attacchi” in grado di mettere fuori uso tutti i suoi archi (es. bombardamenti per una rete stradale o “virus” per una rete di elaboratori) e sia w_{ij} il tempo necessario a ripristinare l'arco ij dopo uno di tali attacchi: se tutti i nodi di G hanno la stessa importanza allora il tempo minimo per ripristinare la funzionalità della rete (intesa come connessione di tutti i suoi nodi) è pari a w^* , almeno nell'ipotesi prudente di ripristino sequenziale degli archi; considerare l'efficienza di G decrescente (piuttosto che crescente) all'aumentare di $mis(G) = w^*$ significa assumere il punto di vista del difensore (piuttosto che quello dell'attaccante). Si osservi che se per il decisore la funzione della rete è di connettere due nodi di particolare importanza (s e t) la stessa situazione è meglio modellata da un problema di cammino minimo: si vede allora come il punto di vista del decisore sia fondamentale per la scelta di $mis(G)$.

5.2 Il Minimo Albero Ricoprente Vincolato

Si consideri la rete non orientata $G = (V, A; w', w'')$ costituita da un grafo con due funzioni peso per i suoi archi: in generale non sarà possibile trovare un albero ricoprente $T \leq G$ che sia minimo rispetto a entrambi i pesi. Un modo per superare questa impossibilità è fissare un valore M di *soglia* per uno dei due pesi—es. $w'(T)$ —e minimizzare l'altro— $w''(T)$ —compatibilmente con il vincolo posto.

Problema 5.2 *Minimo Albero Ricoprente Vincolato*

$$\begin{aligned}
 & \text{minimize} && \sum_{ij \in A} w''_{ij} x_{ij} \\
 & && \sum_{ij \in A} x_{ij} = n - 1 \\
 & && \sum_{ij \in A(U)} x_{ij} \leq |U| - 1 \quad \forall U \subset V \\
 & && \sum_{ij \in A} w'_{ij} x_{ij} \leq M \\
 & && x_{ij} \in \{0, 1\}
 \end{aligned}$$

Questo problema è stato studiato inizialmente da Aggarwal et al. [1] che ne hanno dimostrato la NP-difficoltà proponendo di conseguenza un'euristica per la sua risoluzione esatta.

Teorema 5.3 *Il Problema 5.2 (Minimo Albero Ricoprente Vincolato) è debolmente NP-difficile.*

Successivamente Marathe et al. [42] hanno proposto un algoritmo debolmente polinomiale che è 0.5-approssimato e 0.5-ammissibile, ovvero fornisce una soluzione T^A tale che $w''(T^A) \leq 2 w''(T^{OT})$ e inoltre $w'(T^A) \leq 2 M$ (T^{OT} è una soluzione ottima per il Problema 5.2).

Infine Ravi e Goemans [50] hanno sviluppato lo schema di approssimazione fortemente polinomiale di cui il seguente Teorema 5.4 afferma l'esistenza.

Teorema 5.4 *Per ogni $\epsilon \in (0, 1)$ esiste un algoritmo $A'_{RG}(\epsilon)$ di complessità $O(n^{1/\epsilon} \log^2(n) \Psi(n, m))$ che fornisce una soluzione ottimale ϵ -ammissibile per il Problema 5.2 ($w''(T^A) \leq w''(T^{OT})$ & $w'(T^A) \leq M/(1 - \epsilon)$).*

Gli stessi Ravi e Goemans [50] ricavano da $A'_{RG}(\epsilon)$ un secondo schema di approssimazione, debolmente polinomiale, che descriviamo nel Teorema 5.5.

Teorema 5.5 Per ogni $\epsilon \in (0, 1)$ esiste un algoritmo $A''_{RG}(\epsilon)$ di complessità $O(\log(nW) n^{1/\epsilon} \log^2(n) \Psi(n, m))$ che trova una soluzione ammissibile ϵ -approssimata al Problema 5.2 ($w'(T^A) \leq M$ & $w''(T^A) \leq w''(T^{OT})/(1 - \epsilon)$).

Dimostrazione.

È sufficiente scambiare di ruolo w' e w'' e applicare l'algoritmo $A'_{RG}(\epsilon)$ facendo variare la soglia (che qui chiameremo S) nell'insieme dei valori ammissibili per $w''(T)$. Per mezzo di una ricerca binaria si può allora individuare efficientemente il minimo valore S_{min} per cui $w'(T^A) \leq M$: la soluzione T^A così trovata è una soluzione ammissibile per la quale $w''(T^A) \leq S_{min}/(1 - \epsilon)$ e inoltre $w''(T) < S_{min} \Rightarrow w'(T) > M$. La complessità è determinata da $O(\log(nW))$ esecuzioni di $A'_{RG}(\epsilon)$ poichè in ogni caso $w''(T) \leq nW$. \diamond

5.3 Diminuzione del Minimo Albero Ricoprente

Sia data la rete non orientata $G = (V, A; w^0, w^1, \dots, w^{S-1}; c^0, c^1, \dots, c^{S-1})$ dove $w^k_{ij} \in N \cup \{0\}$ è il peso dell'arco $ij \in A$ quando questo si trovi nello stato k e $c^k_{ij} \in N \cup \{0\}$ è il costo da sostenere per portarlo in tale stato. Per ogni arco ij possiamo ordinare i suoi pesi w^k_{ij} nei diversi stati in ordine decrescente con k e, di conseguenza, avere i relativi costi c^k_{ij} che crescono con k : infatti se $w^h_{ij} \leq w^k_{ij}$ e $c^h_{ij} \leq c^k_{ij}$ lo stato k può essere ignorato preferendogli comunque lo stato h non meno vantaggioso sia come peso che come costo. Coerentemente $k = 0$ individua lo stato iniziale ed è perciò $c^0_{ij} = 0 \forall ij \in A$. Se poi introduciamo le variabili di decisione $y_{ij} \in \{0, 1, \dots, S_{ij} - 1\}$ ($S_{ij} \leq S$ è il numero di stati disponibili per l'arco ij) otteniamo uno spazio degli stati discreto (con $y_{ij} = k$ si pone ij nello stato k). In tale spazio sono ammissibili quegli stati che soddisfano il vincolo $\sum_{ij \in A} c^{y_{ij}}_{ij} \leq B$ sotto il quale si cerca di diminuire in modo ottimo $mis(G, \Sigma) = w^*(G, \Sigma)$.

Problema 5.3 *Diminuzione Discreta del Minimo Albero Ricoprente*

$$\begin{aligned}
 & \text{minimize} && \sum_{ij \in A} w_{ij}^{y_{ij}} x_{ij} \\
 & \sum_{ij \in A} x_{ij} &= & n - 1 \\
 & \sum_{ij \in A(U)} x_{ij} &\leq & |U| - 1 \quad \forall U \subset V \\
 & \sum_{ij \in A} c_{ij}^{y_{ij}} &\leq & B \\
 & x_{ij} &\in & \{0, 1\} \\
 & y_{ij} &\in & \{0, 1, \dots, S_{ij} - 1\}
 \end{aligned}$$

Di questo problema sembra essere presente in letteratura solo una variante in cui gli interventi migliorativi discreti sono da effettuarsi sui nodi piuttosto che sugli archi. In questa variante lo stato di un arco è determinato dallo stato dei nodi che congiunge, i quali possono venire lasciati inalterati oppure migliorati. Paik e Sahni [43] riducono di un fattore $\delta \in (0, 1)$ il peso w_{ij} per ognuno degli estremi di ij che è stato migliorato. Krumke et al. [38] considerano più in generale tre valori per il peso di ij a seconda che nessuno, uno o due tra i e j siano stati migliorati: per tale problema propongono un algoritmo polinomiale approssimato che si dimostra essere ottimo sotto opportune ipotesi.

Il Problema 5.3 può vedersi come un caso particolare del Problema 5.2 (Minimo Albero Ricoprente Vincolato) formulato sul multigrafo che si ottiene da G sostituendo a ogni arco $ij \in A$ gli S_{ij} archi corrispondenti ai suoi possibili stati; in questo caso $w'_{ij(k)} = c_{ij}^k$ e $w''_{ij(k)} = w_{ij}^k$ per ogni arco $ij(k)$ del multigrafo e inoltre $M = B$. Questa interpretazione è corretta perché non è affatto riduttivo (anzi è consigliabile) spendere il *budget* B solo su quegli archi che appartengano, nella rete migliorata, al minimo albero ricoprente; di conseguenza si può parlare indifferentemente di stato Σ o di albero T per riferirsi a una soluzione del Problema 5.3. Tale Problema 5.3 risulta allora non più facile del Problema 5.2 e quindi dalla NP-difficoltà di quest'ultimo (Teorema 5.3) segue il Teorema 5.6.

Teorema 5.6 *Il Problema 5.3 (Diminuzione Discreta del Minimo Albero Ricoprente) è debolmente NP-difficile.*

Un'analisi attenta degli schemi di approssimazione proposti da Ravi e Goemans [50] permette di verificare la loro validità anche per la risoluzione di problemi definiti su multigrafi, onde il teorema che segue.

Teorema 5.7 *Esiste uno schema di approssimazione polinomiale $A'_{RG}(\epsilon)$ che in tempo $O(n^{1/\epsilon} \log^2(nS) \Psi(n, mS))$ fornisce soluzioni ottimali ϵ -ammissibili per il Problema 5.3. Esiste inoltre uno schema di approssimazione polinomiale $A''_{RG}(\epsilon)$ che in tempo $O(\log(nW) n^{1/\epsilon} \log^2(nS) \Psi(n, mS))$ fornisce soluzioni ammissibili ϵ -approssimate per il Problema 5.3.*

Dimostrazione.

Le limitazioni sulla complessità computazionale sono state modificate tenendo conto che l'input in questo caso è un multigrafo con n nodi e al più mS archi. Per ottenere lo stato Σ^A che corrisponde all'albero ricoprente T^A è sufficiente porre $y_{ij} := k$ se $y_{ij(k)} \in T^A$ e $y_{ij} = 0$ altrimenti. \diamond

Vediamo adesso una versione continua del Problema 5.3 definita sulla rete non orientata $G = (V, A; \Xi, \xi; \gamma)$ nella quale:

- $\Xi_{ij} \in N \cup \{0\}$ è il peso originario dell'arco $ij \in A$;
- $\xi_{ij} \in N \cup \{0\}$ è il peso minimo per l'arco $ij \in A$ ($\xi_{ij} \leq \Xi_{ij}$);
- $\gamma_{ij} \in N$ è il costo unitario per ridurre il peso dell'arco $ij \in A$.

Introduciamo le variabili di decisione $y_{ij} \in [0, \alpha_{ij}]$ ($\alpha_{ij} = \Xi_{ij} - \xi_{ij}$ massima riduzione possibile per il peso dell'arco ij) e definiamo così uno spazio degli stati continuo ($y_{ij} = a$ significa $w_{ij} = \Xi_{ij} - a$). Tra gli stati che soddisfano il vincolo di bilancio $\sum_{ij \in A} \gamma_{ij} y_{ij} \leq B$ se ne vuole trovare uno Σ^{OT} nel quale $w^*(\Sigma^{OT})$ sia minimo.

Problema 5.4 *Diminuzione Continua del Minimo Albero Ricoprente*

$$\begin{aligned}
& \text{minimize} && \sum_{ij \in A} (\Xi_{ij} - y_{ij}) x_{ij} \\
& \sum_{ij \in A} x_{ij} &= & n - 1 \\
& \sum_{ij \in A(U)} x_{ij} &\leq & |U| - 1 \quad \forall U \subset V \\
& \sum_{ij \in A} \gamma_{ij} y_{ij} &\leq & B \\
& x_{ij} &\in & \{0, 1\} \\
& y_{ij} &\in & [0, \alpha_{ij}]
\end{aligned}$$

Krumke et al. [39] hanno studiato precedentemente questo problema dimostrando in primo luogo il risultato che segue.

Teorema 5.8 *Il Problema 5.4 (Diminuzione Continua del Minimo Albero Ricoprente) è debolmente NP-difficile.*

A fronte di tale risultato Krumke et al. [39] presentano uno schema di approssimazione pienamente polinomiale in grado di trovare per ogni $\epsilon \in (0, 1)$ una soluzione del Problema 5.4 che è $(1 - \epsilon)$ -ammissibile e “quasi” ϵ -approssimata. Infatti esso, per ogni $\kappa > 0$, fornisce in tempo $O(\log(nW/\epsilon\kappa)\Psi(n, m))$ ($W = \max_{ij \in A} \Xi_{ij}$) una soluzione $\Sigma^A = \{y_{ij}^A\}$ tale che $c(\Sigma^A) \leq B/\epsilon$ e d'altra parte $w^*(\Sigma^A) \leq w^*(\Sigma^{OT})/(1 - \epsilon) + \kappa$. Un siffatto schema di approssimazione non è pienamente soddisfacente perché non permette di garantire l'approssimazione sulla funzione obiettivo indipendentemente da quella sul vincolo di bilancio; esso ha però il pregio di essere valido per funzioni di costo più generali di quelle lineari ($c_{ij}(y_{ij}) = \gamma_{ij} y_{ij}$) che sono qui oggetto di studio.

Analogamente a quanto visto per il Problema 3.5 (Diminuzione Continua del Cammino Minimo—pag. 32) se fosse noto a priori un albero ricoprente T minimo per la rete G nello stato Σ^{OT} potremmo senz'altro ricavare tale stato spendendo il *budget* B sugli archi di T in ordine crescente di costo unitario γ_{ij} (fino all'esaurimento di B o alla massima riduzione possibile per T). Anche le soluzioni del Problema 5.4 (Diminuzione Continua del Minimo Albero Ricoprente) possono allora essere individuate indifferentemente da uno stato Σ o da un albero T .

Nel caso particolare in cui G ha una topologia ad albero vi è un unico albero ricoprente $T = G$ e il Problema 5.4 si riduce così al Problema 2.3 (Bisaccia Continua) risolvibile esattamente in tempo polinomiale con la strategia *greedy* appena descritta (cfr. sez. 2.2); Più in generale possiamo utilizzare l'osservazione fatta per ridurre l'insieme delle soluzioni ammissibili potenzialmente ottime:

1. Tutti gli archi di T vengono o lasciati inalterati ($y_{ij} = 0$) oppure *completati* ($y_{ij} = \alpha_{ij}$); chiameremo questo tipo di soluzione *tutto/niente*.
2. Su un solo arco $qr \in T$ l'intervento viene *parzializzato*, quello durante il miglioramento del quale, nel corso della strategia *greedy*, si esaurirebbe il *budget* B .

Il minimo di $w^*(\Sigma)$ può essere raggiunto considerando solo soluzioni dei due tipi descritti (che prevedono interventi migliorativi solo sugli archi di un albero) e perciò non è riduttivo trascurare soluzioni di altro tipo.

Per soluzioni di tipo *tutto/niente* le variabili di decisione y_{ij} assumono tutte valori interi mentre nel caso vi sia un arco qr parzializzato la relativa variabile y_{qr} ha in generale valore razionale. Se però si considerano le variabili di decisione alternative $z_{ij} = \gamma_{ij} y_{ij} \in [0, c_{ij}]$ ($c_{ij} = \gamma_{ij} \alpha_{ij}$ è il *costo di completamento* dell'arco ij) esse sono tutte intere in entrambi i casi.

La ricerca della migliore soluzione di tipo *tutto/niente* può essere vista come un caso particolare del Problema 5.3 (Diminuzione Discreta del Minimo Albero Ricoprente) in cui $S_{ij} = S = 2 \forall ij \in A$ e inoltre $w_{ij}^0 = \Xi_{ij}$, $w_{ij}^1 = \xi_{ij}$ e $c_{ij}^1 = c_{ij}$. Inoltre se $qr \in A$ è un arco parzializzato in T allora $T - \{qr\}$ è una soluzione *tutto/niente* nella rete che si ottiene da G *contraendo* l'arco qr (ovvero eliminandolo e considerando q e r come un solo nodo).

Il seguente Algoritmo 5.1 sfrutta le considerazioni svolte per realizzare un schema di approssimazione pseudo-polinomiale per il Problema 5.4 nel senso chiarito dal successivo Teorema 5.9.

Algoritmo 5.1 Diminuzione Continua
del Minimo Albero Ricoprente I

1. Calcola mediante l'algoritmo $A'_{RG}(\epsilon)$ del Teorema 5.7 una soluzione T_1^A ϵ -ammissibile e ottimale tra quelle del tipo *tutto/niente*.
2. Calcola (con i seguenti passi) per ogni arco $ij \in A$ una soluzione T_{ij}^A ϵ -ammissibile e ottimale nell'ipotesi che tale arco ij sia parzializzato:
 - (a) Calcola (con i seguenti passi) $\forall b \in \{1, 2, \dots, K\}$ ($K = \min\{c_{ij} - 1, B\}$) una soluzione $T_{ij}^A(b)$ ϵ -ammissibile e ottimale nell'ipotesi che sia $z_{ij} = b$:
 - i. Considerando un budget $B - b$ calcola mediante l'algoritmo $A'_{RG}(\epsilon)$ del Teorema 5.7 una soluzione $T^A[ij, b]$ ϵ -ammissibile e ottimale tra quelle di tipo *tutto/niente* della rete che si ottiene da G *contraendo* l'arco ij .
 - ii. Poni $T_{ij}^A(b) := T^A[ij, b] \cup \{ij\}$.
 - (b) Prendi T_{ij}^A tale che sia $w(T_{ij}^A) = \min_{1 \leq b \leq K} w(T_{ij}^A(b))$.
3. Prendi T_2^A tale che sia $w(T_2^A) = \min_{ij \in A} w(T_{ij}^A)$.
4. Prendi T^A tale che sia $w(T^A) = \min_{k=1,2} w(T_k^A)$.
5. Determina Σ^A a partire da T^A .

Teorema 5.9 L'Algoritmo 5.1 (Diminuzione Continua del Minimo Albero Ricoprente I) trova una soluzione ottimale ϵ -ammissibile per il Problema 5.4 (ovvero $c(\Sigma^A) \leq B/(1 - \epsilon)$ & $w^*(\Sigma^A) \leq w^*(\Sigma^{OT})$). La sua complessità computazionale è $O(mB n^{1/\epsilon} \log^2(n) \Psi(n, m))$.

Dimostrazione.

Tutte le soluzioni generate sono ϵ -ammissibili poiché

$$c(T_{ij}^A(b)) = b + c(T^A[ij, b]) \leq b + (B - b)/(1 - \epsilon) \leq B/(1 - \epsilon).$$

Se Σ^{OT} è del tipo *tutto/niente* allora

$$w(T_1^A) \leq w^*(\Sigma^{OT}).$$

Se T^{OT} ha l'arco qr parzializzato con $z_{qr}^{OT} = b$ allora

$$w(T^A[qr, b]) \leq w(T^{OT} - \{qr\}) \Rightarrow w(T_{qr}^A(b)) \leq w(T^{OT}).$$

La selezione di T^A mediante operazioni di minimo garantisce la sua ottimalità.

Per quanto riguarda invece la complessità computazionale si hanno chiaramente $1 + \sum_{ij \in A} K(ij) \in O(mB)$ esecuzioni dell'algoritmo $A'_{RG}(\epsilon)$. \diamond

L'Algoritmo 5.2 che segue è ricavato dall'Algoritmo 5.1 mediante una tecnica di "scalamento" dei dati che lo rende uno schema di approssimazione *polinomiale* nel senso chiarito dal successivo Teorema 5.10.

Algoritmo 5.2 PTAS-I per la Diminuzione Continua del Minimo Albero Ricoprente

1. Poni $\eta := \epsilon/(2 - \epsilon)$.
2. Calcola mediante l'algoritmo $A'_{RG}(\epsilon)$ del Teorema 5.7 una soluzione T_1^A ϵ -ammissibile e ottimale tra quelle del tipo *tutto/niente*.
3. Poni $t := \max\{0, \lfloor \log(\eta B) \rfloor\}$.
4. Calcola (con i seguenti passi) per ogni arco $ij \in A$ una soluzione T_{ij}^A ϵ -ammissibile e ottimale nell'ipotesi che tale arco ij sia parzializzato:
 - (a) Calcola (con i seguenti passi) per ogni $b \in \{1, 2, \dots, K\}$ ($K = \min\{c_{ij} - 1, B\}$) che sia multiplo di 2^t (e in ogni caso per $b = K$) una soluzione $T_{ij}^A(b)$ ϵ -ammissibile e ottimale nell'ipotesi che sia $z_{ij} = b$:

i. Considerando un budget $B - b + 2^t - 1$ calcola mediante l'algoritmo $A'_{RG}(\eta)$ del Teorema 5.7 una soluzione $T^A[ij, b]$ η -ammissibile e ottimale tra quelle di tipo tutto/niente della rete che si ottiene da G contraendo l'arco ij .

ii. Poni $T_{ij}^A(b) := T^A[ij, b] \cup \{ij\}$.

(b) Prendi T_{ij}^A tale che sia $w(T_{ij}^A) = \min_{1 \leq b \leq K} w(T_{ij}^A(b))$.

5. Prendi T_2^A tale che sia $w(T_2^A) = \min_{ij \in A} w(T_{ij}^A)$.

6. Prendi T^A tale che sia $w(T^A) = \min_{k=1,2} w(T_k^A)$.

7. Determina Σ^A a partire da T^A .

Teorema 5.10 *L'Algoritmo 5.2 (PTAS-I per la Diminuzione Continua del Minimo Albero Ricoprente) trova una soluzione ottimale ϵ -ammissibile per il Problema 5.4 (ovvero $c(\Sigma^A) \leq B/(1 - \epsilon)$ & $w^*(\Sigma^A) \leq w^*(\Sigma^{OT})$). La sua complessità computazionale è $O((m/\epsilon) n^{1/\epsilon} \log^2(n) \Psi(n, m))$.*

Dimostrazione.

Se $\eta B < 2$ l'Algoritmo 5.2 prende $t = 0$ e si riduce all'Algoritmo 5.1 ($\eta < \epsilon$).

Le soluzioni generate sono tutte ϵ -ammissibili perché

$$c(T_{ij}^A(b)) = b + c(T^A[ij, b]) \leq b + \frac{B - b + 2^t - 1}{1 - \eta} \leq \frac{B + \eta B}{1 - \eta} = \frac{B}{1 - \epsilon}$$

in quanto $\eta B \geq 2 \Rightarrow t = \lfloor \log(\eta B) \rfloor$.

Se esiste Σ^{OT} del tipo tutto/niente allora T_1^A è ottimale.

Se invece qr è l'arco parzializzato di T^{OT} e $z_{qr}^{OT} = \beta \in \{1, \dots, K(qr)\}$ allora viene calcolata $T_{qr}^A(b)$ per $b = \min\{\lceil \beta/2^t \rceil 2^t, K(ij)\}$ che è ottimale perché essendo $b - 2^t + 1 \leq \beta \leq b$ si ha

$$c(T^{OT} - \{qr\}) = B - \beta \leq B - b + 2^t - 1 \Rightarrow w(T^A[qr, b]) \leq w(T^{OT} - \{qr\})$$

e quindi

$$w(T_{qr}^A(b)) \leq w(T^{OT}).$$

Per quanto riguarda la complessità è $1/\eta \in O(1/\epsilon)$ e $\eta B < 2^{t+1}$ (se $t = 0$ perché $\eta B < 2$ altrimenti perché $t + 1 > \log(\eta B)$); di conseguenza le $O(mB/2^t)$ esecuzioni dell'algoritmo $A'_{RG}(\eta)$ sono $O(m/\epsilon)$. \diamond

Poiché il peso $w^*(\Sigma^{OT})$ assume in generale valori razionali non sembra possibile organizzare una ricerca binaria sul suo valore invertendo i ruoli di

costi e pesi in G (come nel Teorema 5.5 per ricavare $A''_{RG}(\epsilon)$ da $A'_{RG}(\epsilon)$). Si può però osservare che se T^{OT} ha l'arco qr parzializzato allora $T - \{qr\}$ è una soluzione *tutto/niente* nella rete $G^{qr} = (V^{qr}, A^{qr})$ che si ottiene da G contraendo l'arco qr e ha perciò peso intero Δ^{OT} ; inoltre $T - \{qr\}$ ha costo minimo tra le soluzioni *tutto/niente* di tale peso (altrimenti sarebbe compromessa l'ottimalità di T^{OT}). Pertanto è interessante considerare il Problema 5.5 (Problema Ausiliario) descritto di seguito (dove $w_{ij}^0 = \Xi_{ij}$ e $w_{ij}^1 = \xi_{ij}$).

Problema 5.5 *Problema Ausiliario*

$$\begin{aligned}
& \text{minimize} && \sum_{ij \in A^{qr}} c_{ij} y_{ij} x_{ij} \\
& \sum_{ij \in A^{qr}} x_{ij} &= & n - 1 \\
& \sum_{ij \in A^{qr}(U)} x_{ij} &\leq & |U| - 1 \quad \forall U \subset V^{qr} \\
& \sum_{ij \in A^{qr}} w_{ij}^{y_{ij}} y_{ij} &\leq & \Delta \\
& x_{ij} &\in & \{0, 1\} \\
& y_{ij} &\in & \{0, 1\}
\end{aligned}$$

Il Problema 5.5 è un problema di Diminuzione Discreta del Minimo Albero Ricoprente con ruoli di pesi e costi invertiti e con *budget* Δ . La sua risoluzione approssimata è alla base dell'Algoritmo 5.3 che, secondo il successivo Teorema 5.11, è uno schema di approssimazione pseudo-polinomiale per il Problema 5.4 (Diminuzione Continua del Minimo Albero Ricoprente).

Algoritmo 5.3 Diminuzione Continua del Minimo Albero Ricoprente II

1. Calcola mediante l'algoritmo $A''_{RG}(\epsilon)$ del Teorema 5.7 una soluzione ammissibile ϵ -approssimata T^A tra quelle del tipo *tutto/niente*.
2. Poni $\Delta := 0$.
3. Calcola (con i seguenti passi) per ogni arco $ij \in A$ un'eventuale soluzione ammissibile ϵ -approssimata $T_{ij}^A(\Delta)$ nell'ipotesi che tale arco ij sia parzializzato e che $w(T - \{ij\}) = \Delta$:

(a) Calcola mediante l'algoritmo $A'_{RG}(\epsilon)$ del Teorema 5.7 una soluzione $T^A[ij, \Delta]$ ϵ -ammissibile e ottimale ($c(T^A[ij, \Delta]) \leq c(T^{OT}[ij, \Delta])$ e $w(T^A[ij, \Delta]) \leq \Delta/(1 - \epsilon)$) per il Problema 5.5 (Problema Ausiliario) con $qr = ij$.

(b) Se $c(T^A[ij, \Delta]) \leq B$ esegui i seguenti passi:

i. Poni $T_{ij}^A(\Delta) := T^A[ij, \Delta] \cup \{ij\}$
con $z_{ij}^A(\Delta) := \min\{c_{ij}, B - c(T^A[ij, \Delta])\}$.

ii. Se $w(T_{ij}^A(\Delta)) < w(T^A)$ poni $T^A := T_{ij}^A(\Delta)$.

4. Se $\Delta < w(T^A)$ poni $\Delta := \Delta + 1$ e ripeti il punto 3.

5. Determina Σ^A a partire da T^A .

Teorema 5.11 L'Algoritmo 5.3 (Diminuzione Continua del Minimo Albero Ricoprente II) fornisce una soluzione ammissibile ϵ -approssimata per il Problema 5.4 (ovvero $c(\Sigma^A) \leq B$ & $w^*(\Sigma^A) \leq w^*(\Sigma^{OT})/(1 - \epsilon)$). La sua complessità computazionale è $O(mW n^{1/\epsilon} \log^2(n) \Psi(n, m))$ ($W = \max_{ij \in A} \Xi_{ij}$).

Dimostrazione.

Tutte le soluzioni generate sono ammissibili per costruzione. Se esiste Σ^{OT} di tipo *tutto/niente* già al punto 1 viene trovata una soluzione ϵ -approssimata (che verrà scartata solo per una di peso minore).

Se invece Σ^{OT} ha un arco qr parzializzato allora $T^{OT} - \{qr\}$ è una soluzione ottima del problema ausiliario per $\Delta = \Delta^{OT}$; perciò è necessariamente $c(T^A[qr, \Delta^{OT}]) \leq c(T^{OT} - \{qr\})$ e quindi

$$w(T_{qr}^A(\Delta^{OT})) \leq \Xi_{qr} - \frac{z_{qr}^{OT}}{\gamma_{ij}} + \frac{\Delta^{OT}}{1 - \epsilon} \leq \frac{w(T^{OT})}{1 - \epsilon}.$$

Il valore Δ^{OT} viene sempre assunto da Δ perché altrimenti esisterebbe una soluzione ammissibile T^A tale che

$$w(T^A) < \Delta^{OT} \leq w(T^{OT})$$

contro l'ottimalità di T^{OT} .

Per quanto riguarda la complessità computazionale l'algoritmo $A'_{RG}(\epsilon)$ viene eseguito $\lceil m w(T^A) \rceil$ volte, con $w(T^A) \leq nW$ e $n^{1+1/\epsilon} \in O(n^{1/\epsilon})$ (il tempo richiesto dall'esecuzione di $A''_{RG}(\epsilon)$ è invece trascurabile). \diamond

Anche in questo caso è possibile modificare l'Algoritmo 5.3 mediante una tecnica di "scalamento" dei dati e ottenere così l'Algoritmo 5.4 le cui prestazioni di schema polinomiale sono descritte dal successivo Teorema 5.12.

Algoritmo 5.4 PTAS-II per la Diminuzione Continua del Minimo Albero Ricoprente

1. Calcola mediante l'algoritmo $A''_{RG}(\epsilon)$ del Teorema 5.7 una soluzione ammissibile ϵ -approssimata T^A tra quelle del tipo tutto/niente.
2. Poni $\eta := \epsilon/(2 - \epsilon)$ e $M := 0$.
3. Poni $t := \max\{0, \lfloor \log(\eta M/2) \rfloor\}$ e $\Delta := 0$.
4. Calcola (con i seguenti passi) per ogni arco $ij \in A$ un'eventuale soluzione ammissibile ϵ -approssimata nell'ipotesi che tale arco ij sia parzializzato e che $w(T - \{ij\}) = \Delta$:
 - (a) Calcola mediante l'algoritmo $A'_{RG}(\eta)$ del Teorema 5.7 una soluzione $T^A[ij, \Delta]$ η -ammissibile e ottimale ($c(T^A[ij, \Delta]) \leq c(T^{OT}[ij, \Delta])$ e $w(T^A[ij, \Delta]) \leq \Delta/(1 - \eta)$) per il Problema 5.5 (Problema Ausiliario) con $qr = ij$.
 - (b) Se $c(T^A[ij, \Delta]) \leq B$ esegui i seguenti passi:
 - i. Poni $T_{ij}^A(\Delta) := T^A[ij, \Delta] \cup \{ij\}$ con $z_{ij}^A(\Delta) := \min\{c_{ij}, B - c(T^A[ij, \Delta])\}$.
 - ii. Se $w(T_{ij}^A(\Delta)) < w(T^A)$ poni $T^A := T_{ij}^A(\Delta)$.
5. Se $\Delta < M$ poni $\Delta := \Delta + 2^t$ e ripeti il punto 4.
6. Se $M < nW$ poni $M := \max\{1, 2M\}$ e ritorna al punto 3.
7. Determina Σ^A a partire da T^A .

Teorema 5.12 L'Algoritmo 5.4 (PTAS-II per la Diminuzione Continua del Minimo Albero Ricoprente) trova una soluzione ammissibile ϵ -approssimata per il Problema 5.4 (ovvero $c(\Sigma^A) \leq B$ & $w^*(\Sigma^A) \leq w^*(\Sigma^{OT})/(1 - \epsilon)$). La sua complessità computazionale è $O(\log(nW)(m/\epsilon) n^{1/\epsilon} \log^2(n) \Psi(n, m))$.

Dimostrazione.

L'algoritmo genera solo soluzioni ammissibili. Se Σ^{OT} è di tipo tutto/niente la soluzione generata al punto 1 è ϵ -approssimata.

Se invece Σ^{OT} ha un arco qr parzializzato si considera il più piccolo valore di M non minore di Δ^{OT} : in corrispondenza di tale valore si può avere $t = 0$ o $t = \lfloor \log(\eta M/2) \rfloor$.

Se $t = 0$ allora $T^{OT} - \{qr\}$ è una soluzione ottima del problema ausiliario per $\Delta = \Delta^{OT}$; perciò $c(T^A[qr, \Delta^{OT}]) \leq c(T^{OT} - \{qr\})$ e di conseguenza

$$w(T_{qr}^A(\Delta^{OT})) \leq \Xi_{qr} - \frac{z_{qr}^{OT}}{\gamma_{ij}} + \frac{\Delta^{OT}}{1 - \eta} \leq \frac{w(T^{OT})}{1 - \epsilon}.$$

Se invece $t = \lfloor \log(\eta M/2) \rfloor$ allora $T^{OT} - \{qr\}$ è una soluzione ammissibile del problema ausiliario con *budget* $\Delta^* = \lceil \Delta^{OT}/2^t \rceil 2^t$ e quindi si ha $c(T^A[qr, \Delta^*]) \leq c(T^{OT} - \{qr\})$. D'altra parte $\Delta^* \leq \Delta^{OT} + 2^t$ e perciò

$$w(T_{qr}^A(\Delta^*)) \leq \Xi_{qr} - \frac{z_{qr}^{OT}}{\gamma_{ij}} + \frac{\Delta^{OT} + 2^t}{1 - \eta} \leq \frac{w(T^{OT}) + \eta M/2}{1 - \eta}$$

da cui si ricava

$$w(T_{qr}^A(\Delta^*)) \leq \frac{w(T^{OT})}{1 - \epsilon}$$

essendo $\Delta^{OT} \leq w(T^{OT})$ e $M \leq 2\Delta^{OT}$ per come $M > 1$ è stato scelto (se $M \leq 1$ necessariamente $t = 0$).

Per quanto riguarda la complessità computazionale il ciclo esterno (M) consta di $O(\log(nW))$ ripetizioni del ciclo interno (Δ). Il ciclo interno a sua volta consta di $O(m/\epsilon)$ esecuzioni dell'algoritmo $A'_{RG}(\eta)$ ($M/2^t < 4/\eta$). L'esecuzione del punto 1 richiede infine un tempo trascurabile. \diamond

5.4 Accrescimento del Minimo Albero Ricoprente

Sia data la rete non orientata $G = (V, A; \xi, \Xi; \gamma)$ nella quale:

- $\xi_{ij} \in N \cup \{0\}$ è il peso originario dell'arco $ij \in A$;
- $\Xi_{ij} \in N \cup \{0\}$ è il peso massimo per l'arco $ij \in A$ ($\Xi_{ij} \geq \xi_{ij}$);
- $\gamma_{ij} \in N$ è il costo unitario per aumentare il peso dell'arco $ij \in A$.

Introducendo le variabili di decisione $y_{ij} \in [0, \beta_{ij}]$ ($\beta_{ij} = \Xi_{ij} - \xi_{ij}$ massimo accrescimento possibile per il peso dell'arco ij) si definisce uno spazio degli stati continuo nel quale $w_{ij} = \xi_{ij} + y_{ij}$. L'obiettivo è trovare fra gli stati che soddisfano il vincolo di bilancio $\sum_{ij \in A} \gamma_{ij} y_{ij} \leq B$ uno stato Σ^{OT} nel quale $w^*(\Sigma^{OT})$ sia massimo.

Problema 5.6 *Accrescimento Continuo del Minimo Albero Ricoprente*

$$\begin{aligned}
 & \text{maximize} && w^*(\Sigma) \\
 & c(\Sigma) &\leq & B \\
 & \Sigma &= & \{y_{ij}\}_{ij \in A} \\
 & y_{ij} &\in & [0, \beta_{ij}] \quad \forall ij \in A
 \end{aligned}$$

Frederickson e Solis-Oba [22] hanno studiato questo problema nel caso $\Xi_{ij} = \infty \quad \forall ij \in A$ arrivando a dare un algoritmo fortemente polinomiale; nel seguito si descrive tale algoritmo adattandolo al caso generale $\Xi_{ij} \leq \infty$.

Per ogni insieme di archi $S \subseteq A$ di una rete G in uno stato Σ chiameremo

$$rank(S, \Sigma) = \min_{T: w(T) = w^*(\Sigma)} |S \cap T|$$

il minimo numero di archi che un minimo albero ricoprente deve condividere con l'insieme S . Chiameremo inoltre $lift[l, S]$ l'operazione di costo unitario $\gamma(S) = \sum_{ij \in S} \gamma_{ij}$ consistente nell'aumentare di l unità il peso di tutti gli archi in S ($y_{ij} := y_{ij} + l \quad \forall ij \in S$). Definita poi $tolerance(S, \Sigma)$ come il minimo valore di l (eventualmente infinito) per cui dopo l'operazione $lift[l, S]$ decresce $rank(S, \Sigma)$ si ha che per $l \leq tolerance(S, \Sigma)$ tale operazione aumenta w^* di $l \cdot rank(S, \Sigma)$ unità a un costo unitario $cost(S, \Sigma) = \gamma(S) / rank(S, \Sigma)$. Per la rete G nello stato Σ si può allora formulare il problema di ottimo che segue.

Problema 5.7 *Sottoinsieme Ottimo*

$$\begin{aligned}
 & \text{minimize} && cost(S, \Sigma) \\
 & S &\subseteq & A
 \end{aligned}$$

Al fine di risolvere il Problema 5.7 (Sottoinsieme Ottimo) osserviamo che per il Teorema 5.1 (Condizioni di Ottimalità sui Cicli) ogni minimo albero ricoprente contiene lo stesso numero di archi aventi un peso assegnato. Allora si può scrivere

$$rank(S, \Sigma) = \sum_{w: \exists ij \in A | w_{ij} = w} rank(S(w), \Sigma)$$

dove $S(w) = \{ij \in S | w_{ij} = w\}$ e quindi

$$cost(S, \Sigma) \geq \min_{w: \exists ij \in A | w_{ij} = w} cost(S(w), \Sigma)$$

cosicché limitarsi a considerare sottoinsiemi di archi aventi tutti lo stesso peso non è riduttivo.

Per una rete $G = (V, A; \gamma)$ definiamo (secondo Cunningham [19])

$$strenght(G) = \min_{S \subseteq A} \frac{\gamma(S)}{\kappa(S, G)}$$

dove $\kappa(S, G)$ è il numero di componenti connesse aggiuntive che si ottengono ponendo $A := A - S$; $strenght(G)$ può essere calcolata in tempo fortemente polinomiale assieme a un insieme S che realizzi tale minimo.

Se G_w è la rete che si ottiene da G eliminando da A gli archi di peso maggiore o uguale a w allora si consideri la rete G_w^* avente un nodo per ogni componente connessa di G_w e un arco per ogni $ij \in A$ tale che $w_{ij} = w$ (tra i nodi corrispondenti alle componenti connesse cui appartengono i e j): per un insieme S di archi aventi tutti lo stesso peso w si ha $rank(S) = \kappa(S, G_w^*)$ e di conseguenza $\min(cost(S)) = strenght(G_w^*)$.

Aggiungere al Problema 5.7 (Sottoinsieme Ottimo) l'ulteriore vincolo che sull'insieme S possa essere eseguita l'operazione $lift[l, S]$ (cioè richiedere $y_{ij} < \beta_{ij} \forall ij \in S$) equivale a considerare alcuni archi di G_w^* indistruttibili. L'Algoritmo 5.5, le cui prestazioni sono caratterizzate dal successivo Teorema 5.13, risolve il Problema 5.7 tenendo conto anche di quest'ultimo aspetto.

Algoritmo 5.5 Sottoinsieme Ottimo

1. Crea una LISTA ordinata dei pesi w_{ij} .
2. Poni $G_{aux} := (V, \emptyset)$ e $s_{min} := \infty$.
3. Ripeti i seguenti passi finché non si verifica la condizione $LISTA = \emptyset$:
 - (a) Estrai w_{ij} da LISTA.
 - (b) Poni $A_{aux} := A_{aux} \cup A(w_{ij})$.
 - (c) Contrai gli archi $ij \in A_{aux}$ per i quali $y_{ij} \geq \beta_{ij}$.
 - (d) Se $strenght(G_{aux}) < s_{min}$ poni $s_{min} := strenght(G_{aux})$.
 - (e) Contrai tutti gli archi $ij \in A_{aux}$.
4. Fornisci in output $S \subseteq A \mid cost(S, \Sigma) = s_{min}$.

Teorema 5.13 *L'Algoritmo 5.5 (Sottoinsieme Ottimo) trova una soluzione ottima per il Problema 5.7 col vincolo che $lift[l, S]$ sia eseguibile. Esso richiede un tempo $O(n^2 m \log(m^2/n))$.*

Dimostrazione.

La correttezza discende dall'essere $G_{aux} = G_{w_{ij}}^*$ quando viene considerato il peso w_{ij} ; il punto 3(c) garantisce che almeno per $l > 0$ sufficientemente piccolo $lift(l, S)$ sia eseguibile. Resta sottinteso che per l'esecuzione del punto 4 l'Algoritmo 5.5 memorizzerà assieme a s_{min} l'insieme S corrispondente.

Per quanto riguarda invece la complessità computazionale risultano asintoticamente dominanti le al più m computazioni di $strenght(G_{aux})$ che possono complessivamente essere eseguite in $O(n^2 m \log(m^2/n))$ (Frederickson e Solis-Oba [22]). \diamond

Il seguente Algoritmo 5.6 risolve il Problema 5.6 (Accrescimento Continuo del Minimo Albero Ricoprente) utilizzando l'Algoritmo 5.5 (Sottoinsieme Ottimo) secondo una strategia di tipo *greedy*: le prestazioni di tale approccio vengono descritte dal successivo Teorema 5.14.

Algoritmo 5.6 Accrescimento Continuo
del Minimo Albero Ricoprente

1. Per ogni arco $ij \in A$ poni $y_{ij} := 0$.
2. Finché è verificata la condizione $B > 0$ esegui i seguenti passi:
 - (a) Utilizza l'Algoritmo 5.7 (Sottoinsieme Ottimo) per trovare $S \subseteq A$ tale che $cost(S, \Sigma)$ sia minimo sotto il vincolo $w_{ij} < \Xi_{ij} \forall ij \in S$.
 - (b) Calcola $l := tolerance(S, \Sigma)$.
 - (c) Per ogni arco $ij \in S$ se $l > \beta_{ij} - y_{ij}$ poni $l := \beta_{ij} - y_{ij}$.
 - (d) Se $l \cdot \gamma(S) > B$ poni $l := B/\gamma(S)$.
 - (e) Esegui $lift(l, S)$.
 - (f) Poni $B := B - l \cdot \gamma(S)$.
3. Fornisci in output $\Sigma = \{y_{ij}\}$.

Teorema 5.14 L'Algoritmo 5.6 (Accrescimento Continuo del Minimo Albero Ricoprente) trova una soluzione ottima per il Problema 5.6 in un tempo $O(n^3 m^2 \log(m^2/n))$.

Dimostrazione.

Conviene immaginare di accrescere i pesi degli archi in modo continuo e quindi indicare con S_b ($0 \leq b \leq B$) l'ultimo insieme su cui è stata effettuata l'operazione $lift(l, S)$ al momento in cui risulta spesa una frazione b del budget B : sia $\Sigma(b) = \{y_{ij}(b)\}$ lo stato di G in tale momento.

Supponiamo per assurdo che esista uno stato ammissibile $\Sigma' = \{y'_{ij}\}$ migliore di quello trovato dall'Algoritmo 5.6 ($w^*(\Sigma') > w^*(\Sigma)$) e definiamo per ogni arco $ij \in A$ la quantità $y'_{ij}(b) := \min\{y'_{ij}, y_{ij}(b) + \Delta_b\}$ dove Δ_b è una costante tale che $\sum_{ij \in A} \gamma_{ij} y'_{ij}(b) = b$. Possiamo allora definire un algoritmo A' che, similmente all'Algoritmo 5.6, effettui una successione di operazioni $lift(l, S'_b)$ con $ij \in S'_b$ se e solo se $y'_{ij}(b - \epsilon) < y'_{ij}$ per $0 < \epsilon \leq b$. In pratica l'algoritmo A' trova la soluzione Σ' cercando di accrescere gli archi come l'Algoritmo 5.6: quando questo non è possibile accresce parimenti tutti gli archi per i quali la variabile di stato non è ancora pari a y'_{ij} .

Definito

$$smeq(qr, \Sigma) = \{ij \in A \mid w_{ij} \leq w_{qr}\}$$

si verifica che

$$smeq(qr, \Sigma(b)) \subseteq smeq(qr, \Sigma'(b))$$

per ogni arco $qr \in S'_b$ ($0 \leq b \leq B$).

Da ciò segue (Frederickson e Solis-Oba [22, Lemma 3.1])

$$rank(S'_b, \Sigma'(b)) \leq rank(S'_b, \Sigma(b))$$

e quindi

$$cost(S_b, \Sigma(b)) \leq cost(S'_b, \Sigma'(b))$$

per la scelta *greedy* di S_b .

Si ha così un assurdo: l'algoritmo A' genera Σ' incrementando w^* a un costo unitario sempre maggiore di quello con il quale l'Algoritmo 5.6 genera Σ .

Per quanto riguarda il calcolo di $tolerance(S, \Sigma)$ al punto 2(b) è sufficiente trovare il minimo peso w_{qr} per il quale

$$\kappa(S, G_{w_{qr}} \cup A(qr)) < \kappa(S, G_{w_{ij}} \cup A(ij)) = rank(S, \Sigma)$$

e questo può essere fatto (Frederickson e Solis-Oba [22]) in tempo trascurabile rispetto al punto 2(a).

La complessità asintotica è allora determinata dal punto 2(a): questo viene ripetuto una volta per poi esaurire il budget B , al più m volte per poi portare un arco ij al suo peso massimo Ξ_{ij} e al più $(m - 1)(n - 1)$ volte (si veda Frederickson e Solis-Oba [22]) per poi eseguire l'operazione $lift(l, S)$ con $l = tolerance(S)$. ◇

| | Diminuzione | Accrescimento |
|-------------------|---|------------------------|
| Modalità Discreta | deb. NP-difficile (teo. 5.6) PTAS ϵ -ammis. (teo. 5.7) PTAS ϵ -appros. (teo. 5.7) | deb. NP-difficile |
| Modalità Continua | deb. NP-difficile (teo. 5.8) PTAS ϵ -ammis. (alg. 5.2) PTAS ϵ -appros. (alg. 5.4) | risol. pol. (alg. 5.6) |

Tabella 5.1: Quadro riassuntivo dei problemi di minimo albero ricoprente.

5.5 Prospettive e Sviluppi

In questo capitolo sono stati studiati alcuni problemi di ottimizzazione su grafi che sorgono quando si voglia intervenire sugli archi di una rete per modificare, sotto un vincolo di bilancio, la lunghezza del suo minimo albero ricoprente. La Tabella 5.1 riassume i risultati ottenuti classificando i problemi in funzione del loro *obiettivo* (diminuzione o accrescimento) e in funzione della *modalità di intervento* che essi prevedono (discreta o continua). Il seguito di questa sezione commenta tale tabella evidenziando sia i risultati originali che i punti dove sarebbe interessante svolgere ulteriore ricerca.

Il Problema 5.3 (Diminuzione Discreta del Minimo Albero Ricoprente) non sembra essere stato oggetto di studio in precedenza, sebbene lo si possa vedere come un problema di Minimo Albero Ricoprente Vincolato definito su un multigrafo e di conseguenza ricondurre (cfr. sez. 5.3) agli studi di Ravi e Goemans [50]. Viceversa il Problema 5.4 (Diminuzione Continua del Minimo Albero Ricoprente) era già stato formulato da Krumke et al. [39] e risolto mediante un algoritmo pienamente polinomiale $(1 - \epsilon)$ -ammisibile e “quasi” ϵ -approssimato (cfr. sez. 5.3). Gli schemi di approssimazione proposti in questo capitolo costituiscono un passo avanti nella risoluzione del Problema 5.4 perché, sebbene polinomiali in modo non pieno, permettono di fissare l’approssimazione desiderata indipendentemente sull’ottimalità o sull’ammisibilità. Resta comunque da chiarire se sia possibile realizzare dei FPTAS (*Fully Polynomial Time Approximation Schemes*) per il Problema 5.4 mantenendo l’indipendenza suddetta tra l’errore di ammissibilità e quello di ottimalità.

Il Problema 5.6 (Accrescimento Continuo del Minimo Albero Ricoprente) è stato formulato e risolto esattamente in tempo polinomiale da Frederickson e Solis-Oba [22] nel caso in cui non vi sia una limitazione superiore al peso che un arco può assumere; nella sezione 5.4 il loro algoritmo è stato leggermente modificato per permettere la risoluzione del Problema 5.6 in forma generale.

Il problema dell'Accrescimento Discreto del Minimo Albero Ricoprente non è invece stato trattato per niente. Risulta abbastanza semplice dimostrare che si tratta di un problema debolmente NP-difficile costruendo una topologia ad albero a partire da un problema di *Bisaccia 0-1* (cfr. Teorema 3.1). In questo modo si può anche dimostrare direttamente la debole NP-difficoltà del Problema 5.3 (Diminuzione Discreta del Minimo Albero Ricoprente). Frederickson e Solis-Oba [22] hanno anche studiato l'accrescimento del minimo albero ricoprente mediante la rimozione di un insieme di archi *essenziali* per la rete (cfr. sez. 3.5).

Varianti interessanti per i problemi studiati in questo capitolo si ottengono considerando misure di efficienza alternative al minimo albero ricoprente. Una prima possibilità è considerare il problema dell'albero ricoprente con minimo collo di bottiglia, essendo in questo modello il peso di un albero pari al peso del suo arco più pesante ($w(T) = \max_{ij \in T} w_{ij}$). Mentre un minimo albero ricoprente ha anche collo di bottiglia minimo non è in generale vero il viceversa (cfr. Ahuja et al. [2, es. 13.32]). Krumke et al. [38] hanno studiato la diminuzione di questa misura di efficienza mediante interventi discreti sui nodi. Un'altra possibilità è considerare il problema del minimo albero di Steiner, essendo un albero di Steiner tale da *ricoprire* un assegnato sottoinsieme di nodi in V . Per un'introduzione al problema di Steiner (che è un problema molto noto oltre che NP-difficile) si rimanda al lavoro monografico di Hwang et al. [33]. Krumke et al. [39] accennano a una risoluzione efficiente per una sua versione vincolata. Un'ulteriore possibilità è fornita dal problema dell'albero ricoprente di minimo diametro, essendo il diametro di un albero la massima distanza tra due suoi nodi: Plesnik [47] ha studiato la complessità computazionale di tale misura, misura considerata anche da Marathe et al. [42].

Capitolo 6

Situazioni di Rischio o Incertezza

Quest'ultimo capitolo fornisce alcune indicazioni per affrontare quei casi nei quali si debba misurare l'efficienza di una rete in condizioni di rischio o comunque di incertezza. Si dice che un decisore opera in condizioni di *incertezza* quando non conosce deterministicamente i dati del problema e in particolare si dice che opera in condizioni di *rischio* quando la conoscenza dei dati del problema è di tipo probabilistico. Coerentemente con l'impostazione seguita finora supporremo che la topologia della rete sia nota con certezza mentre le sue etichette possano assumere valori dipendenti dal *caso*.

Nella prima sezione si introduce il concetto di *utilità*, strumento fondamentale per le decisioni in condizioni di rischio, mentre le tre sezioni successive sono dedicate rispettivamente al cammino minimo, al flusso massimo e al minimo albero ricoprente: l'obiettivo, in tutti questi casi, è trasformare un problema aleatorio in un problema *equivalente deterministico*, così da potere utilizzare tecniche di risoluzione note.

Una volta ricavato un equivalente deterministico per il problema del cammino minimo (o del flusso massimo, o del minimo albero ricoprente) con lunghezze (o capacità, o pesi) di natura stocastica è chiaramente possibile costruire su tale equivalente un problema di diminuzione o accrescimento secondo le modalità viste nei capitoli precedenti. Per la modalità di intervento discreta tutto fila liscio mentre per la modalità di intervento continua i costi unitari (costanti) del modello equivalente possono non corrispondere a un'ipotesi realistica nel modello aleatorio. Pertanto la pianificazione ottima di interventi migliorativi in condizioni di rischio o incertezza è un argomento che presenta ampie possibilità di approfondimento. A questo proposito, poiché i problemi affrontati nei capitoli precedenti sono caratterizzati da un vincolo di bilancio (cfr. sez. 2.2), vale la pena segnalare che Carraway et

al. [13] hanno studiato un modello stocastico di *Bisaccia* con ingombri certi e profitti normalmente distribuiti.

Il riferimento principale per la materia di questo capitolo è un articolo di Loui [41] che tratta diffusamente il problema del cammino minimo con lunghezze stocastiche: si è cercato di estendere gli approcci ivi proposti anche ai problemi di flusso massimo e di minimo albero ricoprente.

6.1 Nozione di Utilità

In pratica spesso e volentieri i valori delle etichette di una rete dipendono dalla sorte e per questo non sono deterministicamente noti: ci si trova allora in una situazione di incertezza che diventa di rischio non appena si considerino le etichette variabili aleatorie conosciute tramite le loro distribuzioni di probabilità.

Per un'introduzione completa al calcolo delle probabilità e alle variabili aleatorie si può vedere per esempio Dall'Aglio [20] mentre per un'introduzione più snella alla probabilità come misura soggettiva dell'ignoranza sulla realtà si suggerisce Scozzafava [51].

Come esempio si considerino aleatori i tempi di percorrenza elencati in Tabella 2.1 (pag. 15): conseguentemente sarà aleatorio il tempo necessario a percorrere un qualsiasi cammino del grafo di Figura 2.1 (pag. 14) e quindi non sarà chiaro in che senso un cammino debba essere considerato più breve di un altro (e quindi preferibile).

Un'idea semplice è di considerare i valori attesi ma questo significa, di fatto, fare entrare la probabilità dalla porta e farla uscire dalla finestra: infatti ciò che caratterizza questa situazione di rischio è proprio che il decisore possa preferire un cammino mediamente non breve se questo dà garanzie di non rivelarsi mai "troppo" lungo. È una situazione molto simile a quella di un investitore che preferisca un guadagno moderato ma certo a uno maggiore ma inaffidabile: il decisore è *avverso al rischio*. In altri casi può al contrario accadere che il decisore sia *propenso al rischio*, come nel caso del giocatore di lotto che considera accettabile un pagamento della vincita minore di quello *equo* poiché esso può comunque cambiargli la vita.

La maggiore o minore propensione al rischio del decisore viene modellata in letteratura (cfr. Gambarelli e Pederzoli [25]) per mezzo di una molto generale *funzione di utilità della moneta* e di un criterio detto della *massima utilità attesa*. Non vi sono difficoltà a considerare il tempo di percorrenza al posto della moneta: è sufficiente tenere presente (cfr. sez. 6.2) che in questo caso i valori piccoli sono preferibili a quelli grandi.

Una funzione di utilità associa a ogni quantità m di moneta un numero $u(m)$ indicativo di quanto essa sia utile per il decisore: in generale avere più moneta non riduce i suoi possibili impieghi e perciò sarà sempre $u(m)$ non decrescente con m ($\frac{du}{dm} \geq 0$). Per quanto riguarda invece la derivata seconda (non si considera limitativo ipotizzare che u la possedga) si possono identificare tre casi molto significativi:

1. $\frac{d^2u}{dm^2} = 0$ per un decisore indifferente al rischio (funzione *affine*);
2. $\frac{d^2u}{dm^2} > 0$ per un decisore propenso al rischio (funzione *convessa*);
3. $\frac{d^2u}{dm^2} < 0$ per un decisore avverso al rischio (funzione *concava*).

Questi tre casi, il primo dei quali corrisponde peraltro all'uso dei valori attesi, possono essere facilmente verificati una volta che si sia affermato e accettato il principio secondo il quale è da preferirsi in ogni caso la scelta o la situazione (per es. il cammino) avente massima utilità attesa.

Sulla base di quanto detto si può quantificare l'avversione al rischio del decisore mediante un coefficiente $r(m) = -\frac{u''(m)}{u'(m)}$ dipendente in generale dalla quantità di moneta m : tanto maggiore il coefficiente tanto maggiore l'avversione al rischio di fronte alla quantità di moneta m . Esempi di funzioni utilità saranno dati nella prossima sezione.

6.2 Problemi di Cammino Minimo

Cominciamo con una situazione (cfr. Ahuja et al. [2, es. 4.39]) in cui, pur comparando valori di probabilità, non si può parlare né di rischio né di incertezza: tale situazione è interessante sia di per sé sia per farvi riferimento nel seguito. Si consideri una rete $G = (V, A; p)$ nella quale $p_{ij} \in [0, 1]$ è la probabilità di “fuori-servizio” dell'arco (i, j) (tale arco può rappresentare un ponte che cede in caso di sisma—Augusti e Ciampoli [8]—come pure un *link* per telecomunicazioni non disponibile nel momento in cui l'elaboratore s vuole comunicare con l'elaboratore t).

Una possibile misura di efficienza per la rete descritta è la sua *affidabilità* $R_{s,t}(G)$ definita come la probabilità che sia “in servizio” almeno un cammino da s a t . Il calcolo di $R_{s,t}(G)$ è però un problema NP-difficile (Valiant [53]) e perciò ripieghiamo sul calcolo di una sua limitazione inferiore: l'affidabilità $R(P_{s,t})$ di un cammino P che vada da s a t , ovvero la probabilità che tale cammino sia “in servizio”. È chiaro che la probabilità che sia “in servizio” almeno un cammino è non minore di quella che sia “in servizio” un cammino assegnato e che la limitazione è tanto più stringente quanto

più $R(P_{s,t})$ è elevata. Misureremo perciò l'efficienza della rete per mezzo della massima affidabilità $R_{s,t}^P(G) = \max_{P_{s,t} \in G} R(P_{s,t})$ di un suo cammino, assumendo così un punto di vista prudente.

Nell'ipotesi che gli archi di G siano stocasticamente indipendenti l'affidabilità di un cammino, cioè la probabilità che tutti i suoi archi siano "in servizio", potrà calcolarsi come $R(P) = \prod_{(i,j) \in P} (1 - p_{ij})$ e sarà quindi possibile calcolare $R_{s,t}^P(G)$ risolvendo un problema di cammino massimo con il prodotto al posto della somma; un tale problema può a sua volta essere formulato come un classico problema di cammino minimo con le etichette non negative $l_{ij} = -\log(1 - p_{ij})$. Poiché le uniche due operazioni effettuate sulle lunghezze nell'ambito di un problema di cammino minimo (per essere concreti si pensi a un algoritmo di tipo Dijkstra) sono il confronto e la somma, si può evitare il calcolo esplicito dei logaritmi—cioè un'operazione teoricamente "infinita"—sfruttando le due note proprietà seguenti:

$$\begin{aligned} \log x \leq \log y &\Leftrightarrow x \leq y \\ \log x + \log y &= \log(xy). \end{aligned}$$

Abbiamo così ricondotto al noto problema del cammino minimo il calcolo dell'affidabilità di una rete (come sopra definita) e di conseguenza possiamo porci il problema di rendere tale rete più o meno affidabile diminuendo o accrescendo la lunghezza di un cammino minimo: se però adottiamo un modello continuo i costi unitari costanti γ_{ij} nel dominio delle lunghezze l_{ij} valgono anche nel dominio delle probabilità p_{ij} solo quando queste tendono a zero (ipotesi di rete molto affidabile).

Passando da una situazione in cui le etichette esprimono la natura aleatoria della rete a quelle situazioni, argomento vero e proprio di questo capitolo, in cui l'aleatorietà è propria delle etichette, presentiamo per primo un approccio (associato da Loui [41] al nome di *Hurwicz*) che presume solo la conoscenza di un valore minimo l_{ij}^m e di un valore massimo l_{ij}^M per ogni lunghezza l_{ij} . Si tratta dunque di un approccio molto generale che si applica a tutte quelle situazioni di incertezza in cui le lunghezze possano considerarsi di valore finito, ovvero nelle quali non vi sia concreta possibilità di rottura per gli archi. D'altro canto esso richiede che si valuti il pessimismo del decisore per mezzo di un parametro $\alpha \in [0, 1]$ avente natura in qualche modo arbitraria.

Semplicemente si usano le lunghezze pesate $l_{ij}^\alpha = (1 - \alpha)l_{ij}^m + \alpha l_{ij}^M$ che tengono conto, tramite α , del maggiore o minore ottimismo da parte del decisore. Nei due casi estremi di massimo ottimismo ($\alpha = 0$) e massimo pessimismo ($\alpha = 1$) si realizzano, rispettivamente, i criteri di *maximax* e *maximin* indicati da Gambarelli e Pederzoli [25, sez. 2.2] per le

decisioni in condizioni di incertezza. Il limite maggiore di questo metodo consiste nella difficoltà, a carattere prettamente empirico, di scegliere un valore “buono” per α ; in alternativa si possono considerare diversi valori di α (es. $\alpha = 0, 0.25, 0.5, 0.75, 1$) e affidare a un decisore umano il confronto tra i—diversi—risultati ottenuti (similmente a quanto si fa in casi semplici di programmazione multi-obiettivo—cfr. Augusti e Ciampoli [8]).

Infine consideriamo una situazione di rischio in cui $P_{ij}(l) = Prob\{l_{ij} \leq l\}$ è la distribuzione di probabilità per la lunghezza dell’arco (i, j) e presentiamo innanzi tutto l’approccio *chance-constrained*, ovvero un’idea fondamentale della programmazione stocastica (si veda per es. Charnes e Cooper [16]).

Nell’approccio *chance-constrained* si interpretano i vincoli in cui compaiono variabili aleatorie come da essere soddisfatti con probabilità non minore di β . Il valore di β (*livello di significatività*) viene scelto arbitrariamente come “accettabile” da un decisore umano ed è in genere prossimo all’unità (es. $\beta = 0.95 = 95\%$). Per quanto riguarda il caso specifico del cammino minimo ci si riferisce, secondo Loui [41], al Problema 3.2 (Massima Tensione) e si fa l’ipotesi che sia $P_{ij}(l_{ij}^\beta) = \beta$ (per i valori di β e le distribuzioni di probabilità più comuni gli l_{ij}^β , detti in inglese *percentiles*, sono tabulati in molti libri di statistica—es. Papoulis [45]).

Problema 6.1 *Cammino Minimo* Chance-Constrained

$$\begin{aligned} & \text{maximize} && d(t) - d(s) \\ d(j) - d(i) & \leq && l_{ij}^{1-\beta} \quad \forall (i, j) \in A \end{aligned}$$

La risoluzione del Problema 6.1 dà un cammino minimo per le lunghezze $l_{ij}^{1-\beta}$: se tale cammino ha lunghezza $l^*(\beta)$ ed è formato da $k \in \{1, \dots, n-1\}$ archi possiamo affermare, nell’ipotesi di archi stocasticamente indipendenti, che esso è lungo al più $l^*(\beta)$ con probabilità almeno β^k . Si palesa così il maggiore difetto di questo metodo: la brevità di un cammino è meno garantita per cammini composti da un numero maggiore di archi o, in altre parole, la soluzione fornita è *polarizzata* verso i cammini con un numero elevato di archi.

Il modo più corretto e generale di affrontare una situazione di rischio è però senz’altro quello di introdurre una funzione di utilità per le lunghezze dei cammini: si tratterà allora di individuare il cammino *più utile* da s a t , ovvero un cammino che abbia massima utilità attesa (cfr. sez. 6.1).

La più semplice funzione di utilità possibile è di tipo *affine*

$$u(l) = -al + b \quad , \quad a > 0$$

dove il segno negativo tiene conto del fatto che le lunghezze più utili sono quelle più piccole. In questo caso per il generico cammino P l'utilità attesa è semplicemente

$$E[u(P)] = -aE[l(P)] + b = -a \sum_{(i,j) \in P} E[l_{ij}] + b$$

e un cammino massimamente utile è semplicemente un cammino a minimo valore atteso ($a = 1$ e $b = 0$ senza perdita di generalità). Di fatto l'aleatorietà dei dati è stata trascurata considerando al posto della variabile aleatoria l_{ij} il suo valore atteso $E[l_{ij}]$ e si è tornati in una situazione deterministica.

Consideriamo ora una funzione di utilità *esponenziale*

$$u(l) = a e^{-bl} \quad , \quad a > 0 \ \& \ b > 0$$

per la quale l'utilità attesa di un cammino P risulta essere

$$E[u(P)] = aE[e^{-bl(P)}] = a \prod_{(i,j) \in P} E[e^{-bl_{ij}}]$$

nell'ipotesi di archi stocasticamente indipendenti. Il cammino più utile è pertanto un cammino massimo con l'operazione prodotto ($a = 1$ senza perdita di generalità) ed è come se volessimo calcolare $R_{s,t}^P(G)$ con $p_{ij} = E[e^{-bl_{ij}}]$ nel problema di affidabilità presentato all'inizio della sezione.

Il motivo per cui le funzioni affini ed esponenziali sono semplici da trattare risiede nel fatto che per esse i sottocammini di un cammino ottimo devono essere ottimi anch'essi. Questo *principio di ottimalità della programmazione dinamica* permette di individuare il cammino più utile con un algoritmo di tipo Dijkstra e perciò efficientemente. Il risultato che segue mostra come non vi siano altri casi semplici: le funzioni di utilità agevoli da trattare corrispondono perciò (cfr. sez. 6.1) a un decisore caratterizzato da un coefficiente di avversione al rischio $r(l) = b$ costante.

Teorema 6.1 *Per funzioni di utilità aventi le prime due derivate e per una rete con archi stocasticamente indipendenti le sole funzioni di utilità che soddisfano il principio di ottimalità della programmazione dinamica sono quelle affini ed esponenziali.*

Per la dimostrazione del Teorema 6.1 si rimanda a Loui [41] dove viene anche affrontato il caso di lunghezze multidimensionali; l'algoritmo proposto per tale caso può risultare, in determinate circostanze, abbastanza efficiente. Tra le applicazioni del caso multidimensionale vi sono quelli monodimensionali in cui la funzione di utilità è polinomiale (es. quadratica) oppure generale per distribuzioni di probabilità chiuse sotto convoluzione e unicamente determinate dai loro primi momenti (es. Poisson, Binomiale, Gaussiana, Gamma).

6.3 Problemi di Flusso Massimo

In questa sezione, sulla falsariga di quanto visto nella sezione 6.2 per il problema del cammino minimo, si svolge il caso in cui le capacità u_{ij} che limitano il flusso su G non siano note con deterministica certezza.

Il cosiddetto *principio di Hurwicz* è senz'altro applicabile anche al caso del flusso massimo: si tratta semplicemente di sostituire ogni capacità u_{ij} con una media pesata $u_{ij}^\alpha = (1 - \alpha)u_{ij}^m + \alpha u_{ij}^M$ del suo minimo e del suo massimo valore possibile; il parametro $\alpha \in [0, 1]$ misura in questo caso l'ottimismo del decisore rispetto alla casualità dei valori di capacità e coerentemente il caso limite $\alpha = 0$ corrisponde a un criterio di *maximin* mentre il caso limite $\alpha = 1$ corrisponde a un criterio di *maximax* (cfr. Gambarelli e Pederzoli [25, sez. 2.2]). Si coglie l'occasione per osservare che la nozione di ottimismo è data in questo contesto di incertezza relativamente ai problemi classici (per cui si spera in un cammino breve e in un flusso grande) ma essa può in qualche modo ribaltarsi a seconda che venga successivamente considerato un problema di accrescimento o un problema di diminuzione.

Anche l'approccio di tipo *chance-constrained* può essere convenientemente applicato al caso del flusso massimo in condizioni di rischio, a partire dalla formulazione del Problema 4.1 (Flusso Massimo).

Problema 6.2 Flusso Massimo Chance-Constrained

$$\begin{aligned} & \text{maximize } \nu \\ & \sum_{j:(v,j) \in A} x_{vj} - \sum_{i:(i,v) \in A} x_{iv} = \begin{cases} +\nu & v = s \\ 0 & v \in V - \{s, t\} \\ -\nu & v = t \end{cases} \\ & 0 \leq x_{ij} \leq u_{ij}^{1-\beta} \end{aligned}$$

Se $P_{ij}(u) = \text{Prob}\{u_{ij} \leq u\}$ è la distribuzione di probabilità per la capacità dell'arco (i, j) si richiede che il flusso su tale arco non sia eccessivo con probabilità almeno β :

$$\text{Prob}\{x_{ij} \leq u_{ij}\} \geq \text{Prob}\{u_{ij} > u_{ij}^{1-\beta}\} = 1 - P_{ij}(u_{ij}^{1-\beta}) = \beta.$$

Ciò garantisce, nell'ipotesi di archi stocasticamente indipendenti, che un flusso di valore $\nu^*(\beta)$ ottimo per il Problema 6.2 sia effettivamente sostenibile da G con probabilità non minore di β^m ($m = |A|$).

Per quanto riguarda invece il criterio della massima utilità attesa esso non sembra adattabile al caso del flusso massimo in quanto non è nemmeno chiaro cosa si debba intendere per valore atteso di un flusso (l'aleatorietà di un flusso non è nel suo valore ma nella sua ammissibilità). Aneja e Nair [6]

hanno comunque fatto un tentativo in tal senso e hanno studiato un caso particolare di flusso a massimo valore atteso. Si osservi che è cosa diversa sostituire alle variabili aleatorie u_{ij} i loro valori attesi $E[u_{ij}]$, cioè di fatto considerare trascurabile il rischio.

6.4 Problemi di Minimo Albero Ricoprente

Si consideri la rete $G = (V, A; p)$ in cui $p_{ij} \in [0, 1]$ è la probabilità che il nodo i comunichi con il nodo j senza che il contenuto della comunicazione diventi di pubblico dominio. Se tale rete è una rete di spie e l'obiettivo è la diffusione di un messaggio segreto su tutta la rete allora una strategia ottima è individuata da un albero ricoprente che sia massimo rispetto al prodotto dei pesi (si fa l'ipotesi che gli archi siano stocasticamente indipendenti).

Questo esempio, che risale addirittura a Prim [48], presenta una forte analogia con quello della sezione 6.2 relativo al calcolo dell'affidabilità di una rete. Se infatti si interpreta p_{ij} come la probabilità che l'arco ij sia "in servizio" e si definisce l'affidabilità $R(T)$ di un albero $T \leq G$ come la probabilità che tutti i suoi archi siano "in servizio" allora il problema sopra esposto corrisponde alla ricerca di un albero avente massima affidabilità $R^T(G)$ (sempre nell'ipotesi di archi stocasticamente indipendenti). Si osservi che definendo l'affidabilità $R(G)$ della rete come la probabilità che essa sia connessa (ovvero che sia "in servizio" almeno un albero) si ottiene una misura di efficienza per G il cui calcolo è NP-difficile (Provan and Ball [49]) e quindi diventa interessante, almeno da un punto di vista prudente, considerare la sua limitazione inferiore $R^T(G)$.

Per affrontare la situazione descritta, che analogamente al caso del cammino minimo è di grande interesse pur non essendo né di rischio né di incertezza, non è neanche necessario passare ai logaritmi per ricondursi al caso di pesi additivi; questo perché gli algoritmi per il calcolo del minimo albero ricoprente (Kruskal, Prim) prevedono solo operazioni di confronto sui pesi (oltre alle necessarie verifiche sulla topologia della rete).

Passando a una situazione di etichette incerte si può senz'altro seguire il *principio di Hurwicz*: a tal fine è sufficiente rimpiazzare ogni peso w_{ij} con la media pesata $w_{ij}^\alpha = (1 - \alpha) w_{ij}^m + \alpha w_{ij}^M$ tra il suo valore minimo e il suo valore massimo; resta il problema di fissare opportunamente il parametro α per modellare il punto di vista del decisore rispetto ai valori estremi w_{ij}^m e w_{ij}^M . Osserviamo per inciso che se costruiamo secondo un modello continuo un problema di diminuzione o accrescimento per l'equivalente deterministico fornito dal *principio di Hurwicz* esso trova corrispondenza nel modello aleato-

rio con la contemporanea variazione continua dei valori minimo e massimo: si tratta tutto sommato di un'ipotesi realistica.

Venendo alle situazioni di rischio, per potere applicare l'approccio *chance-constrained* si deve innanzi tutto formulare il problema del minimo albero ricoprente portando il peso dell'albero (la funzione obiettivo) a formare un vincolo che possa essere interpretato in termini probabilistici. Ciò viene fatto a partire dal Problema 5.1 (Minimo Albero Ricoprente) ottenendo così il seguente Problema 6.3. (Minimo Albero Ricoprente con Pesi Stocastici) nel quale è già esplicitata l'interpretazione probabilistica del nuovo vincolo.

Problema 6.3 *Minimo Albero Ricoprente con Pesi Stocastici*

$$\begin{aligned}
 & \text{minimize} && z \\
 \text{Prob} \left\{ \sum_{ij \in A} w_{ij} x_{ij} \leq z \right\} & \geq && \beta \\
 \sum_{ij \in A} x_{ij} & = && n - 1 \\
 \sum_{ij \in A(U)} x_{ij} & \leq && |U| - 1 \quad \forall U \subset V \\
 x_{ij} & \in && \{0, 1\}
 \end{aligned}$$

Se a questo punto si ipotizza che i pesi w_{ij} siano variabili aleatorie stocasticamente indipendenti e normalmente distribuite ($P_{ij}(w) = N_{\mu_{ij}, \sigma_{ij}}$) diventa possibile (si veda al proposito Ishii et al. [35]) ricavare l'equivalente deterministico che segue ($K_\beta | N_{0,1}(K_\beta) = \beta$).

Problema 6.4 *Minimo Albero Ricoprente Chance-Constrained*

$$\begin{aligned}
 & \text{minimize} && \sum_{ij \in A} \mu_{ij} x_{ij} + K_\beta \sqrt{\sum_{ij \in A} \sigma_{ij}^2 x_{ij}} \\
 \sum_{ij \in A} x_{ij} & = && n - 1 \\
 \sum_{ij \in A(U)} x_{ij} & \leq && |U| - 1 \quad \forall U \subset V \\
 x_{ij} & \in && \{0, 1\}
 \end{aligned}$$

Questo equivalente deterministico non è un problema classico di minimo albero ricoprente e pertanto non è banale costruirci sopra un problema di diminuzione o accrescimento del tipo di quelli visti nei capitoli precedenti. Comunque sia esso è risolubile in tempo polinomiale: Ishii et al. [35] danno un algoritmo che fornisce una soluzione esatta in tempo $O(m^2 \Psi(n, m))$.

Geetha e Nair [28] hanno proposto algoritmi più efficienti per il Problema 6.4 e ne hanno studiato una variante in cui anche la scelta di β fa parte del processo decisionale. Assad e Xu [7] hanno invece studiato il problema del minimo albero ricoprente secondo una funzione di costo quadratica: si tratta di un problema NP-difficile di cui si ottiene un caso particolare formulando il Problema 6.3 nell'ipotesi che i pesi siano normalmente distribuiti ma stocasticamente dipendenti. Infine Ishii e Nishida [34] hanno studiato il problema dell'albero ricoprente con minimo collo di bottiglia nella sua versione *chance-constrained* (cfr. sez. 5.5).

Per quanto riguarda il criterio della massima utilità attesa le funzioni di utilità affini ed esponenziali si gestiscono come nel caso del cammino minimo (cfr. sez. 6.2) mentre per funzioni di utilità più generali il problema è aperto non essendovi in questo caso alcun risultato negativo che corrisponda al Teorema 6.1.

Bibliografia

- [1] V. Aggarwal, Y. Aneja, and K. Nair. Minimal spanning tree subject to a side constraint. *Computers & Operations Research*, 9:287–296, 1982.
- [2] R. Ahuja, T. Magnati, and J. Orlin. *Network Flows*. Prentice-Hall, 1993.
- [3] R. Ahuja, K. Mehlhorn, J. Orlin, and R. Tarjan. Faster algorithms for the shortest path problem. *Journal of ACM*, 37:213–223, 1990.
- [4] R. Ahuja, J. Orlin, and R. Tarjan. Improved time bounds for the maximum flow problem. *SIAM Journal on Computing*, 18:939–954, 1989.
- [5] N. Alon. Generating pseudo-random permutations and maximum flow algorithms. *Information Processing Letters*, 35:201–204, 1990.
- [6] Y. Aneja and K. Nair. Maximal expected flow in a network subject to arc failures. *Networks*, 10:45–57, 1980.
- [7] A. Assad and W. Xu. The quadratic minimum spanning tree problem. *Naval Research Logistics*, 39:399–417, 1992.
- [8] G. Augusti and M. Ciampoli. On the seismic risk of highway networks and its reduction. In *RISK ANALYSIS: Proceedings of a Symposium at University of Michigan (Ann Arbor, USA)*, pages 25–36, Agosto 1994.
- [9] G. Ausiello, A. Marchetti Spaccamela, and M. Protasi. *Teoria e Progetto di Algoritmi Fondamentali*. FrancoAngeli, 1994.
- [10] M. Ball, B. Golden, and R. Vohra. Finding the most vital arcs in a network. *Operations Research Letters*, 8:73–76, 1989.
- [11] C. Batini, L. Carlucci Aiello, M. Lenzerini, A. Marchetti Spaccamela, and A. Miola. *Fondamenti di Programmazione dei Calcolatori Elettronici*. FrancoAngeli, 1992.

- [12] O. Berman. Improving the location of minisum facilities through network modification. *Annals of Operations Research*, 40:1–16, 1992.
- [13] R. Carraway, R. Schmidt, and L. Weatherford. An algorithm for maximizing target achievement in the stochastic knapsack problem with normal returns. *Naval Research Logistics*, 40:161–173, 1993.
- [14] I. Cattaneo Gasparini. *Complementi di Geometria e Algebra—parte II*. Masson-ed. Veschi, 1993.
- [15] R. Chandrasekaran. Minimal ratio spanning trees. *Networks*, 7:335–342, 1977.
- [16] A. Charnes and W. Cooper. Deterministic equivalents for optimizing and satisficing under chance constraints. *Operations Research*, 11:18–39, 1963.
- [17] J. Cheriyan, T. Hagerup, and K. Mehlhorn. Can a maximum flow be computed in $O(mn)$ time? In *Proceedings of the 17th International Colloquium on Automata, Languages and Programming*, pages 235–248, 1990.
- [18] S. Cook. The complexity of theorem proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.
- [19] W. Cunningham. Optimal attack and reinforcement of a network. *Journal of ACM*, 32:549–561, 1985.
- [20] G. Dall’Aglio. *Calcolo delle Probabilità*. Zanichelli, 1987.
- [21] E. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.
- [22] G. Frederickson and R. Solis-Oba. Increasing the weight of minimum spanning trees. In *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 539–546, 1996.
- [23] D. Fulkerson and G. Harding. Maximizing the minimum source-sink path subject to a budget constraint. *Mathematical Programming*, 13:116–118, 1977.
- [24] H. Gabow, Z. Galil, T. Spencer, and R. Tarjan. Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. *Combinatorica*, 6:109–122, 1986.

- [25] G. Gambarelli and G. Pederzoli. *Metodi di Decisione*. Hoepli, 1992.
- [26] G. Gambolati. *Elementi di Calcolo Numerico*. Edizioni Libreria Cortina, 1992.
- [27] M. Garey and D. Johnson. *Computers and Intractability: a Guide to the Theory of NP-completeness*. Freeman, 1979.
- [28] S. Geetha and K. Nair. On stochastic spanning tree problem. *Networks*, 23:675–679, 1993.
- [29] B. Golden. A problem in network interdiction. *Naval Research Logistics*, 25:711–713, 1978.
- [30] M. Grötschel, L. Lovász, and A. Shrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag, 1993.
- [31] R. Hassin. Approximation schemes for the restricted shortest path problem. *Mathematics of Operations Research*, 17:36–42, 1992.
- [32] J. Hopcroft and R. Tarjan. Efficient planarity testing. *Journal of ACM*, 21:549–568, 1974.
- [33] F. Hwang, D. Richards, and P. Winter. *The Steiner Tree Problem*. Number 53 in Annals of Discrete Mathematics. North-Holland, 1992.
- [34] H. Ishii and T. Nishida. Stochastic bottleneck spanning tree problem. *Networks*, 13:443–449, 1983.
- [35] H. Ishii, S. Shiode, and T. Nishida. Stochastic spanning tree problem. *Discrete Applied Mathematics*, 3:263–273, 1981.
- [36] D. Johnson. A priority queue in which initialization and queue operations take $O(\log \log d)$ time. *Mathematical Systems Theory*, 15:295–309, 1982.
- [37] V. Klee. String algorithm for shortest path in directed networks. *Operations Research*, 12:428–432, 1964.
- [38] S. Krumke, M. Marathe, H. Noltemeier, R. Ravi, S. Ravi, R. Sundaram, and H. Wirth. Improving spanning trees by upgrading nodes. In *Proceedings of 24th International Colloquium on Automata, Languages and Programming*, 1997. In corso di pubblicazione.

- [39] S. Krumke, H. Noltemeier, M. Marathe, S. Ravi, and K. Drangmeister. Modifying networks to obtain low cost trees. In *Proceedings of Workshop on Graph Theoretic Concepts in Computer Science (WG'96)*, pages 293–307, 1996.
- [40] J. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7:48–50, 1956.
- [41] L. Loui. Optimal paths in graphs with stochastic or multidimensional weights. *Communications of the ACM*, 26:670–676, 1983.
- [42] M. Marathe, R. Ravi, R. Sundaram, S. Ravi, D. Rosenkrantz, and H. Hunt III. Bicriteria network design problems. In *Proceedings of the 22nd International Colloquium on Automata, Languages and Programming*, pages 487–498, 1995.
- [43] D. Paik and S. Sahni. Network upgrading problems. *Networks*, 26:45–58, 1995.
- [44] C. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, 1982.
- [45] A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, 1991.
- [46] C. Phillips. The network inhibition problem. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, pages 776–785, 1993.
- [47] J. Plesnik. The complexity of designing a network with minimum diameter. *Networks*, 11:77–85, 1981.
- [48] R. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36:1389–1401, 1957.
- [49] J. Provan and M. Ball. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM Journal on Computing*, 12:777–788, 1983.
- [50] R. Ravi and R. Goemans. The constrained minimum spanning tree problem. In *Proceedings of Scandinavian Workshop on Algorithmic Theory (SWAT'96)*, pages 66–75, 1996.

- [51] R. Scozzafava. *La Probabilità Soggettiva e le Sue Applicazioni*. Masson-ed.Veschi, 1993.
- [52] R. Tarjan. Efficiency of a good but non linear set union algorithm. *Journal of ACM*, 22:215–225, 1975.
- [53] L. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8:410–421, 1979.
- [54] J. Welsh and J. Elder. *Introduzione al Pascal*. ed.ESA, 1991.
- [55] R. Wilkow. Analysis and design of reliable computer networks. *IEEE Transactions on Communications*, 20:660–678, 1972.