

Università di Roma “La Sapienza”
Sede di Latina (Università Pontina)
Corso di Laurea in Ingegneria Informatica

Appunti dalle lezioni di
RICERCA OPERATIVA

Anno Accademico 2003-2004

Indice

Introduzione	5
Che cosa è la Ricerca Operativa	5
Breve storia della Ricerca Operativa	5
La Ricerca Operativa oggi	6
1 I Modelli della Ricerca Operativa	10
1.1 L'approccio modellistico	10
1.2 Modelli di Ottimizzazione	13
1.3 Esempi di costruzione di modelli di Programmazione Matematica	16
2 La Programmazione Lineare	21
2.1 Definizione del problema	21
2.2 Semplici esempi di problemi di programmazione lineare	23
2.2.1 Un problema di produzione (allocazione di risorse)	23
2.2.2 Un problema di miscelazione	24
2.2.3 Un problema di trasporto	24
2.3 Interpretazione grafica di un Problema di PL	25
2.4 Teorema fondamentale della Programmazione Lineare	33
2.5 Cenni sul Metodo del Simplex	38
2.6 Modellazione dei Problemi di Programmazione Lineare	40
2.7 Esempi tipici di PL	42
2.7.1 Modelli di allocazione ottima di risorse	42
2.7.2 Modelli di miscelazione	52
2.7.3 Modelli di trasporto	59
3 La dualità nella Programmazione Lineare	67
3.1 Teoria della dualità	67
3.1.1 Risultati fondamentali della teoria della dualità	71
3.1.2 Condizioni di complementarità	75
3.2 Interpretazione della Dualità	79
3.2.1 Interpretazione economica della dualità e prezzi ombra	79
3.2.2 Il duale del problema di allocazione ottima di risorse	83
3.2.3 Il duale del problema di miscelazione	84
3.2.4 Il duale del problema dei trasporti	86

4 Problemi di Programmazione Lineare Intera	88
4.1 Formulazioni Classiche di Problemi Lineari Interi	88
4.1.1 Knapsack binario	88
4.1.2 Assegnamento	89
4.1.3 Problema del costo fisso.	89
4.1.4 Capital Budgeting	91
4.1.5 Localizzazione	94
4.1.6 Scheduling (Sequenziamento)	97
4.2 Tecniche di soluzione per problemi di PLI	98
4.2.1 Soluzione per enumerazione totale	99
4.2.2 Soluzione approssimata per arrotondamento	99
4.2.3 La Tecnica del Branch and Bound	100
4.3 Esempi	106
5 Grafi: nozioni fondamentali	112
5.1 Definizioni fondamentali	112
5.2 Rappresentazioni di un grafo	117
5.3 Alcuni esempi	119
6 Cammini minimi	125
6.1 Il problema del cammino minimo e alcuni esempi di applicazioni	125
6.1.1 Percorso di tempo minimo su una rete stradale	126
6.1.2 Costruzione di una autostrada	126
6.2 Cammini minimi e massimi su grafi aciclici	127
6.2.1 Numerazione topologica dei nodi di un grafo	127
6.2.2 Un algoritmo per il cammino minimo su grafi aciclici	129
6.2.3 Un algoritmo per il cammino massimo su grafi aciclici	132
6.3 Cammini minimi su grafi con pesi positivi: algoritmo di Dijkstra	133
6.4 Due esempi	137
6.4.1 Tecniche reticolari di programmazione delle attività	137
6.4.2 Gestione delle scorte.	145
7 Massimo flusso	151
7.1 Il problema del massimo flusso.	151
7.2 Alcuni risultati preliminari	153
7.3 Cammini aumentanti	155
7.4 Condizioni di ottimalità	158
7.5 L'algoritmo di Ford e Fulkerson.	161
7.6 Calcolo e scelta dei cammini aumentanti	162
7.7 Esempi	165
7.8 Accoppiamento bipartito	168
7.9 Il problema di distribuzione di flusso a costo minimo	170

8 Euristiche per la soluzione di problemi di ottimizzazione combinatoria	174
8.1 L'ottimizzazione combinatoria	175
8.1.1 Il problema dell'accoppiamento massimo su grafi bipartiti	176
8.1.2 Il Problema del Commesso Viaggiatore.	176
8.1.3 Il problema del partizionamento di un grafo.	177
8.1.4 PL01	178
8.2 Un sistema multitaxi per il servizio Aeroporto Fiumicino - Roma Centro	179
8.3 Euristiche di tipo "Greedy"	182
8.3.1 Applicazione dell'algoritmo greedy generico al problema del Commesso Viaggiatore.	183
8.3.2 Applicazione dell'algoritmo greedy generico al problema del Partizionamento di Grafi (clustering).	186
8.3.3 Una diversa forma del generico algoritmo greedy.	190
8.4 Ricerca Locale	192
8.4.1 Algoritmo generico di Ricerca Locale	197
8.5 Estensione del modello al caso Roma Centro - Fiumicino Aeroporto	198

Nota Queste dispense sono riprese integralmente da quelle dei corsi di Ricerca Operativa tenuti presso la sede di Roma, in particolare da quelle di Francisco Facchini e Carlo Mannino per i Cap. 1,2,4,5,6,7,8, e da quelle di Stefano Lucidi e Massimo Roma per il Cap. 3.

Introduzione

La Ricerca Operativa (di seguito indicata anche con l'acronimo *RO*) si occupa dello sviluppo e dell'applicazione di metodi quantitativi per la soluzione di problemi di decisione che si presentano nella gestione di imprese e organizzazioni.

La Ricerca Operativa è una disciplina relativamente giovane. Il termine *Ricerca Operativa* è stato coniato in ambito militare verso la fine degli anni '30 e deriva dal termine inglese "*Operational Research*", ovvero la "ricerca sulle operazioni (militari)".

Quando la complessità dei sistemi era relativamente contenuta, e la quantità di dati disponibili estremamente limitata, il personale esperto era sufficiente a prendere le decisioni necessarie alla conduzione dell'impresa. La crescente complessità dei sistemi aziendali e l'integrazione internazionale delle imprese, congiuntamente all'enorme quantità di dati messa a disposizione dall'informatizzazione capillare, ha reso indispensabile l'utilizzo di strumenti automatici di decisione che attraverso la modellazione matematica permettano la soluzione di problemi di grandi dimensioni.

La RO, quindi, è caratterizzata dall'uso di modelli matematici definiti e risolti al fine di fornire indicazioni ai "decisori" nell'atto della scelta. Non a caso, la RO è anche nota come *management science*, e cioè la Scienza della Gestione, definizione che ne sintetizza finalità e ambizioni.

Breve storia della Ricerca Operativa

Il termine Ricerca Operativa, si è detto, è legato alle prime applicazioni della RO per aumentare l'efficienza di operazioni militari della Seconda Guerra Mondiale. Tuttavia esistono esempi importanti di anticipazione dei metodi della RO in anni più lontani; il più famoso risale a F. Taylor che nel 1885 elaborò uno studio sui metodi di produzione; prima ancora, nel 1776, G. Monge aveva studiato un problema di trasporti. Tuttavia la nascita della RO è legata agli studi che negli anni immediatamente precedenti alla Seconda Guerra Mondiale vennero condotti in Gran Bretagna per risolvere problemi strategici e tattici in operazioni militari. Più in particolare questi studi erano legati all'uso efficiente di un nuovo strumento di difesa: il radar. Infatti nel 1937 la Royal Air Force iniziò degli esperimenti di un sistema di controllo della difesa aerea basato sull'uso di una stazione radar situata a Bawdsey Research Station, nella costa est; già dai primi esperimenti si resero conto che era molto difficile gestire efficientemente le informazioni provenienti dal radar. Nel luglio 1938 furono compiuti altri esperimenti con l'aggiunta di quattro stazioni radar lungo la costa nella speranza che il sistema di controllo migliorasse sia in copertura sia in efficienza; invece non fu così; dai nuovi esperimenti emersero seri problemi: c'era la necessità di coordinare e correlare le tante informazioni, spesso anche in conflitto tra di loro, che venivano ricevute dalle stazioni radar aggiunte. Nell'imminenza della Guerra si rese necessario tentare qualche nuovo approccio; perciò il sovrintendente della Bawdsey Research Station propose di sviluppare un programma di ricerca che riguardasse gli aspetti *operativi* del sistema e non più solamente quelli prettamente tecnici che erano da considerare soddisfacenti. Il termine "*Operational Research*" – Ricerca nelle operazioni (militari) – fu coniato per descrivere questa nuova branca delle scienze applicate. Fu quindi selezionato un gruppo di scienziati di vari discipline per costituire un "OR team"; il progetto fu diretto dal comandante in capo della Royal Air Force, Air Chief Marshal Sir Hugh Dowding. Nell'estate del 1939 la Gran Bretagna

effettuò l'ultima esercitazione pre-bellica dove si evidenziò un notevole miglioramento nelle operazioni di difesa aerea grazie al contributo del gruppo di scienziati. Nacque quindi una vera e propria sezione che più tardi, nel 1941, prese il nome formale di "Operational Research Section". Durante il conflitto mondiale ci furono importanti contributi strategici di questa sezione che permisero di salvare piloti e aerei impegnati nel conflitto. Nonostante gli scopi bellici, anche se di difesa, del progetto, per la prima volta in questa occasione si ebbe una convergenza di scienziati di diverse discipline con l'obiettivo di determinare la più efficiente utilizzazione di risorse limitate usando tecniche quantitative.

Al termine della guerra, alcuni degli scienziati coinvolti nel progetto formarono nuclei di ricercatori per lo sviluppo post bellico e la loro attività si estese a campi diversi da quello militare; in particolare, con l'espandersi delle iniziative industriali e con l'avvento dei computer che sono uno strumento essenziale per la risoluzione dei problemi, c'è stata un'espansione dell'utilizzo della RO all'interno di diverse realtà applicative.

Dagli anni '60 in poi le applicazioni della RO hanno avuto diffusione crescente, inizialmente nell'ambito di grandi gruppi industriali e successivamente, grazie anche alla disponibilità di grandi potenze di calcolo a costi contenuti, in quasi ogni settore industriale, nei servizi e nelle amministrazioni pubbliche.

La Ricerca Operativa oggi

Ai nostri giorni la rilevanza applicativa delle tecniche della RO è riconosciuta e apprezzata in ambito industriale. Negli ultimi cinque anni il numero di addetti del settore è infatti cresciuto di un fattore 100. Contestualmente, si è allargata la richiesta di esperti di RO nelle imprese manifatturiere e di servizi: un laureato, esperto di tecniche della RO può ragionevolmente aspirare, per esempio, a ricoprire incarichi di responsabilità nelle industrie manifatturiere, nella assicurazioni, nel marketing, nelle società di consulenza aziendale, nella pianificazione e, sempre di più, nelle telecomunicazioni.

Alcuni esempi di problemi possono essere risolti per mezzo delle tecniche della RO sono i seguenti:

- *Finanza e Investimenti;*

si vuole rispondere a domande del tipo: quanto dobbiamo investire, e come? Dove rimediare i capitali necessari? Quanto ci costerà? Alcuni esempi sono:

- *Selezione degli investimenti;*

si tratta di scegliere, fra un vasto insieme di alternative di investimento, quali attivare e quali no in presenza di vincoli di budget e con l'obiettivo di massimizzare i ricavi.

- *Scelta del portafoglio;*

consiste nel decidere in quali titoli e con quali quote investire i nostri capitali in modo da massimizzare il ricavo atteso, oppure minimizzare il rischio, etc.

- *Determinazione del prezzo di derivati finanziari;*

si vuole determinare il prezzo di un prodotto derivato finanziario (per esempio di un'opzione o di un future) in funzione del tempo e dell'andamento del titolo sottostante.

- *pianificazione della produzione;*

come assegnare la forza lavoro alle varie attività della nostra impresa? Su quali macchine e per quanto tempo ci conviene effettuare i nostri processi?

Si tratta di pianificare i livelli di produzione e/o l'utilizzazione di risorse; si hanno spesso problemi di *allocazione ottima di risorse* cioè problemi riguardanti la distribuzione di risorse limitate tra alternative concorrenti in modo da minimizzare il costo complessivo o massimizzare il guadagno totale; tali risorse possono essere materie prime, manodopera, tempi di lavoro su macchine, capitali investiti.

- *gestione ottima delle scorte;*
si tratta di determinare i livelli di produzione e di scorte nella gestione di materiali grezzi, prodotti in lavorazione etc.; quando e quanto conviene riordinare materiali o beni in modo da ottenere il miglior compromesso fra costi di riordino e di produzione/acquisto e costi di immagazzinamento. Convieni, cioè, ordinare o produrre più spesso minori quantità per far fronte alla domanda corrente, oppure ordinare/produrre maggiori quantità e lasciarle in magazzino per soddisfare anche la domanda futura?
- *localizzazione e dimensionamento di impianti;*
dove ci conviene costruire le stazioni di base di una rete GSM/UMTS per coprire efficacemente il territorio e il traffico, e con che potenza dovranno trasmettere? Quanti depositi di un'impresa di distribuzione alimentare costruire e dove localizzarli per servire i negozi a dettaglio in un'area d'interesse? Dove costruire degli ospedali (o scuole o stazioni dei vigili del fuoco) in modo da ottimizzare il servizio fornito?
In senso lato, si tratta di problemi in cui si deve decidere dove installare “impianti di produzione” in modo da “rifornire” in modo ottimale aree distribuite su un territorio.
- *progetto di reti di telecomunicazione;*
si tratta di definire i collegamenti e dimensionare le capacità di una rete di telecomunicazione, di trasmissione dati, etc., in modo da garantire il traffico tra le varie origini e destinazioni e minimizzare il costo complessivo; ad esempio, per instradare le comunicazioni fra Roma e Venezia, conviene costruire una nuova linea ad alta velocità in fibra ottica fra Firenze e Bologna oppure installare un ponte radio a larga banda?
- *assegnazione di frequenze di trasmissione;*
quali frequenze (prese da una banda limitata) devo assegnare a una rete di trasmettitori radio-televisivi in modo da minimizzare le interferenze reciproche o massimizzare la copertura del territorio?
- *sequenziamento;*
quali processo o operazione effettuare prima e quali dopo? Per esempio, come sequenziare i treni sulla rete in modo da evitare conflitti sulle tratte e minimizzare i tempi morti, le attese alle stazioni, etc.?
- *project planning;*
come sequenziare le molteplici attività di un progetto? Quanto durerà il progetto? Come devono essere gestite le risorse?
- *allocazione ottima di componenti elettronici (VLSI design);*
come disegnare una piastra madre in modo da minimizzare le lunghezze dei percorsi seguiti dai segnali elettrici?
- *determinazione dei turni del personale;*
si tratta, ad esempio, di assegnare ai convogli il personale viaggiante sui treni (conducenti, bigliettai, etc.) in modo da minimizzare il numero di viaggi ”a vuoto” (necessari per riportare il personale alla loro sede). Un problema analogo si presenta nell'assegnazione di equipaggi (piloti, hostess, steward) a voli.
- *manutenzione di beni;*
cioè il problema di decidere quando e se effettuare la manutenzione di alcuni beni soggetti ad usura, in modo da minimizzare il costo complessivo.
- *istradamento di veicoli;*
quali percorsi devono seguire i veicoli di una flotta di automezzi per, ad esempio, raccogliere l'immondizia, o rifornire una rete di negozi, in modo da minimizzare le distanze complessive percorse?

- *studi sulla struttura del DNA;*
come assegnare sequenze a geni minimizzando la probabilità dell'errore sperimentale? Come determinare un albero filogenetico massimizzando la verosimiglianza?
- *progettazione di forme ottime;*
che forma deve avere una macchina in modo da presentare meno resistenza possibile all'aria? Che profilo deve avere l'ala di un aereo in modo da massimizzare la portanza?
- *calcolo delle traiettorie ottime;*
qual è la traiettoria che permette ad un veicolo spaziale di arrivare sulla luna e tornare usando la quantità minima di carburante?
- *ricostruzione di immagini;*
come si possono visualizzare le informazioni fornite, per esempio, da una TAC in modo da renderle più leggibili possibili per il medico?
- *progettazione strutturale ;*
qual è il progetto di un ponte o di un grattacielo che resiste meglio a venti molto forti o alle sollecitazioni derivanti da un terremoto?

Questa lista, lungi dall'essere esaustiva, serve a mettere in evidenza le potenzialità degli strumenti della RO nella risoluzione di problemi applicativi complessi e disparati.

In Italia la penetrazione della RO è stata piuttosto lenta nonostante l'inadeguatezza delle tecniche utilizzate in rapporto a problemi di complessità crescenti. La situazione è rovesciata negli Stati Uniti e nell'Europa Centro-Settentrionale ove la crescita del settore è stata formidabile. Le ragioni del ritardo sono in primo luogo culturali: mancanza di conoscenze approfondite da parte delle aziende, insufficiente disseminazione dei risultati da parte dell'accademia. Lentamente, questa situazione va modificandosi anche in Italia, e la sensibilità delle aziende è fortemente cresciuta negli ultimi due-tre anni. In particolare ci si è resi conto che l'informatizzazione capillare e l'accresciuta potenza di calcolo non sono sufficienti a risolvere i problemi dell'organizzazione aziendale in modo ottimale.

A confermare questo asserto si consideri il seguente, illuminante esempio (dovuto a G. B. Dantzig): si supponga di essere a capo di un'azienda che impiega 70 dipendenti e deve assegnare ciascuno di essi a 70 differenti mansioni; poiché le capacità lavorative di ogni singolo dipendente sono diverse, non è indifferente per l'azienda come effettuare l'assegnamento. Naturalmente si deve fare in modo che ciascun dipendente sia assegnato ad una sola mansione e che ciascuna mansione sia svolta esattamente da un dipendente. Il problema consiste nel confrontare le $70!$ possibilità che ci sono per selezionare quella migliore nel senso che permetta di ottenere il maggiore utile per l'azienda. Le possibilità sono un numero molto grande, più grande di 10^{100} . Ora si supponga di disporre di un calcolatore capace di effettuare un milione di calcoli al secondo e che sia in funzione dal tempo del big bang, 15 milioni di anni fà; avrebbe questo calcolatore oggi nell'anno 2000 esaminato tutte le $70!$ combinazioni possibili ? La risposta è no. Supponiamo allora di disporre di un calcolatore che possa effettuare un bilione di assegnamenti per ogni nano secondo; la risposta sarebbe ancora no. Supponiamo allora di riempire la superficie terrestre di calcolatori di questo tipo che lavorano in parallelo; la risposta sarebbe ancora no. Si dovrebbe disporre in verità di 10^{40} terre ciascuna ricoperta di calcolatori di questo tipo, in funzione dal tempo del big bang fino a quando il sole si raffredderà.

Da questo esempio facile da enunciare si deduce come in certe situazioni sia assolutamente impossibile esaminare tutti i casi possibili per determinare qual è il migliore. Per questo, prima dell'avvento della RO, l'unica possibilità era affidarsi al buon senso di persone guidate dall'esperienza che stabilivano regole "ad hoc" di base che dovevano essere seguite per risolvere i problemi ("*ad hoc*" *ground-rule approach*).

A questo tipo di approccio si contrappone la RO, il cui contributo centrale consiste nell'introduzione del cosiddetto *approccio modellistico-ottimizzatorio* per la soluzione di un problema di decisione. In questo approccio si organizza l'analisi di un problema reale in due fasi:

- la rappresentazione del problema attraverso un *modello matematico* che ne astragga gli aspetti essenziali e che schematizzi le interrelazioni esistenti tra i diversi aspetti del fenomeno che si sta studiando;
- lo sviluppo di *metodi matematici efficienti* (algoritmi di soluzione) per determinare una soluzione ottima del problema o una sua buona approssimazione.

Naturalmente, per costruire correttamente un modello matematico-ottimizzatorio che rappresenti un particolare fenomeno, si devono individuare i parametri di controllo significativi e un criterio per la valutazione della qualità della soluzione. La determinazione del modello è un'attività complessa e non completamente formalizzabile, che deve far ricorso da una parte a una conoscenza approfondita delle caratteristiche del problema in esame e dall'altra a strumenti che provengono da diverse branche della matematica. Una volta determinato il modello corretto, la RO si occupa di fornire una procedura esplicita per determinare una soluzione di un problema; tale procedura può essere rappresentata da metodi matematici analitici o, come più spesso accade, da metodi numerici che determinano la soluzione del problema mediante specifici algoritmi di calcolo. Da quanto detto si può capire come la RO sia una metodologia tipicamente interdisciplinare, applicabile nei più svariati contesti e come proprio dagli stimoli provenienti da campi anche molto distanti tra di loro traggia una delle principali ragioni della sua attuale vitalità.

Capitolo 1

I Modelli della Ricerca Operativa

1.1 L'approccio modellistico

Il termine *modello* è di solito usato per indicare una struttura appositamente costruita per mettere in evidenza le caratteristiche principali di alcuni oggetti reali. Alcune volte possono essere concreti (come ad esempio i modelli rappresentanti prototipi di aerei o auto), ma più spesso, come nella Ricerca Operativa, si tratta di *modelli astratti* cioè *modelli matematici* che usano il simbolismo dell'algebra per mettere in evidenza le relazioni principali dell'oggetto che deve essere modellato. I modelli di cui si tratterà in seguito sono quindi modelli matematici, e sono costituiti da un insieme di relazioni che descrivono in modo semplificato, ma sempre rigoroso, uno o più fenomeni del mondo reale. La nozione di modello matematico per rappresentare il mondo reale non è certo nuova: già Pitagora nel IV secolo a.C. tentava di costruire un modello matematico dell'Universo anche se sotto una luce più esoterica che scientifica. L'interesse per la modellistica matematica è notevolmente cresciuto negli anni più recenti e ai giorni nostri è sempre più viva la convinzione che ricorrendo a modelli matematici sia possibile analizzare i molteplici aspetti del mondo reale e studiare l'influenza che l'uomo può esercitare su di essi. Ciò ha portato ad un enorme sviluppo delle applicazioni della modellistica matematica anche al di fuori delle tradizionali applicazioni alle scienze fisiche. Si è così avuta di fatto una vasta utilizzazione di modelli matematici in settori lontani dagli ambiti più tradizionali come, ad esempio, le scienze sociali, la biologia, le scienze ambientali, la psicologia. Come esempi concreti, si pensi agli studi sulla dinamica della popolazione, sulla diffusione delle epidemie, sul risanamento ambientale. Questa notevole diffusione della modellistica matematica è anche dovuta al fatto che l'evoluzione di un modello matematico può essere rapidamente studiata grazie all'uso di moderni calcolatori elettronici.

È evidente come in molti casi le situazioni rappresentate da un modello sono molto complesse e alcune volte influenzate da fenomeni di natura aleatoria; per questa ragione, sono state definite diverse classi di modelli matematici: *modelli stocastici* che considerano grandezze che possono essere influenzate da fenomeni aleatori e *modelli deterministici* che considerano grandezze esatte; inoltre a seconda che le interazioni tra le grandezze sono immediate o distribuite nel tempo, si parla di *modelli statici* e di *modelli dinamici*.

L'approccio modellistico per risolvere un problema di decisione o, più in generale, l'impiego di metodi matematici per la soluzione di problemi applicativi, viene di solito realizzato attraverso diverse fasi. Tali fasi possono essere schematizzate nel seguente modo:

- **Analisi del problema**
- **Costruzione del modello**
- **Analisi del modello**

- **Soluzione numerica**
- **Validazione del modello**

La prima fase consiste nell'*analisi della struttura del problema* per individuare i legami logico-funzionali e gli obiettivi.

Nella successiva fase di *costruzione del modello*, chiamata anche *formulazione*, si descrivono in termini matematici le caratteristiche principali del problema; questa fase di costruzione verrà descritta in dettaglio nel seguito.

Segue l'*analisi del modello* che prevede la deduzione per via analitica, in riferimento a determinate classi di problemi, di alcune importanti proprietà; le principali sono:

- *esistenza* della soluzione ottima;
- *condizioni di ottimalità*, cioè una caratterizzazione analitica della soluzione ottima;
- *stabilità* delle soluzioni al variare dei dati o di eventuali parametri presenti.

La successiva fase di *soluzione* avviene mediante opportuni algoritmi di calcolo e la soluzione numerica così ottenuta deve poi essere interpretata dal punto di vista applicativo in modo da evitare che abbia scarso rilievo pratico; in questo caso le eventuali cause di inaccettabilità devono essere inglobate nel modello stesso costruendo così un nuovo modello più completo del precedente. Tale “*validazione*” del modello può avvenire attraverso una *verifica sperimentale* oppure con metodi di *simulazione*. La definizione di un modello si configura quindi come un processo di raffinamento iterativo, che può essere schematizzato come rappresentato in Figura 1.1.

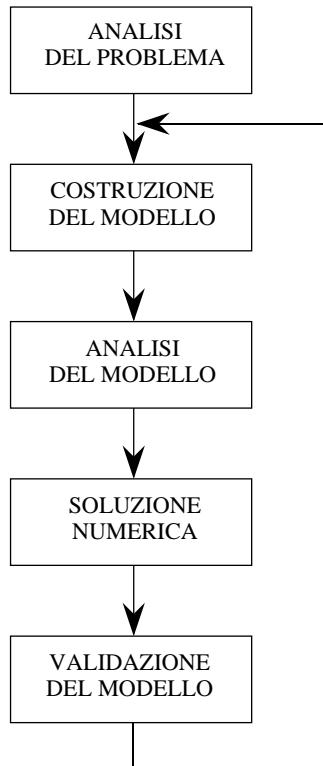


Figura 1.1: Fasi dell'approccio modellistico

Costruzione di un modello matematico

L'approccio modellistico per risolvere un problema di decisione necessita come primo passo della costruzione di un adeguato modello matematico. Infatti, come già discusso in precedenza, solo un modello costruito tenendo presente tutte le caratteristiche essenziali del fenomeno che si sta studiando permette di comprendere gli aspetti più importanti e di esercitare un intervento pratico efficace.

Nella fase di costruzione del modello matematico si deve fornire una descrizione formalizzata del problema di decisione facendo uso del linguaggio formale della matematica. Si dovrà cercare, quindi, una corrispondenza tra relazioni del mondo reale (relazioni tecnologiche, leggi fisiche, vincoli di mercato, etc.) e relazioni matematiche (equazioni, disequazioni, dipendenze logiche, etc.).

$$\boxed{\text{relazioni del mondo reale}} \quad \longleftrightarrow \quad \boxed{\text{relazioni matematiche}}$$

La costruzione di un modello richiede, quindi, scelte e valutazioni in modo da evidenziare gli aspetti più significativi del problema reale e che meglio sono suscettibili di una formalizzazione matematica. Tale procedimento di scelta spesso non è riconducibile ad un procedimento sistematico e quindi è necessario che chi costruisce il modello abbia da un lato una conoscenza approfondita del settore applicativo per evitare che le risposte ottenute dal modello abbiano scarsa rilevanza pratica; dall'altro deve avere una notevole conoscenza dei metodi matematici disponibili per la ricerca della soluzione per evitare che la formulazione matematica porti ad un problema per il quale non esistono algoritmi risolutivi utilizzabili.

È importante ribadire che un modello è definito per mezzo delle relazioni che lo costituiscono ed è quindi necessario che tali relazioni siano il più possibile indipendenti dai dati introdotti nel modello; questo perché uno stesso modello deve poter essere usato in differenti occasioni con dati (cioè costi, disponibilità di risorse, limiti tecnologici, etc.) diversi. Lo studio di questo aspetto, come già detto, rientra nella fase di analisi del modello sotto il nome di analisi della stabilità del modello rispetto ai dati introdotti.

Vantaggi dell'approccio modellistico

Esistono diverse ragioni per adottare l'approccio modellistico per la soluzione di problemi: si riassumono di seguito le principali.

- *Possibilità di risolvere matematicamente il problema.*

Grazie al modello è possibile analizzare matematicamente il problema ed ottenere così una soluzione che, soprattutto in riferimento a scopi di pianificazione, permette di adottare strategie che da una sola analisi strutturale del problema non apparirebbero evidenti o che a volte potrebbero essere perfino controidintuitive.

- *Maggiore comprensione del problema.*

Il modello è una rappresentazione semplificata del problema e spesso la sua costruzione consente di individuare proprietà strutturali del problema che altrimenti non sarebbero affatto evidenti.

- *Deduzione analitica di importanti proprietà.*

Nella fase di analisi del modello è possibile dedurre per via analitica alcune importanti proprietà del problema sulla base dei risultati disponibili per la classe di problemi a cui si fa riferimento.

- *Possibilità di simulazioni.*

Con un modello è possibile effettuare esperimenti che spesso non è possibile effettuare direttamente nella realtà; ad esempio, l'uso di un modello consente di studiare gli effetti dell'adozione di una particolare misura economica in un paese senza la necessità di sperimentarla direttamente.

Critiche all'approccio modellistico

Le principali critiche all'approccio modellistico e, quindi, alla costruzione di modelli per la soluzione di problemi di decisione possono essere sintetizzate nei seguenti due punti:

- Impossibilità di quantificare soddisfacentemente con opportuni valori numerici alcuni dati richiesti dal modello; questo accade, ad esempio, nel tentativo di quantificare con un costo o con un profitto alcuni valori sociali soprattutto in relazione a scopi di pianificazione.
- La qualità delle risposte che un modello produce potrebbero dipendere profondamente dall'accuratezza dei dati introdotti.

Il primo punto riguarda la possibilità di dover trattare concetti non facilmente quantificabili, ma ogni approccio scientifico può difficilmente evitare tale difficoltà; il modo migliore per superare tale problema consiste nell'incorporare tale quantificazione nel modello stesso.

La seconda critica riguarda la possibile mancanza di precisione di alcuni dei dati immessi nel modello; tale critica è meno rilevante della precedente, in quanto anche se alcuni dati introdotti sono poco accurati, è ancora possibile che la struttura del modello sia tale da garantire che la soluzione sia sufficientemente accurata.

All'estremo opposto di queste critiche si può collocare un atteggiamento di totale fiducia del modello che induca ad accettare la prima risposta prodotta dal modello senza ulteriori analisi. Tale atteggiamento, in realtà molto raro, è assai pericoloso in quanto tale risposta potrebbe rappresentare un piano operativo non accettabile nella realtà; in tal caso i motivi della non accettabilità devono essere evidenziati e incorporati in un nuovo modello modificato: si tratta, in realtà, della già citata fase di validazione del modello che quindi non può essere trascurata e che costituisce un valido mezzo per costruire modelli sempre più completi e significativi.

In conclusione, come spesso accade, l'atteggiamento corretto si colloca tra le due situazioni estreme precedentemente citate e consiste nel considerare la costruzione del modello un mezzo assai utile per affrontare un problema di decisione: rimane il fatto che la qualità delle risposte che un modello produce dipende dall'accuratezza della sua struttura e quindi non è trascurabile la fase di validazione che consente di interpretare la soluzione numerica ottenuta ed eventualmente permette di completare il modello introducendo elementi trascurati in una prima fase, in assenza dei quali la soluzione risulta non accettabile oppure di scarso rilievo dal punto di vista applicativo.

1.2 Modelli di Ottimizzazione

In questa sezione ci occuperemo più nel dettaglio di quei particolari modelli matematici noti come *Modelli di Ottimizzazione* che rivestono un ruolo centrale nella RO. In termini generali, data una funzione $f: \mathbb{R}^n \rightarrow \mathbb{R}$, ed $S \subseteq \mathbb{R}^n$, un *problema di Ottimizzazione* può essere formulato nella forma¹

$$\left\{ \begin{array}{l} \min f(x) \\ x \in S. \end{array} \right. \quad (PO)$$

Quindi un problema di Ottimizzazione consiste nel determinare, se esiste, un punto di minimo della funzione f tra i punti dell'insieme S . I problemi di ottimizzazione sono spesso denominati, con terminologia equivalente, problemi di Programmazione Matematica.

La funzione f viene chiamata *funzione obiettivo* e l'insieme S *insieme ammissibile* cioè l'insieme delle possibili soluzioni del problema. Un punto $x \in S$ si chiama *soluzione ammissibile*.

¹Si parlerà indifferentemente di problemi di massimo o di minimo in quanto vale $\min_{x \in S} f(x) = -\max_{x \in S} (-f(x))$.

L'insieme ammissibile S è un sottoinsieme di \mathbb{R}^n e quindi $x = (x_1, x_2, \dots, x_n)^T$ è una variabile vettoriale n -dimensionale e la funzione obiettivo f è una funzione di n variabili reali $f(x_1, x_2, \dots, x_n)$.

Si riportano di seguito alcune definizioni fondamentali riguardanti i problemi di Ottimizzazione.

Definizione 1.2.1 *Il problema di ottimizzazione (PO) si dice inammissibile se $S = \emptyset$, cioè se non esistono soluzioni ammissibili.*

Definizione 1.2.2 *Il problema di ottimizzazione (PO) si dice illimitato (inferiormente) se comunque scelto un valore $M > 0$ esiste un punto $x \in S$ tale che $f(x) < -M$.*

Un esempio di PO illimitato inferiormente è dato da $f(x) = x^3$ e $S = \mathbb{R}$. Infatti, al tendere di x a $-\infty$ la funzione obiettivo tende anch'essa a $-\infty$. Notiamo che se, con la stessa funzione obiettivo, si cambia l'insieme S , e si pone $S = \{x : x \geq 0\}$, il problema non è più illimitato inferiormente.

Definizione 1.2.3 *Si dice che il problema di ottimizzazione (PO) ammette soluzione ottima (finita) se esiste un $x^* \in S$ tale che risulti $f(x^*) \leq f(x)$ per ogni $x \in S$. Il punto x^* è detto soluzione ottima o minimo globale e il corrispondente valore $f(x^*)$ di dice valore ottimo.*

Per esempio, se si pone $f = x^2$ e $S = \mathbb{R}$, l'ottimo è l'origine, e il corrispondente valore ottimo è zero. Se si prende $S = \{x : x \geq 2\}$, l'ottimo è 2 e il valore ottimo 4.

All'interno dei problemi di Ottimizzazione si possono distinguere le seguenti importanti classi di problemi:

• **Problemi di Ottimizzazione Continua.**

Le variabili possono assumere tutti i valori reali ($x \in \mathbb{R}^n$); ed inoltre si parla di problemi di ottimizzazione continua

- *vincolata* se $S \subset \mathbb{R}^n$
- *non vincolata* se $S = \mathbb{R}^n$.

• **Problemi di Ottimizzazione Discreta.**

Le variabili sono vincolate ad essere numeri interi ($x \in \mathbb{Z}^n$); si possono distinguere all'interno di questa classe di problemi altre due classi:

- *programmazione a numeri interi* se $S \subseteq \mathbb{Z}^n$
- *ottimizzazione booleana* se $S \subseteq \{0, 1\}^n$.

• **Problemi misti.**

Solo alcune delle variabili sono vincolate ad essere intere.

Di solito l'insieme ammissibile S viene descritto da una numero finito di diseguaglianze del tipo $g(x) \leq b$, dove g è una funzione definita su \mathbb{R}^n a valori reali e $b \in \mathbb{R}$. Cioè, formalmente, date m funzioni $g_i : \mathbb{R}^n \rightarrow \mathbb{R}$, $i = 1, \dots, m$ ed m scalari $b_i \in \mathbb{R}$, $i = 1, \dots, m$ si esprime S nella forma

$$S = \{x \in \mathbb{R}^n \mid g_1(x) \leq b_1, g_2(x) \leq b_2, \dots, g_m(x) \leq b_m\}.$$

Ogni diseguaglianza $g_i(x) \leq b_i$ prende nome di *vincolo* e l'insieme ammissibile è quindi formato da tutti quei punti $x \in \mathbb{R}^n$ che sono soluzione del sistema di diseguaglianze

$$\begin{cases} g_1(x) \leq b_1 \\ g_2(x) \leq b_2 \\ g_3(x) \leq b_3 \\ \vdots \\ g_m(x) \leq b_m \end{cases}$$

Osservazione 1.2.4 In questa formulazione dell'insieme S si sono utilizzati vincoli di diseguaglianza nella forma di minore o uguale, ma è chiaro che questa notazione include i casi in cui i vincoli sono espressi con vincoli di diseguaglianza nella forma di maggiore o uguale e vincoli di uguaglianza; infatti si può sempre trasformare un vincolo di maggiore o uguale del tipo $g(x) \geq b$ in un vincolo di minore o uguale semplicemente riscrivendolo nella forma $-g(x) \leq -b$. Inoltre un vincolo di uguaglianza $g(x) = b$ può essere riscritto nella forma equivalente delle due diseguaglianze $g(x) \leq b$ e $-g(x) \leq -b$.

Quindi si può riscrivere il problema di ottimizzazione (PO) nella forma

$$\begin{cases} \min f(x) \\ g_i(x) \leq b_i, \quad i = 1, \dots, m. \end{cases} \quad (1.1)$$

I punti dell'insieme ammissibile di questo tipo di problemi sono quelli per i quali tutti i vincoli sono soddisfatti cioè tutti quei punti x tali che tutte le diseguaglianze $g_i(x) \geq b_i, i = 1, \dots, m$ sono verificate.

I problemi di Programmazione Matematica si possono classificare in base alla struttura delle funzioni che li definiscono; si parla di

- problema di *Programmazione Lineare (PL)* se la funzione obiettivo $f(x)$ e tutte le funzioni che definisco i vincoli $g_i(x), i = 1, \dots, m$ sono *lineari*, cioè esprimibili nella forma

$$c_1x_1 + c_2x_2 + \dots + c_nx_n;$$

- problema di *Programmazione Non Lineare (PNL)* se almeno una delle funzioni che definiscono un problema di Programmazione Matematica non è lineare.

Si formalizzano nella definizione che segue alcune semplici concetti riguardanti i vincoli di un problema di Programmazione Matematica.

Definizione 1.2.5 Si consideri un vincolo di diseguaglianza del tipo $g(x) \geq b$; esso si dice *violato* in un punto \bar{x} se $g(\bar{x}) < b$; esso si dice *attivo* in un punto \bar{x} se $g(\bar{x}) = b$.

Alcuni esempi di problemi di Programmazione Matematica sono i seguenti:

Esempio 1.2.6 Si consideri una funzione obiettivo di due variabili $f(x_1, x_2) = x_1 + x_2$ che si vuole minimizzare, con i vincoli $2x_1 + x_2 \geq 1, x_1 \geq 0, x_2 \geq 0$. Si ottiene il problema

$$\begin{cases} \min x_1 + x_2 \\ 2x_1 + x_2 \geq 1 \\ x_1 \geq 0 \\ x_2 \geq 0 \end{cases}$$

che è nella forma (1.1) dove $g_1(x_1, x_2) = 2x_1 + x_2, g_2(x_1, x_2) = x_1, g_3(x_1, x_2) = x_2, b_1 = 1, b_2 = b_3 = 0$. L'insieme ammissibile è descritto attraverso questi tre vincoli e poiché tutte le funzioni che compaiono sono lineari nelle variabili x_1 e x_2 , questo problema è un problema di Programmazione Lineare.

Esempio 1.2.7 Si consideri una funzione obiettivo $f(x_1, x_2) = (x_1 - \frac{1}{2})^2 + (x_2 - \frac{1}{2})^2$ che si vuole massimizzare, con i vincoli $x_1 + x_2 \geq 1$, $x_1 \leq 1$, $x_2 \leq 1$. Si ottiene il problema

$$\begin{cases} \max (x_1 - \frac{1}{2})^2 + (x_2 - \frac{1}{2})^2 \\ x_1 + x_2 \geq 1 \\ x_1 \leq 1 \\ x_2 \leq 1 \end{cases}$$

che è un problema di Programmazione Non Lineare (quadratico).

Esempio 1.2.8 Si consideri una funzione obiettivo $f(x_1, x_2) = 3x_1^3 + 7x_1^2 + x_2$ che si vuole minimizzare, con vincoli $x_1 + x_2 \leq \frac{1}{2}$, $x_1 \geq 0$, $x_2 \geq 1$. Si ottiene il problema

$$\begin{cases} \min 3x_1^3 + 7x_1^2 + x_2 \\ x_1 + x_2 \leq \frac{1}{2} \\ x_1 \geq 0 \\ x_2 \geq 1 \end{cases}$$

che è un problema di Programmazione Non Lineare che può essere facilmente ricondotto nella forma (1.1) riscrivendo gli ultimi due vincoli nella forma $-x_1 \leq 0$ e $-x_2 \leq 0$.

Esempio 1.2.9 Si consideri una funzione obiettivo $f(x_1, x_2) = x_1 + x_2$ che si vuole minimizzare sulla regione ammissibile descritta dal vincolo di uguaglianza $4x_1 - x_2 = -2$. Il problema di Programmazione Lineare risultante è

$$\begin{cases} \min x_1 + x_2 \\ 4x_1 - x_2 = -2 \end{cases}$$

che è un problema di Programmazione Lineare con un solo vincolo di uguaglianza.

In generale, la costruzione formale di un modello di Programmazione Matematica si effettua a partire da una descrizione logica e qualitativa di un problema di decisione e richiede di:

1. Associare opportune *variabili di decisione* di decisione alle grandezze reali. Tali variabili costituiscono le incognite del problema.
2. Esprimere quantitativamente i *legami* esistenti tra le variabili e le *limitazioni* derivanti da considerazioni di carattere fisico, economico, etc. Tali legami e limitazioni definiscono i *vincoli*. L'insieme dei valori delle variabili per cui i vincoli sono soddisfatti costituisce l'*insieme ammissibile*.
3. Esprimere formalmente l'*obiettivo* che si intende minimizzare o massimizzare.

1.3 Esempi di costruzione di modelli di Programmazione Matematica

Come primi esempi di costruzione di modelli verranno ora analizzati un semplice problema di pianificazione della produzione, un problema di pianificazione degli investimenti e un problema di progettazione industriale.

Esempio 1.3.1 Un'industria chimica fabbrica 4 tipi di fertilizzanti, **Tipo 1, Tipo 2, Tipo 3, Tipo 4**, la cui lavorazione è affidata a due reparti dell'industria: il reparto produzione e il reparto confezionamento. Per ottenere fertilizzante pronto per la vendita è necessaria naturalmente la lavorazione in entrambi i reparti. La tabella che segue riporta, per ciascun tipo di fertilizzante i tempi (in ore) necessari di lavorazione in ciascuno dei reparti per avere una tonnellata di fertilizzante pronto per la vendita.

	Tipo 1	Tipo 2	Tipo 3	Tipo 4
Reparto produzione	2	1.5	0.5	2.5
Reparto confezionamento	0.5	0.25	0.25	1

Dopo aver dedotto il costo del materiale grezzo, ciascuna tonnellata di fertilizzante dà i seguenti profitti (prezzi espressi in Euro per tonnellata)

	Tipo 1	Tipo 2	Tipo 3	Tipo 4
profitti netti	250	230	110	350

Determinare le quantità che si devono produrre settimanalmente di ciascun tipo di fertilizzante in modo da massimizzare il profitto complessivo, sapendo che settimanalmente il reparto produzione può lavorare al più 100 ore mentre il reparto confezionamento può lavorare al più 50 ore settimanali.

Analisi del problema e costruzione del modello.

Si tratta di un problema di pianificazione della produzione industriale in cui le incognite, che saranno le variabili del problema, sono le quantità di fertilizzante di ciascun tipo che si devono produrre. Costruiamo un modello di Programmazione Matematica rappresentante il problema in analisi supponendo di voler pianificare la produzione settimanale.

- *Variabili di decisione.* È naturale introdurre le variabili reali x_1, x_2, x_3, x_4 rappresentanti rispettivamente le quantità di prodotto del **Tipo 1, Tipo 2, Tipo 3, Tipo 4** da fabbricare in una settimana.
- *Funzione Obiettivo.* Ciascuna tonnellata di fertilizzante contribuisce al profitto totale secondo la tabella data. Quindi il profitto totale sarà

$$250x_1 + 230x_2 + 110x_3 + 350x_4. \quad (1.2)$$

L'obiettivo dell'industria sarà quello di scegliere le variabili x_1, x_2, x_3, x_4 in modo che l'espressione (2.2) del profitto sia massimizzata. La (2.2) rappresenta la funzione obiettivo.

- *Vincoli.* Ovviamente la capacità produttiva della fabbrica limita i valori che possono assumere le variabili x_j , $j = 1, \dots, 4$; infatti si ha una capacità massima lavorativa in ore settimanali di ciascun reparto. In particolare per il reparto produzione si hanno a disposizione al più 100 ore settimanali e poiché ogni tonnellata di fertilizzante di **Tipo 1** utilizza il reparto produzione per 2 ore, ogni tonnellata di fertilizzante di **Tipo 2** utilizza il reparto produzione per 1.5 ore e così via per gli altri tipi di fertilizzanti si dovrà avere

$$2x_1 + 1.5x_2 + 0.5x_3 + 2.5x_4 \leq 100. \quad (1.3)$$

Ragionando in modo analogo per il reparto confezionamento si ottiene

$$0.5x_1 + 0.25x_2 + 0.25x_3 + x_4 \leq 50. \quad (1.4)$$

Le espressioni (2.3), (2.4) costituiscono i vincoli del modello. Si devono inoltre esplicitare vincoli dovuti al fatto che le variabili x_j , $j = 1, \dots, 4$ rappresentando quantità di prodotto non possono essere negative e quindi vanno aggiunti i vincoli di non negatività

$$x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0.$$

La formulazione finale sarà quindi

$$\begin{cases} \max (250x_1 + 230x_2 + 110x_3 + 350x_4) \\ 2x_1 + 1.5x_2 + 0.5x_3 + 2.5x_4 \leq 100 \\ 0.5x_1 + 0.25x_2 + 0.25x_3 + x_4 \leq 50 \\ x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0. \end{cases}$$

Questa formulazione è un problema matematico ben definito e costituisce il modello di Programmazione Matematica rappresentante il problema di pianificazione della produzione industriale in analisi. Si tratta, in questo caso, di un problema di programmazione lineare. \square

Esempio 1.3.2 – CAPITAL BUDGETING. Supponiamo di dover investire £ 1000 sul mercato finanziario. Supponiamo inoltre che il mercato offra tre tipi diversi di investimenti **A, B, C** ciascuno caratterizzato da un prezzo d'acquisto e da un rendimento netto, che sono riassunti nella seguente tabella:

	A	B	C
costo	750	200	800
rendimento	20	5	10

Si vuole decidere quali degli investimenti effettuare per massimizzare il rendimento sapendo che gli investimenti **A, B, C** non si possono effettuare in modo parziale cioè non sono frazionabili.

Analisi del problema e costruzione del modello.

Si tratta di un problema di pianificazione degli investimenti. Si devono definire formalmente le variabili di decisione, l'insieme delle soluzioni ammissibili e la funzione obiettivo.

– *Variabili di decisione.* Si tratta quindi di esprimere matematicamente la scelta elementare: effettuare o non effettuare l'investimento. Una scelta naturale delle variabili di decisione è la seguente:

$$x_i = \begin{cases} 0 & \text{non si effettua l'investimento } i\text{--esimo} \\ 1 & \text{si effettua l'investimento } i\text{--esimo} \end{cases} \quad i = \mathbf{A}, \mathbf{B}, \mathbf{C} \quad (1.5)$$

– *Insieme ammissibile.* In base alla definizione delle variabili, le possibili scelte compatibili con il nostro budget sono:

- (0) non si effettuano investimenti $x_A = x_B = x_C = 0$
- (1) si effettua l'investimento **A**; $x_A = 1, x_B = x_C = 0$
- (2) si effettua l'investimento **B**; $x_A = 0, x_B = 1, x_C = 0$
- (3) si effettua l'investimento **C**; $x_A = x_B = 0, x_C = 1$
- (4) si effettuano gli investimenti **A** e **B**; $x_A = x_B = 1, x_C = 0$
- (5) si effettuano gli investimenti **B** e **C**; $x_A = 0, x_B = x_C = 1$.

Notiamo che le possibilità **A, C** e **A, B, C** non sono ammissibili in quanto il costo supera la nostra disponibilità.

L'insieme ammissibile, ovvero l'insieme delle possibili scelte (0) – (5) è dato da:

$$S = \left\{ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \right\}$$

Si tratta quindi di un sottoinsieme dei vettori di \mathbb{R}^3 a componenti 0 – 1 ovvero

$$S \subseteq \{0, 1\}^3$$

– *Funzione obiettivo.* L’obiettivo che ci proponiamo è la massimizzazione del rendimento totale. Quindi dobbiamo esprimere la funzione obiettivo che corrisponde al rendimento netto relativo alla scelta di $x = (x_A, x_B, x_C)^T$ in S , cioè:

$$f(x) = 20x_A + 5x_B + 10x_C.$$

È possibile ottenere la soluzione ottima valutando esaustivamente la funzione obiettivo per ogni elemento di S , ottenendo in relazione alle possibili scelte:

- (0) $f_0 = 0$
- (1) $f_1 = 20$
- (2) $f_2 = 5$
- (3) $f_3 = 10$
- (4) $f_4 = 25$
- (5) $f_5 = 15.$

La soluzione ottima è ovviamente quella corrispondente alla scelta (4), cioè all’effettuare gli investimenti **A** e **B**, con valore della funzione obiettivo pari a £25.

Questo *non è un modello corretto* per due motivi:

1. *L’insieme ammissibile S è rappresentato in modo estensivo*, cioè elencando tutte le soluzioni ammissibili. In questo caso la cardinalità dell’insieme ammissibile è al più quella di $\{0, 1\}^3$ cioè 2^3 , ma in generale, se la dimensione del problema fosse più grande sarebbe impossibile valutare esaustivamente le soluzioni del problema. Se, ad esempio, il numero degli investimenti fosse stato 100 (che dal punto di vista delle applicazioni reali è del tutto verosimile) la cardinalità dell’insieme ammissibile sarebbe stata 2^{100} e per la valutazione di 2^{100} possibilità anche supponendo di utilizzare un calcolatore che effettui 10^{10} valutazioni al secondo (velocità superiore a quella raggiungibile dai calcolatori attuali) occorrerebbero 10^{20} secondi, cioè 3000 miliardi di anni !
2. *Il modello non è indipendente dai dati del problema*, cioè cambiando i dati del problema (prezzi e/o rendimenti) sarebbe necessario cambiare completamente il modello.

In generale, in un modello corretto, si cerca di dare una *rappresentazione intensiva* dell’insieme ammissibile S , cioè individuare le proprietà $P(x)$ che consentono di distinguere le soluzioni ammissibili dagli elementi dell’insieme $\{0, 1\}^3$ che non lo sono. Si vuole quindi scrivere l’insieme S in una forma del tipo:

$$S = \{x \in \{0, 1\}^3 : \text{vale la proprietà } P(x)\}.$$

Nell’esempio, la proprietà distintiva degli elementi di S è il costo complessivo che non deve essere superiore a £1000. Possiamo esprimere matematicamente questa relazione come:

$$P(x) : 750x_A + 200x_B + 800x_C \leq 1000$$

e quindi l’insieme ammissibile si può scrivere

$$S = \{x = (x_A, x_B, x_C)^T \in \{0, 1\}^3 : 750x_A + 200x_B + 800x_C \leq 1000\}.$$

In conclusione, il modello matematico corretto per il nostro problema di decisione è:

$$\begin{cases} \max (20x_A + 5x_B + 10x_C) \\ 750x_A + 200x_B + 800x_C \leq 1000 \\ x_i \in \{0, 1\} \quad i = \mathbf{A}, \mathbf{B}, \mathbf{C}. \end{cases}$$

Si tratta di un problema di programmazione lineare intera.

□

Esempio 1.3.3 Un'industria deve costruire un silos di forma cilindrica per contenere grandi quantitativi di un liquido che verrà poi distribuito in piccole confezioni pronte per la vendita al minuto. Tale silos deve essere posto in un magazzino appoggiato su una delle basi. Tale magazzino è a pianta rettangolare di dimensioni metri 20×10 ed ha un tetto spiovente lungo il lato di 10 metri, che ha altezza massima di metri 5 e altezza minima di metri 3. Per costruire questo silos deve essere usato del materiale plastico sottile flessibile che può essere tagliato, modellato e incollato saldamente. Sapendo che si dispone di non più di 200 m^2 di tale materiale plastico si costruisca un modello che permetta di determinare le dimensioni del silos (raggio di base ed altezza) in modo da massimizzare la quantità di liquido che può esservi contenuto.

Analisi del problema e costruzione del modello.

Si tratta di determinare il dimensionamento ottimale di un contenitore cilindrico per uso industriale cercando di massimizzare il suo volume tenendo presente che deve essere contenuto in un magazzino di dimensioni fissate.

Si devono definire formalmente le variabili di decisione, l'insieme delle soluzioni ammissibili e la funzione obiettivo.

– *Variabili di decisione.* È immediato introdurre due variabili x e y che rappresentano rispettivamente la lunghezza (in metri) del raggio di base e dell'altezza del contenitore cilindrico.

– *Funzione obiettivo.* La funzione obiettivo è rappresentata dal volume del contenitore cilindrico ed è data da

$$\pi x^2 y.$$

– *Vincoli.* Il diametro della base non può superare le dimensioni del magazzino e quindi deve essere

$$2x \leq 10.$$

La limitazione dell'altezza del contenitore varia al variare del diametro di base in quanto il tetto è spiovente. Dato che la pendenza del tetto è del 20%, dovrà risultare

$$y \leq 5 - 0.2 \cdot 2x.$$

Inoltre disponendo solo di una quantità limitata di materiale plastico la superficie totale del contenitore cilindrico non può superare 200 m^2 e quindi deve risultare

$$2\pi x^2 + 2\pi xy \leq 200.$$

Si devono infine esplicitare i vincoli di non negatività $x \geq 0$, $y \geq 0$.

La formulazione complessiva risulta quindi

$$\begin{cases} \max \pi x^2 y \\ x \leq 5 \\ y \leq 5 - 0.2 \cdot 2x \\ 2\pi x^2 + 2\pi xy \leq 200 \\ x \geq 0, \quad y \geq 0. \end{cases}$$

Il modello è quindi un modello di programmazione non lineare. □

Capitolo 2

La Programmazione Lineare

In questo capitolo esaminiamo in maniera in modo più dettagliato il problema di programmazione lineare e illustriamo una tecnica risolutiva per il caso di due sole variabili che aiuta a comprendere alcune delle caratteristiche più importanti dei problemi di programmazione lineare. Queste caratteristiche sono poi oggetto di una formalizzazione e generalizzazione, che suggeriscono anche i rudimenti per un metodo di soluzione.

2.1 Definizione del problema

Come già detto nel capitolo precedente, è possibile classificare i modelli di Programmazione Matematica in base alla struttura particolare che possono avere la funzione obiettivo e i vincoli. Riprendiamo qui, espandendola, la definizione di *problem di Programmazione Lineare* nei quali sia la funzione obiettivo, sia i vincoli sono rappresentati mediante funzioni lineari nelle variabili di decisione.

Preliminariamente, richiamiamo il concetto di *funzione lineare*.

Definizione 2.1.1 Una funzione reale di n variabili reali $f: \mathbb{R}^n \rightarrow \mathbb{R}$ si dice lineare se può essere scritta nella forma

$$c_1x_1 + c_2x_2 + \dots + c_nx_n$$

con c_1, \dots, c_n costanti reali.

Quindi

$$\begin{aligned} x_1 + 4x_2 - 3.5x_3 \\ -2x_1 + (\sin 4)x_2 + \pi x_3 - 4x_5, \end{aligned}$$

sono funzioni lineari, mentre

$$\begin{aligned} (x_1)^2 + 4x_2 - 3.5x_3 \\ x_1 + 4x_2 - 3.5e^{x_3} \\ -2x_1 + \sin x_2 + \pi x_3 - 4x_5, \end{aligned}$$

non sono funzioni lineari. Un problema di Programmazione Lineare (spesso abbreviato in PL) è caratterizzato da

- una singola *funzione obiettivo lineare* da minimizzare o massimizzare che può essere quindi scritta nella forma

$$f(x_1, \dots, x_n) = c_1x_1 + \dots + c_nx_n = \sum_{j=1}^n c_jx_j.$$

- un numero finito di *vincoli lineari* che, supponendo siano m , possono essere scritti nella forma

$$\begin{array}{llll} a_{11}x_1 + & \dots & + a_{1n}x_n & \leq b_1 \\ a_{21}x_1 + & \dots & + a_{2n}x_n & \leq b_2 \\ \vdots & \dots & \vdots & \vdots \\ a_{m1}x_1 + & \dots & + a_{mn}x_n & \leq b_m. \end{array}$$

Il fatto di impiegare solo vincoli di minore o uguale è, come visto nell'Osservazione 1.2.4, del tutto convenzionale, e infatti nei problemi di Programmazione Lineare alcuni (o tutti) i vincoli possono essere di maggiore uguale o di uguaglianza. Per esempio,

$$\begin{array}{ll} \max & x_1 + x_2 \\ & x_1 + x_2 \geq 1 \\ & x_1 + x_2 \leq 3 \\ & x_1 \geq 0, \quad x_2 \geq 0, \end{array}$$

e

$$\begin{array}{ll} \min & 2x_1 - x_2 + x_3 + 3x_4 \\ & x_1 + x_2 - x_4 = 1 \\ & x_1 + 2x_2 - x_3 + 2x_4 \leq 3 \\ & x_1 \geq 0, \quad x_2 \geq 0, \quad x_4 \geq 0, \end{array}$$

sono problemi di PL.

Introducendo il vettore $c \in \mathbb{R}^n$, definito $c = (c_1, \dots, c_n)^T$ e $x \in \mathbb{R}^n$ definito $x = (x_1, \dots, x_n)^T$ la funzione obiettivo può essere scritta in notazione vettoriale

$$c^T x.$$

Inoltre, introducendo la matrice $(m \times n)$

$$A = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \dots & a_{mn} \end{pmatrix}$$

e il vettore $b = (b_1, \dots, b_m)^T$ la formulazione completa di un generico problema di Programmazione Lineare può essere scritta nella forma

$$\begin{cases} \min c^T x \\ Ax \leq b. \end{cases}$$

La Programmazione Lineare è indubbiamente l'argomento centrale dell'Ottimizzazione e fra i vari modelli della Ricerca Operativa, la Programmazione Lineare è quello che viene più ampiamente utilizzato. Infatti, la PL non solo si applica a numerosi problemi reali che hanno di per sé una struttura lineare, ma è anche un importante strumento di supporto nell'analisi e nella risoluzione di problemi di programmazione matematica più complessi.

Il padre della PL viene comunemente, e giustamente, indicato in George Dantzig che per primo ne ideò, nel 1947, un algoritmo risolutivo (il metodo del Simplex, cui si fa cenno alla fine del capitolo). Tuttavia, alcuni dei concetti fondamentali della programmazione lineare possono essere fatti risalire molto più indietro nel tempo. Già Fourier, nel 1827, aveva studiato come trovare soluzioni ammissibili di un sistema di diseguaglianze lineari; un metodo di calcolo destinato a minimizzare gli errori d'osservazione e dovuto a Vallée Poussin (1910) presenta lati simili al metodo del simplex; infine lavori di von Neumann degli anni venti e trenta, sulla teoria dei giochi e su alcuni modelli economici, sono antecedenti diretti del

lavoro di Dantzig. Nel 1939, poi, il matematico sovietico Kantorovich aveva pubblicato (in russo) una monografia sulla programmazione della produzione che anticipa, sotto molti aspetti importanti, i temi trattati da Dantzig. Purtroppo questo lavoro fu a lungo ignorato in Occidente essendo riscoperto solo venti anni dopo, quando la PL aveva avuto un grande sviluppo.

La caratteristica di tutti i lavori antecedenti quelli di Dantzig era uno scarso interesse verso l'applicabilità pratica, dovuta principalmente all'impossibilità di effettuare i calcoli necessari. Il metodo del Simplex proposto da Dantzig si rivelò invece efficiente in pratica e questo, unitamente al simultaneo avvento dei calcolatori elettronici, decretò il successo della PL e, con esso, l'inizio dello sviluppo rigoglioso della Ricerca Operativa.

2.2 Semplici esempi di problemi di programmazione lineare

Chiariremo meglio i concetti finora esposti con alcuni esempi.

2.2.1 Un problema di produzione (allocazione di risorse)

Un'industria produce un bene di consumo in due versioni, normale e super. Su ogni unità venduta l'industria ha un profitto di 1200 L per il tipo normale e di 1500 L per il tipo super. Nella produzione sono utilizzati tre tipi di macchinari, che indichiamo convenzionalmente con A, B, C, e che settimanalmente non possono essere in esercizio per un numero di ore maggiore di quello indicato nella Tabella 2; per produrre una unità di prodotto è richiesto l'utilizzo delle macchine per il tempo indicato nella stessa tabella 2.

macchinario	ore per settimana	ore di lavoro per unità di prodotto	
		tipo normale	tipo super
A	1500	1.5	1.5
B	2000	0.8	1
C	1800	1	2

Tab. 2
Tempi di lavorazione

L'industria vuole pianificare la produzione settimanale in modo da massimizzare il profitto conseguito. Indicando rispettivamente con x_1 e x_2 le quantità del bene prodotto settimanalmente nella versione normale e nella versione super, il profitto settimanale è dato dalla funzione

$$z(x_1, x_2) = 1200x_1 + 1500x_2.$$

Per non superare la capacità lavorativa delle macchine di tipo A dovrà risultare

$$1.5x_1 + 1.5x_2 \leq 1500$$

e diseguaglianze analoghe valgono per le macchine di tipo B e C, per cui si ha

$$\begin{aligned} 0.8x_1 + x_2 &\leq 2000, \\ x_1 + 2x_2 &\leq 1800. \end{aligned}$$

Poiché infine deve risultare $x_1 \geq 0$, $x_2 \geq 0$, giungiamo alla formulazione del seguente problema di programmazione lineare:

$$\begin{aligned} \max z &= 1200x_1 + 1500x_2 \\ 1.5x_1 + 1.5x_2 &\leq 1500 \\ 0.8x_1 + x_2 &\leq 2000 \\ x_1 + 2x_2 &\leq 1800 \\ x_1 \geq 0, x_2 \geq 0. & \end{aligned}$$

2.2.2 Un problema di miscelazione

Consideriamo il problema di una industria alimentare che produce hamburger e che vuole minimizzare il costo delle materie prime, garantendo una buona qualità del prodotto. Ammettiamo che ogni hamburger debba pesare almeno 100 grammi e che l'impasto sia costituito da carne macinata di manzo e di maiale, in quantità espresse in grammi che indichiamo rispettivamente con x_1 e x_2 . Il macinato di manzo contiene l'80% di polpa e il 20% di grasso, e costa all'industria £6000 al Kg; il macinato di maiale contiene il 68% di polpa e il 32% di grasso e costa all'industria £3500 al Kg. Quanta carne di ciascun tipo dovrà impiegare l'industria in ogni hamburger se vuole minimizzare il costo della carne utilizzata ed evitare che il contenuto grasso dell'hamburger superi i 25 grammi? Tale problema può essere formulato come problema di programmazione lineare nel seguente modo. Come funzione obiettivo si può considerare il costo della carne utilizzata in ogni hamburger; tenendo conto del fatto che x_1 e x_2 sono espresse in grammi, il costo (espresso in migliaia) è dato da

$$z(x_1, x_2) = 6x_1 + 3,5x_2.$$

Il peso di ogni hamburger deve essere di almeno 100 grammi, e ciò vincola le variabili x_1 e x_2 nel seguente modo:

$$x_1 + x_2 \geq 100;$$

inoltre ogni hamburger conterrà $0.2x_1$ grammi di grasso proveniente dalla carne di manzo e $0.32x_2$ grammi di grasso provenienti dalla carne di maiale; poiché i grammi di grasso in totale devono essere in ogni hamburger non più di 25, si deve avere:

$$0.2x_1 + 0.32x_2 \leq 25.$$

Infine le quantità x_1 e x_2 sono per loro natura non negative, cioè deve risultare $x_1 \geq 0$, $x_2 \geq 0$. In definitiva il problema che l'industria deve risolvere è il seguente:

$$\begin{aligned} \min z &= 6x_1 + 3,5x_2 \\ x_1 + x_2 &\geq 100 \\ 0.2x_1 + 0.32x_2 &\leq 25 \\ x_1 &\geq 0 \\ x_2 &\geq 0, \end{aligned}$$

e questo è evidentemente un problema di programmazione lineare.

2.2.3 Un problema di trasporto

Consideriamo un'industria che produce un bene di consumo in due stabilimenti di produzione, situati rispettivamente a Pomezia e a Caserta. La produzione viene prima immagazzinata in due depositi, situati uno a Roma e l'altro a Napoli. Quindi i prodotti vengono distribuiti alla rete di vendita al dettaglio. Per ogni unità di prodotto, il costo del trasporto dallo stabilimento al deposito è dato dalla Tabella 1:

Trasporto	lire
Pomezia-Roma	1000
Pomezia-Napoli	3000
Caserta-Napoli	500
Caserta-Roma	3500

Tab. 1
Costi di trasporto

La capacità produttiva dei due stabilimenti è limitata, per cui ogni settimana il bene in questione non può essere prodotto in più di 10000 unità nello stabilimento di Pomezia e in più di 8000 unità nello stabilimento

di Caserta. Inoltre le statistiche di vendita informano che ogni settimana vengono vendute mediamente 11000 unità tramite il deposito di Roma e 4600 unità tramite il deposito di Napoli. L'industria vuole minimizzare il costo del trasporto della merce dagli stabilimenti ai depositi, assicurando che i depositi ricevano settimanalmente le quantità medie prima indicate.

Le variabili di decisione sono le quantità del bene di consumo trasportate settimanalmente, che possiamo associare alle variabili x_1, x_2, x_3, x_4 nel seguente modo:

$$\begin{array}{ll} \text{quantità trasportata da Pomezia a Roma} & x_1 \\ \text{quantità trasportata da Pomezia a Napoli} & x_2 \\ \text{quantità trasportata da Caserta a Napoli} & x_3 \\ \text{quantità trasportata da Caserta a Roma} & x_4 \end{array}$$

La funzione obiettivo è il costo sostenuto settimanalmente per il trasporto:

$$z(x_1, x_2, x_3, x_4) = 1000x_1 + 3000x_2 + 500x_3 + 3500x_4.$$

Poichè i due stabilimenti hanno capacità produttiva limitata deve essere

$$\begin{array}{l} x_1 + x_2 \leq 10000 \\ x_3 + x_4 \leq 8000. \end{array}$$

Poichè si vuole garantire il rifornimento medio settimanale, deve essere

$$\begin{array}{l} x_1 + x_4 = 11000 \\ x_2 + x_3 = 4600. \end{array}$$

Infine evidentemente deve risultare $x_i \geq 0$, $i = 1, 2, 3, 4$, e quindi il problema di programmazione lineare per l'industria dell'esempio è il seguente:

$$\begin{array}{ll} \min z = & 1000x_1 + 3000x_2 + 500x_3 + 3500x_4 \\ & x_1 + x_2 \leq 10000 \\ & x_3 + x_4 \leq 8000 \\ & x_1 + x_4 = 11000 \\ & x_2 + x_3 = 4600 \\ & x_i \geq 0, \quad i = 1, 2, 3, 4. \end{array}$$

I semplici problemi descritti hanno una funzione essenzialmente esemplificativa. In casi concreti, un problema di programmazione lineare può avere un numero di variabili di decisione e un numero di vincoli dell'ordine delle decine e centinaia di migliaia.

2.3 Interpretazione grafica di un Problema di PL

Quando il vettore x delle variabili di decisione ha solo due componenti il problema di programmazione lineare può essere facilmente risolto per via grafica. Gli esempi che seguono, analizzati graficamente, illustrano le situazioni tipiche che si presentano nella soluzione di problemi di programmazione lineare.

Ricordiamo preliminarmente che nel piano cartesiano con assi coordinati (x_1, x_2) l'equazione

$$ax_1 + bx_2 = c$$

individua una retta. Tale retta al crescere di c si muove parallelamente a se stessa nella direzione individuata dal vettore $[a \ b]^T$; e di conseguenza la disequazione

$$ax_1 + bx_2 \geq c$$

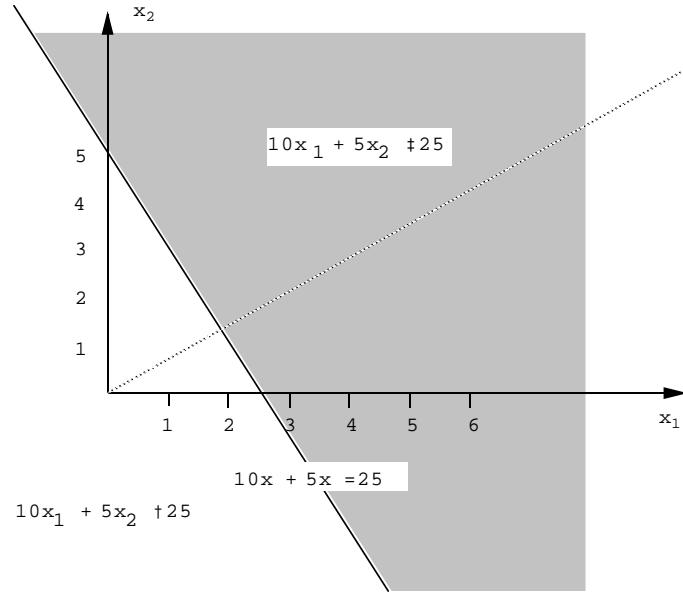


Figura 2.1: Retta e semipiani individuati da un'equazione lineare.

individua il semipiano posto rispetto alla retta $ax_1 + bx_2 = c$ dalla parte verso cui è orientato il vettore $[a \ b]^T$. Ad esempio, nella figura 2.1 sono rappresentati la retta $10x_1 + 5x_2 = 25$, il vettore $[10 \ 5]^T$, e, punteggiato, il semipiano in cui risulta $10x_1 + 5x_2 \geq 25$. Consideriamo ora un esempio di problema di programmazione lineare con due variabili di decisione:

$$\begin{aligned}
 \max \quad & 5x_1 + 10x_2 \\
 1) \quad & 10x_1 + 5x_2 \leq 25 \\
 2) \quad & 4x_1 + 10x_2 \leq 20 \\
 3) \quad & x_1 + 1.5x_2 \leq 4.5 \\
 & x \geq 0.
 \end{aligned}$$

I vincoli 1), 2) e 3) individuano i semipiani punteggiati rispettivamente nelle figure 2.2, 2.3 e 2.4.

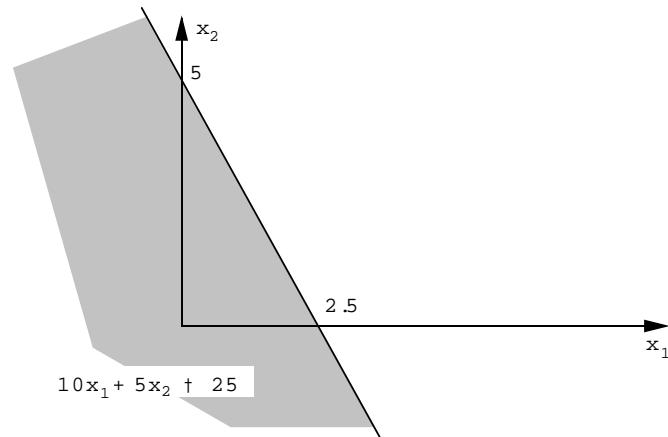


Figura 2.2: Regione individuata dal vincolo 1).

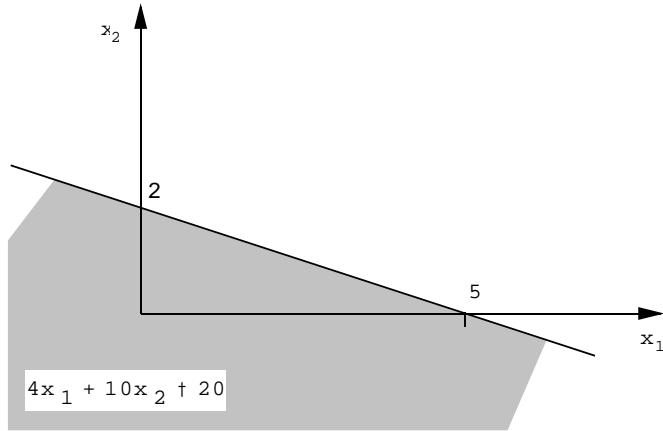


Figura 2.3: Regione individuata dal vincolo 2).

I tre vincoli insieme, e quelli di non negatività delle variabili x_1 e x_2 , individuano la regione del piano (x_1, x_2) punteggiata in figura 2.5, in cui nessun vincolo è violato; tale regione è detta *regione ammissibile*. Notiamo, per inciso, che ai fini della determinazione della regione ammissibile, il terzo vincolo è superfluo.

La retta $5x_1 + 10x_2 = z$ è rappresentata in figura 2.6 per $z = 10$, $z = 20$ e $z = 30$. Sovrapponendo le figure 2.5 e 2.6 si constata che al crescere di z le rette corrispondenti alla funzione obiettivo, dette anche *linee di livello*, intersecano la regione ammissibile finché risulta $z \leq 21.875$; non esistono invece punti della regione ammissibile in cui la funzione obiettivo assume un valore maggiore di 21.875. Questo valore è quindi il massimo che la funzione obiettivo può assumere compatibilmente con il soddisfacimento dei vincoli. Il massimo vincolato della funzione obiettivo è assunto nel punto $(x_1 = 1.875, x_2 = 1.25)$, come mostrato nella figura 2.7. In alcuni casi la soluzione del problema può non essere unica, nel senso che vi possono essere più punti del piano (x_1, x_2) che forniscono lo stesso valore massimo della funzione obiettivo. Ad esempio, se con i vincoli del caso precedente la funzione obiettivo, anziché quella data, fosse la $z = 4x_1 + 10x_2$, le rette corrispondenti all'equazione $4x_1 + 10x_2 = z$, al variare di z , sarebbero parallele al segmento del perimetro della regione ammissibile determinato dal secondo vincolo, e quindi ogni punto di questo segmento fornirebbe una soluzione non unica del problema. La figura 2.8 illustra questo caso. Nei due esempi precedenti la soluzione del problema esiste, anche se nel secondo caso non è unica. In altri problemi di programmazione lineare la soluzione può non esistere. Un primo caso in cui si verifica questa circostanza è quando la regione ammissibile risulta vuota. Ciò avviene, ad esempio, se si cambia il verso della diseguaglianza del vincolo 3). Come evidenziato nella figura 2.9, in tal caso nessun punto del piano in cui risulta $x_1 + 1.5x_2 \geq 4.5$ (zona tratteggiata) appartiene alla regione individuata dagli altri vincoli, e quindi nessun punto del piano soddisfa tutti i vincoli del problema.

In tal caso il problema non ha soluzione, indipendentemente dalla funzione obiettivo. Un altro caso in cui il problema di programmazione lineare può non avere soluzione è quando la regione ammissibile risulta illimitata. Se nell'esempio inizialmente considerato eliminiamo il vincolo di non negatività della variabile x_1 , la regione ammissibile diviene quella, illimitata, punteggiata in figura 2.10; in questo caso la soluzione del problema rimane invariata.

Se però richiediamo che la funzione obiettivo sia minimizzata anziché massimizzata, la stessa figura 2.10 evidenzia il fatto che nella regione ammissibile la funzione obiettivo può assumere valori arbitrariamente grandi in valore assoluto e negativi in segno. In altre parole, è sempre possibile trovare nella

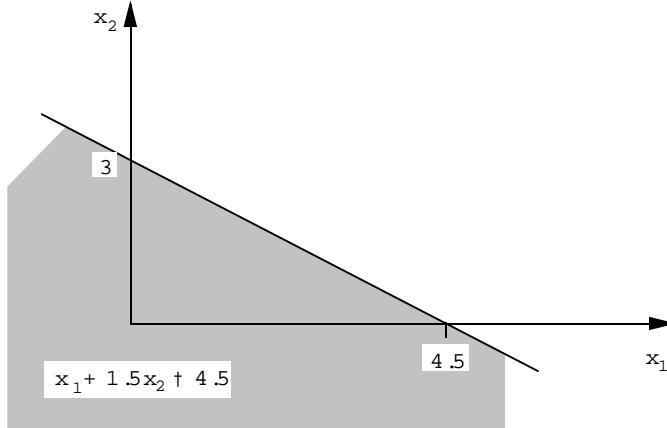


Figura 2.4: Regione individuata dal vincolo 3).

regione ammissibile un punto in cui la funzione obiettivo assume un valore più basso di qualsiasi valore dato, e quindi il problema non ha soluzione.

Riassumendo, l'analisi grafica suggerisce che, per un problema di PL, si possono presentare i seguenti casi:

1. la regione ammissibile è vuota: il problema non ha soluzione;
2. la regione ammissibile è illimitata e la funzione obiettivo è illimitata superiormente (se il problema è di massimizzazione) o inferiormente (se il problema è di minimizzazione): il problema non ha soluzione;
3. il problema ha soluzione ottima:
 - unica;
 - non unica (e sono infinite).

Con riferimento al punto (2) notiamo che se la regione ammissibile è illimitata non necessariamente il problema non ha soluzione (funzione obiettivo illimitata). Osserviamo infine che sembra lecito affermare, sempre in base agli esempi visti, che:

1. se la regione ammissibile è non vuota, essa risulta essere un insieme convesso;
2. se il problema ha soluzioni, almeno una di esse si trova in un vertice del poligono convesso costituito dalla regione ammissibile.

Tutte queste osservazioni sono in effetti *sostanzialmente* corrette. Nel prossimo paragrafo discuteremo questo tipo di argomenti nel caso di problemi di PL con un numero qualunque di variabili, dando le definizioni e precisazioni necessarie.

Esempio 2.3.1 Risolviamo graficamente il problema di miscelazione descritto nel paragrafo 2.2.2.

La regione ammissibile del problema è rappresentata punteggiata in figura 2.11; nella stessa figura sono rappresentate le linee di livello corrispondenti ai valori della funzione obiettivo z pari a 420 e a 630, oltre a quella corrispondente al valore ottimo. La soluzione si trova nel punto $x_1 = 58.33$, $x_2 = 41.66$ ove per la funzione obiettivo si ha il valore minimo vincolato $z = 495.83$. \square

Esempio 2.3.2 Risolviamo graficamente il problema di produzione descritto nel paragrafo 2.2.1. La regione ammissibile del problema è rappresentata punteggiata in figura 2.12; nella stessa figura sono

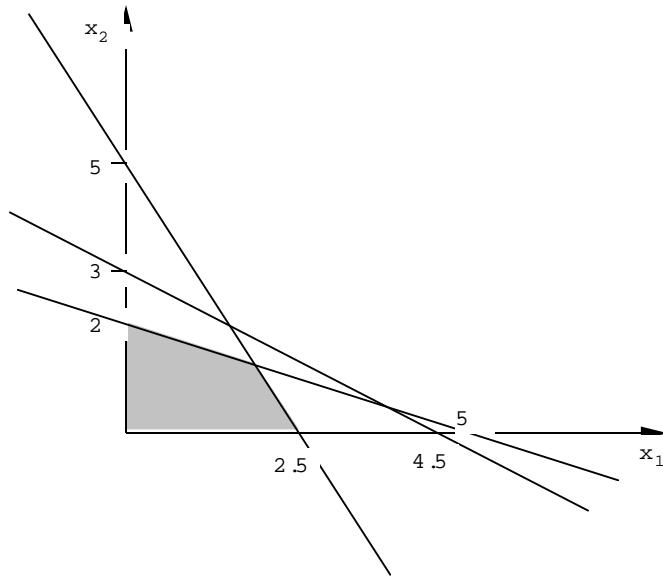


Figura 2.5: Regione ammissibile del problema.

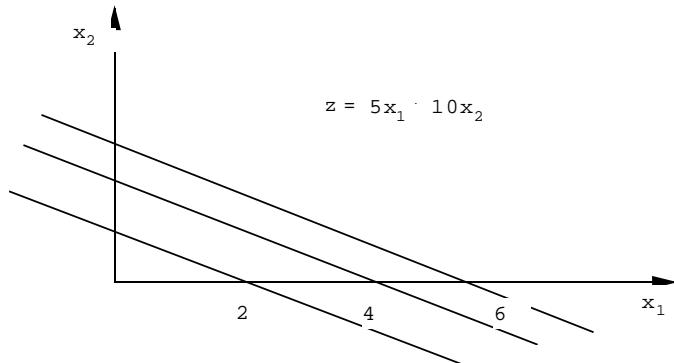


Figura 2.6: Linee di livello della funzione obiettivo.

rappresentate le linee di livello corrispondenti ai valori della funzione obiettivo z pari a 600000 e a 2400000, oltre a quella corrispondente al valore ottimo. La soluzione si trova nel punto $x_1 = 200$, $x_2 = 800$, ove per la funzione obiettivo si ha il valore massimo vincolato $z = 1440000$. \square

Esercizio 2.3.3 Risovi il problema di programmazione lineare determinato dalla funzione obiettivo da massimizzare $z = x_1 + 0.5x_2$ e dalla regione ammissibile punteggiata in figura 2.13.

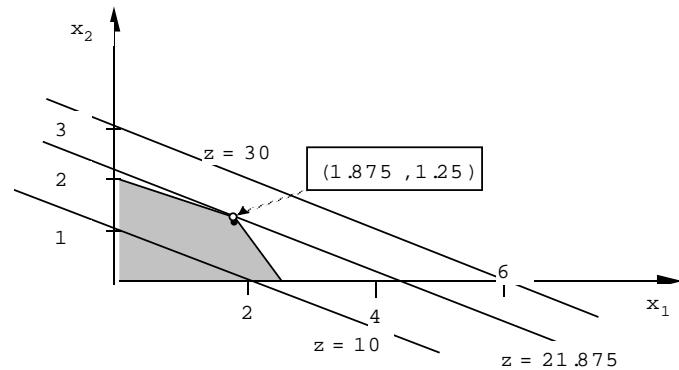


Figura 2.7: Soluzione grafica del problema.

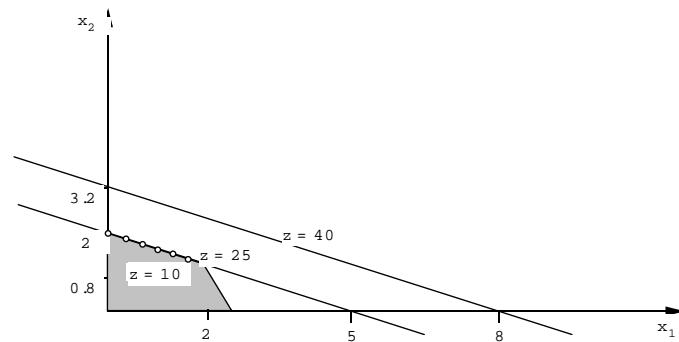


Figura 2.8: Caso di soluzione non unica.

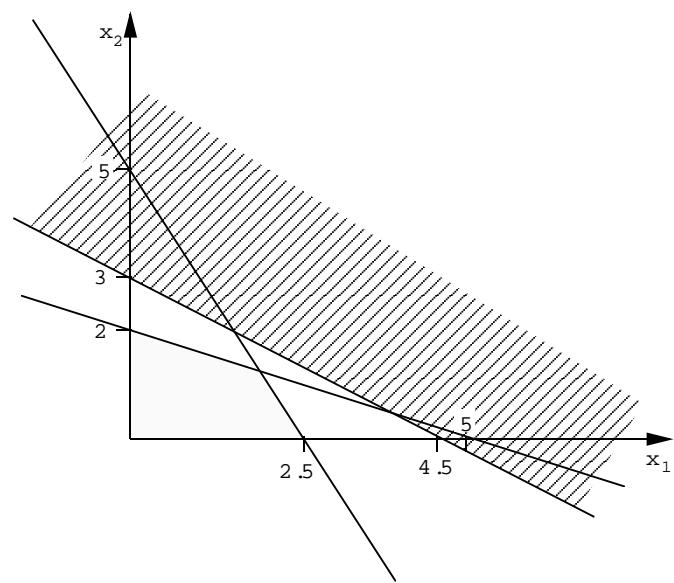


Figura 2.9: Regione ammissibile vuota.

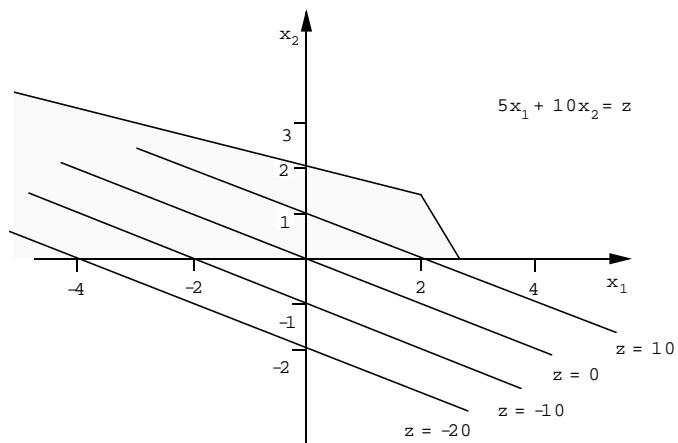


Figura 2.10: Regione ammissibile illimitata.

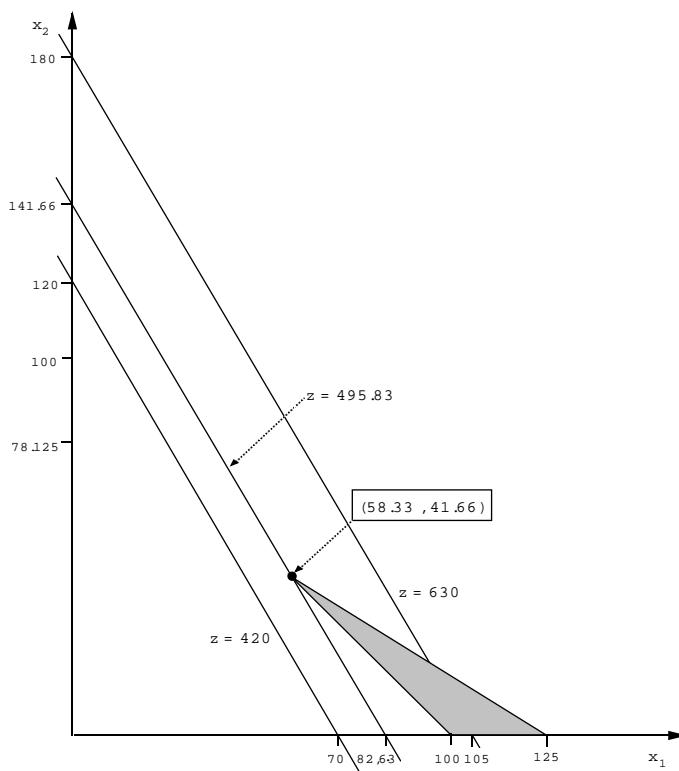


Figura 2.11: Soluzione grafica del problema di miscelazione.

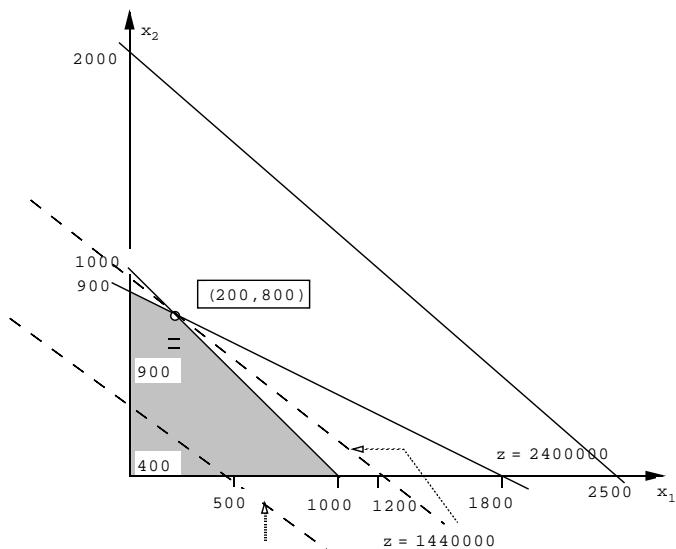


Figura 2.12: Soluzione grafica del problema di produzione.

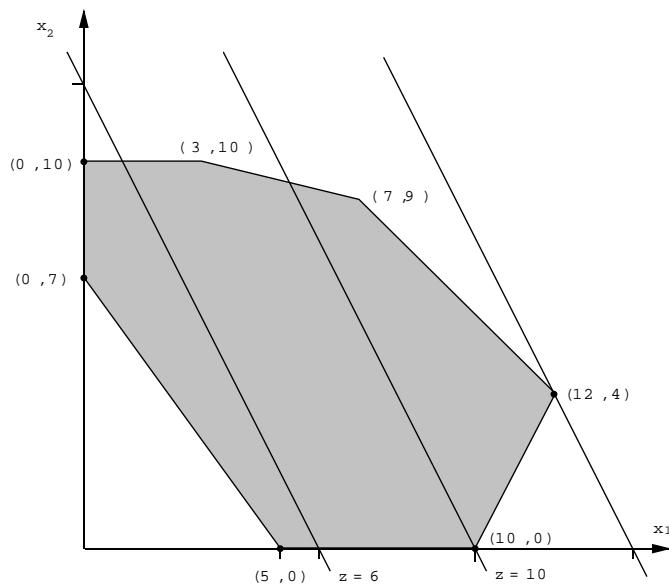


Figura 2.13: Soluzione grafica dell'esercizio 2.3.3.

2.4 Teorema fondamentale della Programmazione Lineare

Il nostro primo compito è quello di introdurre, in maniera elementare, alcune idee base di geometria degli insiemi convessi che ci permetteranno di meglio comprendere la struttura geometrica dei problemi di PL. Questi concetti rendono precise le nozioni di “vertice” e “insieme convesso” già incontrate nel paragrafo precedente e sono utilizzabili anche nel caso di problemi con più di due variabili.

Insiemi Convessi

Definizione 2.4.1 Siano x e y due punti in \mathbb{R}^n . L’insieme dei punti di \mathbb{R}^n ottenuti come

$$z = (1 - \beta)x + \beta y,$$

al variare di β nell’intervallo $[0, 1]$ viene definito come segmento di estremi x e y e viene sinteticamente indicato con $[x, y]$.

Esempio 2.4.2 Nella figura 2.14 è rappresentato il segmento in \mathbb{R}^2 avente per estremi i punti $x = (1, 1)^T$ e $y = (8, 5)^T$. Per $\beta = 0$ ritroviamo il punto x , mentre per $\beta = 1$ ritroviamo il punto y ; i punti segnati nella figura come x_a , x_b e x_c corrispondono rispettivamente a valori di β pari a 0.25, 0.5 e 0.75. \square

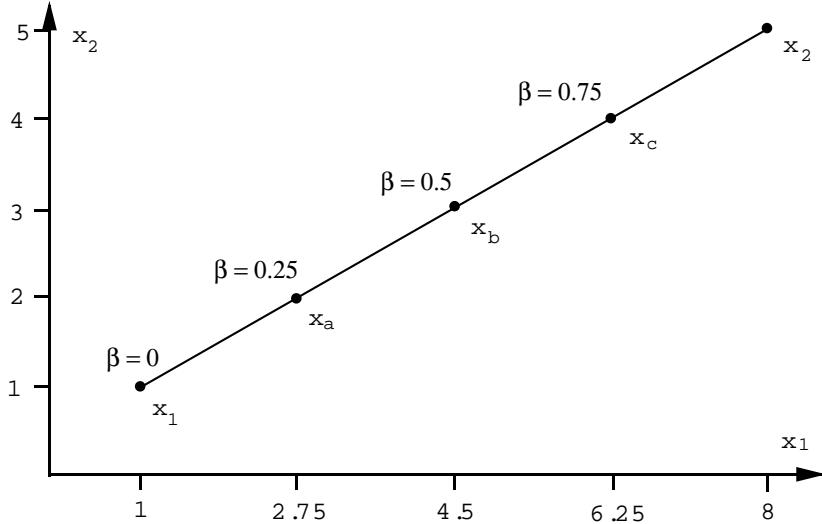


Figura 2.14: Esempio di segmento.

Dalla figura 2.14 risulta ovvio che il concetto di segmento è la generalizzazione, al caso di \mathbb{R}^n dell’usuale concetto di segmento valido nel piano.

Notiamo anche come, nel caso in cui gli estremi appartengano ad \mathbb{R}^1 , e sono quindi due numeri (scalar), diciamo a e b , il concetto di segmento di estremi a e b coincida con quello di intervallo $[a, b]$, fatto che giustifica la notazione $[x, y]$ impiegata per indicare il segmento.

Definizione 2.4.3 Un insieme $X \in \mathbb{R}^n$ è convesso se per ogni coppia di punti appartenenti all’insieme, appartiene all’insieme anche tutto il segmento che li congiunge. \square

Utilizzando il concetto di segmento chiuso, la definizione di insieme convesso può essere riformulata nel modo seguente: Un insieme X è convesso se per ogni coppia di vettori $x, y \in X$ si ha $[x, y] \subseteq X$.

Dalla definizione segue che l’insieme vuoto e l’insieme costituito da un solo vettore sono insiemi convessi (banali). Il più semplice insieme convesso non banale è il *segmento* di estremi $x, y \in \mathbb{R}^n$.

Esempio 2.4.4 In \mathbb{R}^2 gli insiemi (a), (b) della figura 2.15 sono convessi, mentre gli insiemi (c), (d) della stessa figura non lo sono. Infatti agli insiemi (c), (d) appartengono coppie di punti, quali quelle segnate nella figura, tali che il segmento che li congiunge presenta dei punti non appartenenti all'insieme; ciò non avviene invece comunque si prendano coppie di punti negli insiemi (a) e (b). \square

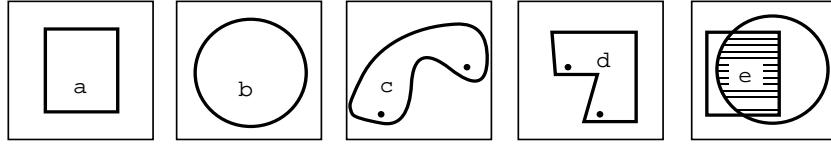


Figura 2.15: Insiemi convessi e non convessi.

Una importante proprietà degli insiemi convessi è espressa dal seguente teorema.

Teorema 2.4.5 *L'intersezione di due insiemi convessi è un insieme convesso.*

Dimostrazione: Siano $X_1, X_2 \subseteq \mathbb{R}^n$ due insiemi convessi e sia $X = X_1 \cap X_2$ la loro intersezione. Siano x ed y due vettori in X , allora $x, y \in X_1$ ed $x, y \in X_2$. Poiché X_1 ed X_2 sono insiemi convessi abbiamo che $[x, y] \subseteq X_1$ e che $[x, y] \subseteq X_2$. Ma allora $[x, y] \subseteq X$ e l'insieme X è convesso \square

Esempio 2.4.6 L'insieme (e) della figura 2.15 è dato dall'intersezione di due insiemi convessi ed è convesso \square

Dal Teorema (2.4.5) si può derivare, con un semplice ragionamento induttivo, il seguente corollario.

Corollario 2.4.7 *L'intersezione di un numero finito di insiemi convessi è un insieme convesso.* \square

Passiamo ora a considerare dei particolari insiemi convessi che rivestono un ruolo importante nella teoria della programmazione lineare.

Definizione 2.4.8 *Sia a un vettore di \mathbb{R}^n e b un numero reale. L'insieme*

$$H = \{x \in \mathbb{R}^n : a^T x = b\}$$

è detto iperpiano definito dall'equazione $a^T x = b$. Gli insiemi

$$\begin{aligned} S^{\leq} &= \{x \in \mathbb{R}^n : a^T x \leq b\} \\ S^{\geq} &= \{x \in \mathbb{R}^n : a^T x \geq b\} \end{aligned}$$

sono detti semispazi chiusi definiti dalle disequazioni $a^T x \leq b$ e $a^T x \geq b$. \square

Nel caso dello spazio \mathbb{R}^2 il concetto di iperpiano coincide con quello di retta, mentre nel caso dello spazio \mathbb{R}^3 il concetto di iperpiano coincide con quello di piano. In maniera intuitiva, i semispazi possono essere pensati come l'insieme dei punti che "giacciono" da una stessa parte rispetto all'iperpiano.

Esempio 2.4.9 Con riferimento alla figura 2.1, l'iperpiano (= retta) $10x_1 + 5x_2 = 25$ divide lo spazio (= piano) in due semispazi: $S^{\geq} = \{x \in \mathbb{R}^2 : 10x_1 + 5x_2 \geq 25\}$, indicato in grigio nella figura, e $S^{\leq} = \{x \in \mathbb{R}^2 : 10x_1 + 5x_2 \leq 25\}$, indicato in bianco nella figura. \square

Notiamo che l'iperpiano H fa parte di tutti e due i semispazi e che l'intersezione dei due semispazi coincide con l'iperpiano. In termini insiemistici abbiamo che

$$H \subset S^{\geq}, \quad H \subset S^{\leq}, \quad S^{\geq} \cap S^{\leq} = H.$$

I semispazi e gli iperpiani sono insiemi convessi.

Teorema 2.4.10 *Un semispazio chiuso è un insieme convesso.*

Dimostrazione: Dimostreremo il teorema per un semispazio $S^{\leq} = \{x \in \mathbb{R}^n : a^T x \leq b\}$, la dimostrazione per il semispazio S^{\geq} ottenuto invertendo il verso della disequazione è analoga. Consideriamo due generici vettori x ed y appartenenti ad S^{\leq} , vogliamo dimostrare che ogni vettore $z \in [x, y]$ appartiene ad S^{\leq} , ovvero soddisfa la relazione $a^T z \leq b$.

Sia $z = \beta x + (1 - \beta)y$ con $0 \leq \beta \leq 1$. Poiché x ed y appartengono ad S^{\leq} abbiamo che $a^T x \leq b$ e $a^T y \leq b$. Inoltre, poiché β ed $1 - \beta$ sono reali non negativi abbiamo che

$$a^T(\beta x + (1 - \beta)y) = \beta a^T x + (1 - \beta)a^T y \leq \beta b + (1 - \beta)b = b$$

e quindi che $a^T z \leq b$ □

Utilizzando il Teorema (2.4.10) e il Teorema (2.4.5) e ora facile dimostrare che anche un iperpiano è un insieme convesso.

Corollario 2.4.11 *Un iperpiano è un insieme convesso.*

Dimostrazione: Un iperpiano è l'intersezione di due semispazi chiusi (S^{\leq} e S^{\geq}). Per il Teorema (2.4.10) un semispazio chiuso è un insieme convesso mentre, per il Teorema (2.4.5), l'intersezione di due insiemi convessi è un insieme convesso. □

Notiamo ora che l'insieme ammissibile di un problema di PL è definito come l'insieme di punti che soddisfa i vincoli, cioè un insieme di equazioni e disequazioni lineari. Usando la terminologia appena introdotta, possiamo anche dire che l'insieme di punti ammissibili di un problema di PL è dato dall'intersezione di un numero finito di semispazi (disequazioni lineari) e iperpiani (equazioni lineari). Quindi, applicando il Teorema (2.4.10), il Corollario 2.4.11 e il Teorema (2.4.5) abbiamo il seguente risultato.

Teorema 2.4.12 *L'insieme ammissibile di un problema di programmazione lineare è un insieme convesso.* □

In particolare è usuale introdurre la seguente definizione:

Definizione 2.4.13 *Un insieme $P \subseteq \mathbb{R}^n$ è un poliedro se è l'intersezione di un numero finito di semispazi chiusi e iperpiani.* □

Usando tale definizione possiamo anche dire che l'insieme ammissibile di un problema di programmazione lineare è un poliedro (convesso). Notiamo che nel caso \mathbb{R}^3 possibili poliedri sono i cubi, i tetraedri etc. In generale è usuale denotare un poliedro con P . Usando questa terminologia, possiamo riformulare un generico problema di PL in forma di minimizzazione come:

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & x \in P. \end{aligned}$$

Vertici

In questa sezione formalizziamo il concetto intuitivo di *vertice*. Questo concetto riveste un ruolo fondamentale nella teoria della programmazione lineare.

Definizione 2.4.14 *Un vettore x appartenente ad un insieme convesso C è detto punto estremo di C se non esistono due punti distinti $x_1, x_2 \in C$ tali che $x \neq x_1, x \neq x_2$ ed $x \in [x_1, x_2]$*

Definizione 2.4.15 *Un punto appartenente a un poliedro è un vertice del poliedro stesso se e solo se è un suo punto estremo.* □

Quindi, nel caso dei poliedri, che saranno gli unici insiemi convessi che prenderemo in considerazione in questo corso, vertici e punti estremi coincidono.

Esempio 2.4.16 Nell'insieme di figura 2.16 il punto A non è un punto estremo, in quanto è interno al segmento che congiunge i punti B e C, anch'essi appartenenti all'insieme; lo stesso vale per il punto D, interno al segmento [E,F]. Sono invece punti estremi dell'insieme i punti E, F, G, H. Notare che l'insieme considerato è un poliedro solo se la congiungente di F ed H è intesa come una linea spezzata (numero finito di segmenti) approssimante l'arco di circonferenza. \square

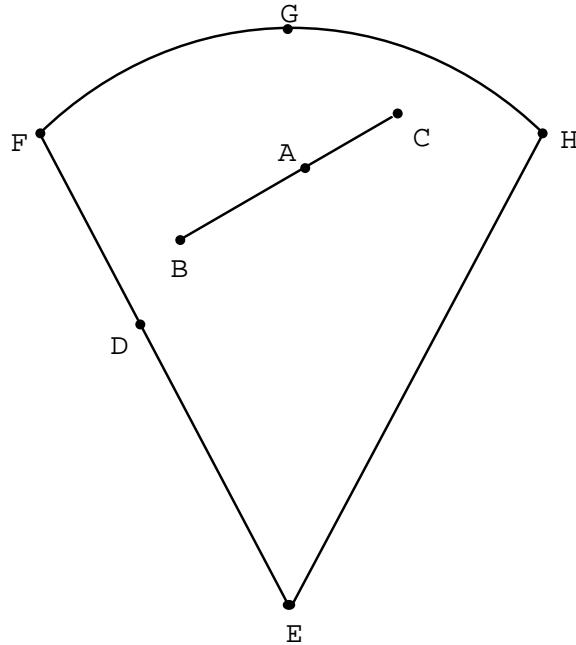


Figura 2.16: Vertici di un insieme.

Riprendendo la soluzione grafica presentata nel Paragrafo 2.3, notiamo che quando il problema di PL ammette soluzione, esiste un vertice del poliedro convesso ammissibile che è soluzione.

Tuttavia bisogna porre attenzione al fatto che esistono insiemi convessi e poliedrali *che non contengono vertici*. Un esempio è dato nella figura 2.17 data dalla parte di piano contenuta tra due rette parallele r_1 e r_2 . Se si considera un problema di PL che abbia la regione rappresentata in 2.17 come regione ammissibile, è facile convincersi che sono possibili due soli casi:

1. Le linee di livello della funzione obiettivo sono parallele a r_1 e r_2 , nel qual caso l'insieme dei punti di massimo o minimo coincidono con r_1 o r_2 ;
2. Le linee di livello della funzione obiettivo non sono parallele a r_1 e r_2 , nel qual caso il corrispondente problema di PL è illimitato sia superiormente che inferiormente.

La peculiarità di questo esempio nasce dal fatto che il poliedro rappresentato nella figura 2.17 non ha vertici e quindi, contrariamente a quanto visto nel paragrafo precedente, il problema di PL con regione ammissibile data da una “striscia” può avere soluzioni ottime, ma nessuna su un vertice. Vedremo fra poco che il caso in cui il poliedro che rappresenta la regione ammissibile di un problema di PL non ha vertici è l'unico caso in cui il problema di PL può avere soluzione ottima senza che nessuna soluzione coincida con un vertice. Risulta quindi interessante capire quando un poliedro può non ammettere vertice.

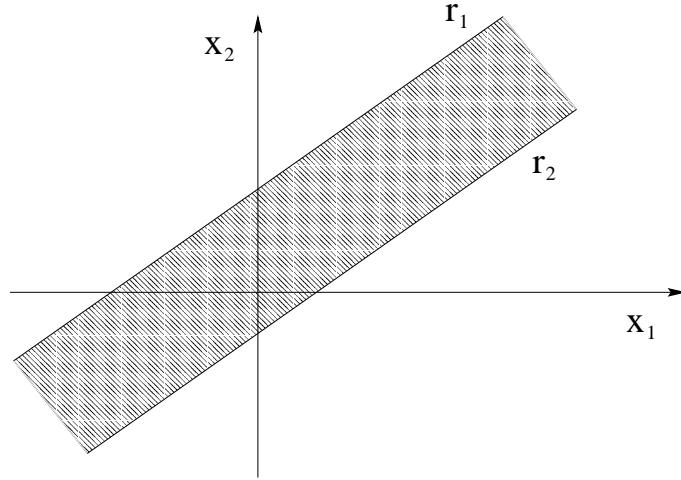


Figura 2.17: Poliedro senza vertici.

La caratterizzazione dei casi in cui un poliedro non ammette vertici è molto semplice ed è riportata nel seguente risultato, di cui omettiamo la prova.

Teorema 2.4.17 *Un poliedro P non ha vertici se e solo se contiene una retta.*

È evidente che il poliedro nella figura 2.17 contiene rette (in particolare contiene, per esempio, r_1, r_2) e quindi, non contiene vertici.

Nelle sezioni precedenti abbiamo visto che spesso in un problema di PL le variabili sono tutte non negative (tra i vincoli compare $x \geq 0$). Questo implica che la regione ammissibile di questi problemi di PL è interamente contenuta nel primo ortante e quindi *non può contenere rette*. Quindi, in base al teorema precedente, tutti i poliedri contenuti nel primo ortante o sono vuoti o hanno dei vertici. Notiamo che questa è sicuramente la classe di poliedri che più frequentemente si incontra nelle applicazioni.

A valle di questa discussione possiamo ora enunciare il cosiddetto *Teorema Fondamentale della Programmazione Lineare*, che riassume e precisa tutte le osservazioni e intuizioni maturate finora.

Teorema 2.4.18 *Sia dato un problema di PL. Allora una e una sola delle seguenti affermazioni è vera:*

1. *La regione ammissibile è vuota;*
2. *Il problema è illimitato;*
3. *Il problema ammette soluzioni ottime.*

Se il problema ammette soluzioni ottime e il poliedro che definisce la regione ammissibile ha dei vertici, allora almeno una soluzione ottima cade su un vertice.

È interessante approfondire un poco la natura dell'insieme delle soluzioni. Nel paragrafo precedente abbiamo notato che sembra essere vero che se un problema di PL ha più di una soluzione ottima, allora ne ammette infinite. Ci proponiamo qui di precisare questa affermazione.

Sia dato un poliedro $P \subseteq \mathbb{R}^n$ e un corrispondente problema di PL:

$$\min c^T x$$

$$x \in P.$$

Supponiamo che questo problema abbia (almeno) una soluzione ottima x^* . Indichiamo con $z^* = c^T x^*$ il *valore ottimo*, cioè il valore assunto dalla funzione obiettivo all'ottimo. È evidente che se \hat{x}^* è una

qualunque altra soluzione ottima, risulta $z^* = c^T \hat{x}^*$. Vice versa, se un punto x è ammissibile, cioè se $x \in P$ e risulta $c^T x = z^*$, allora x è una soluzione ottima per definizione. Riassumendo possiamo affermare che l'insieme delle soluzioni ottime del problema di PL dato è

$$P \cap \{x \in \mathbb{R}^n : c^T x = z^*\}.$$

Questo mostra immediatamente che l'insieme delle soluzioni ottime di un problema di PL è un poliedro contenuto in P , in quanto intersezione di P , definito da un insieme di equazioni e disequazioni lineari con l'iperpiano

$$\{x \in \mathbb{R}^n : c^T x = z^*\}.$$

In effetti è facilmente possibile dire qualcosa di ancora più preciso riguardo all'insieme di soluzioni ottime di un problema di PL, come riportato nel seguente teorema.

Teorema 2.4.19 *Sia dato un problema di PL*

$$\begin{aligned} \min \quad & c^T x \\ x \in & P. \end{aligned}$$

Denotiamo con SOL l'insieme di soluzioni ottime di questo problema. Allora

1. *SOL è un poliedro contenuto in P ;*
2. *Se v è un vertice di SOL , allora v è anche un vertice di P .*

Dimostrazione: Se SOL è vuoto le affermazioni sono ovvie. Supponiamo che SOL non sia vuoto, supponiamo cioè che il problema di PL abbia soluzioni ottime. La prima affermazione è stata dimostrata nelle considerazioni svolte prima del teorema. Passiamo quindi a dimostrare la seconda affermazione.

Sia v un vertice di SOL . Ovviamente v appartiene a P . Supponiamo per assurdo che v non sia un vertice di P . Allora esistono due vettori $x, y \in P$ ambedue differenti da v , tali che

$$v = \beta x + (1 - \beta)y \quad (2.1)$$

per qualche $\beta \in (0, 1)$. Indichiamo come al solito con z^* il valore ottimo del problema. Notiamo che, siccome x e y appartengono al poliedro ammissibile P , si deve avere

$$c^T x \geq z^* \quad \text{e} \quad c^T y \geq z^*.$$

Inoltre possiamo scrivere

$$z^* = c^T v = c^T [\beta x + (1 - \beta)y] = \beta c^T x + (1 - \beta)c^T y.$$

Da queste relazioni discende che deve essere $c^T x = z^*$, $c^T y = z^*$ e quindi $x, y \in SOL$. Ma siccome abbiamo supposto che x e y sono diversi da v abbiamo che la (2.1) contraddice il fatto che v sia un vertice di SOL . Ciò conclude la prova. \square

In conseguenza di quanto appena visto possiamo osservare che se un problema ha due soluzioni ottime distinte, x^* e y^* , allora sicuramente ne contiene infinite. Infatti, poiché SOL è convesso, esso contiene sicuramente almeno tutto il segmento $[x^*, y^*]$, composto da infiniti punti.

2.5 Cenni sul Metodo del Simplex

Dato un problema di PL è ovviamente necessario, se il modello fatto deve essere di qualche utilità, essere capaci di “riolverlo”. Nel caso della programmazione lineare si dice che un algoritmo *risolve* un problema di PL se esso è capace di determinare correttamente se il problema dato è vuoto oppure illimitato oppure,

se nessuno di queste due casi risulta verificato, sia capace di individuare una soluzione ottima. Esistono molti algoritmi per la risoluzione di problemi di PL, spesso implementati in pacchetti software molto diffusi e utilizzati. In questo paragrafo ci limitiamo ad esporre alcune considerazioni molto generali sul *Metodo del Simplex*, che verrà approfondito in altri corsi. Come già accennato, il Metodo del Simplex è stato il primo algoritmo pratico per la risoluzione di problemi di PL ed è tuttora il più usato e uno dei più efficienti in pratica.

Per capire il modo di operare del Metodo del Simplex iniziamo con l'osservare che si può dimostrare il fatto, per altro abbastanza intuitivo, che *un poliedro ha sempre un numero finito di vertici*. Eventualmente i vertici, come abbiamo già avuto modo di osservare, possono essere zero. Comunque, benché finito, il numero di vertici di un poliedro può essere arbitrariamente alto. Basta pensare all'area racchiusa da un poligono nel piano. All'aumentare del numero dei lati cresce il numero di vertici. Per fare un altro esempio, consideriamo il poliedro

$$I = \{x \in \mathbb{R}^n : 0 \leq x_i \leq 1, i = 1, 2, \dots, n\}.$$

Se $n = 1$ abbiamo ovviamente un segmento, con 2 vertici ($2 = 2^1$). Se $n = 2$ abbiamo un quadrato, con 4 vertici ($4 = 2^2$). Se $n = 3$ abbiamo un cubo con 8 vertici ($8 = 2^3$). In generale, per $n > 3$ l'insieme I è noto come ipercubo di dimensione n e ha un numero di vertici pari a 2^n , che quindi cresce *esponenzialmente* con la dimensione dello spazio.

Supponiamo di avere un problema di PL

$$\begin{aligned} \min \quad & c^T x \\ x \in & P, \end{aligned}$$

dove P è un poliedro che non contiene rette, e di sapere che il problema ammette (almeno) una soluzione ottima. Allora, il Teorema Fondamentale della Programmazione Lineare ci autorizza a limitare la ricerca di una soluzione ottima ai vertici del poliedro P . Una procedure teorica per la ricerca dell'ottimo, potrebbe essere quindi

1. Calcola tutti i vertici v_1, v_2, \dots, v_q del poliedro P .
2. Valuta la funzione obiettivo in tutti i vertici e denota con v^* il vertice per cui si raggiunge il valore minimo:

$$c^T v^* \leq c^T v_i \quad i = 1, 2, \dots, q.$$

3. v^* è una soluzione ottima del problema.

Ovviamente questa strategia deve essere completata da procedure in grado di determinare se P è vuoto o se il problema è illimitato. Inoltre bisogna essere in grado di calcolare i vertici. Ma "in nuce" questa è una procedura teoricamente valida. Il problema principale di questo modo di procedere è che siccome il numero di vertici può, ed in generale è, altissimo, essa può diventare computazionalmente così oneroso da risultare impraticabile. Il Metodo del Simplex, nella sostanza, è un modo un po' più raffinato di realizzare lo schema precedente e può, molto grossolanamente, essere schematizzato come segue.

Passo 0: *Determina un vertice iniziale v del poliedro P .*

Passo 1: *Decidi se v è una soluzione ottima.*

Passo 2: *Se v non è una soluzione ottima allora determina "in modo intelligente" un nuovo vertice v di P e torna al Passo 1.*

Il cuore del metodo è racchiuso nella frase “in modo intelligente”. Questo modo intelligente di determinare un nuovo vertice, una volta stabilito che quello corrente non è ottimo, deve essere tale da garantire che, almeno nella maggioranza dei casi pratici, il metodo non debba visitare che una “piccola” frazione dei vertici di P prima di trovare un vertice ottimo. È inoltre necessario integrare questo schema con procedure in grado di determinare se il poliedro è vuoto oppure se il problema è illimitato inferiormente. Deve essere inoltre specificato un criterio di ottimalità al Passo 1 che permetta all’algoritmo di arrestarsi.

Il Metodo del Simplex è di fondamentale importanza sia dal punto di vista storico, sia dal punto di vista dell’utilizzo pratico. Infatti, introdotto da G. Dantzig nel 1947, ha aperto la strada allo sviluppo e alla diffusione della Programmazione Lineare e tutt’oggi è ampiamente utilizzato attraverso implementazioni sofisticate nella maggior parte dei package commerciali più diffusi e più efficienti.

Il Metodo del Simplex sfrutta la possibilità di poter ricondurre un qualsiasi problema di Programmazione Lineare nella forma

$$\begin{cases} \min c^T x \\ Ax = b \\ x \geq 0 \end{cases}$$

che viene chiamata *forma standard*.

È sempre possibile trasformare un problema di Programmazione Lineare dalla forma generale alla forma standard, eventualmente introducendo nuove variabili.

La struttura particolare dell’insieme ammissibile di un problema di Programmazione Lineare in forma standard può essere sfruttata per identificare in maniera più semplice i vertici. Tale possibilità viene sfruttata dal metodo del simplex per determinare efficientemente, ad ogni iterazione, un nuovo vertice. Inoltre tale metodo seleziona in maniera accurata i vertici che visita. Infatti, nelle varie iterazioni vengono scelti solamente i vertici in cui si ottiene una significativa decrescita della funzione obiettivo mentre vengono trascurati gli altri, generando così una successione di valori della funzione obiettivo strettamente decrescente e questo implica che una volta esaminato un vertice, il metodo non può tornarvi. Perciò in un numero finito di passi viene raggiunto un vertice che è ottimo oppure viene selezionato uno “spigolo” del poliedro descritto dall’insieme ammissibile lungo il quale la funzione obiettivo è illimitata inferiormente.

Concludiamo ripetendo che questi pochi cenni non vogliono servire ad altro che a dare una sia pur vaga idea del Metodo del Simplex, che è attualmente, nelle implementazioni commerciali, un algoritmo raffinatissimo e molto complesso, capace di risolvere problemi con decine e centinaia di migliaia di variabili.

2.6 Modellazione dei Problemi di Programmazione Lineare

Le applicazioni della Ricerca Operativa che possono essere formulate mediante l’uso di modelli di programmazione lineare sono molto frequenti e importanti. Lo scopo di questa parte è quello di illustrare alcune classi di problemi di programmazione lineare tipici che si incontrano frequentemente nelle applicazioni reali.

Consigliamo allo studente, al fine di impraticarsi nella formulazione di problemi di PL, di provare da solo a formulare i problemi descritti prima di leggere la formulazione fornita. Ricordiamo che un modello di Programmazione Lineare è costituito da:

- una *singola espressione lineare* (la funzione obiettivo) che deve essere minimizzata o massimizzata.
- un *numero finito di vincoli lineari*, vincoli cioè espressi nella forma

$$f(x_1, \dots, x_n) \leq b$$

o

$$f(x_1, \dots, x_n) \geq b$$

dove f è una funzione lineare e b una costante reale, (in questo caso si parla di vincoli di disuaglianza) oppure

$$f(x_1, \dots, x_n) = b$$

sempre con f funzione lineare e b costante reale; (in questo caso si parla di vincoli di uguaglianza).

In riferimento alle applicazioni di tipo economico la funzione obiettivo ha di solito il significato di profitto (da massimizzare) oppure di costo (da minimizzare). Profitti e costi sono ottenuti come somma dei *profitti e costi marginali* cioè di quelli relativi a ciascuna unità di prodotto.

Quando è richiesta la massimizzazione di un profitto, il modello contiene, di solito, vincoli che esprimono limitazioni superiori sulle risorse (vincoli di capacità produttiva, disponibilità di materie prime); se invece è richiesta la minimizzazione di un costo sono di solito presenti vincoli sulla domanda (richieste di mercato) che impongono limitazioni inferiori alle variabili. È possibile la presenza di *vincoli di continuità* che esprimono conservazione o riduzione di masse o volumi ed hanno spesso la forma di vincoli di uguaglianza.

Le ipotesi che vengono assunte nel formulare un problema di Programmazione Lineare sono le seguenti:

- *proporzionalità*: il contributo di una variabile di decisione alla funzione obiettivo e ai vincoli è proporzionale secondo una costante moltiplicativa alla quantità rappresentata dalla variabile stessa;
- *additività*: il contributo delle variabili di decisione alla funzione obiettivo e ai vincoli è dato dalla somma dei contributi di ogni singola variabile.
- *continuità*: ogni variabile di decisione può assumere tutti i valori reali nell'intervallo di ammissibilità, e quindi le variabili possono assumere valori frazionari.

La particolare attenzione dedicata ai modelli di Programmazione Lineare deriva dai numerosi vantaggi che essa presenta e che possono essere così sintetizzati:

1. *Generalità e flessibilità*.

I modelli di Programmazione Lineare possono descrivere moltissime situazioni reali anche assai diverse tra loro e quindi hanno un carattere di universalità e di adattabilità alle diverse realtà applicative e anche quando l'ipotesi di linearità non è accettabile, il modello lineare costituisce una buona base di partenza per successive generalizzazioni.

2. *Semplicità*.

I modelli di Programmazione Lineare sono espressi attraverso il linguaggio dell'algebra lineare e quindi sono facilmente comprensibili anche in assenza di conoscenze matematiche più elevate.

3. *Efficienza degli algoritmi risolutivi*.

Come accennato in precedenza i modelli reali hanno dimensioni molto elevate ed è quindi indispensabile l'uso del calcolatore che con opportuni programmi di calcolo possa rapidamente fornire una soluzione numerica. Relativamente ai modelli di Programmazione Lineare esistono programmi molto efficienti e largamente diffusi che sono in grado di risolvere rapidamente problemi con migliaia di vincoli e centinaia di variabili.

4. *Possibilità di analisi qualitative*.

I modelli di Programmazione Lineare permettono di ottenere, oltre la soluzione numerica del problema, anche ulteriori informazioni relative alla dipendenza della soluzione da eventuali parametri presenti, che possono avere significative interpretazioni economiche.

2.7 Esempi tipici di PL

Esamineremo nel seguito tre grandi classi di modelli di Programmazione Lineare che rappresentano situazioni molto diffuse del mondo reale e che abbiamo già incontrato nel paragrafo 2.2; si tratta dei

- *modelli di allocazione ottima di risorse,*
- *modelli di miscelazione,*
- *modelli di trasporto.*

I modelli reali sono ovviamente molto più grandi in termini di dimensioni (numero delle variabili e dei vincoli) e più complessi di quelli analizzati qui, noi, per ovvie ragioni didattiche ci limiteremo alla presentazione di esempi “piccoli”, che tuttavia colgono le caratteristiche essenziali dei modelli usati in pratica.

Va inoltre sottolineato come nella realtà i problemi che si incontrano spesso possono combinare insieme più classi di problemi; la divisione che si è effettuata in problemi di allocazione ottima, problemi di miscelazione e problemi di trasporti ha, evidentemente, scopi essenzialmente didattici e dovrebbe fornire la possibilità di affrontare anche situazioni in cui confluiscano simultaneamente problematiche riconducibili a differenti classi di modelli.

Infine segnaliamo che esistono e vengono comunemente incontrati casi pratici il cui modello di PL non rientra in nessuna delle classi sopra indicate o in una loro combinazione. Quindi, ripetiamo, lo scopo di questo capitolo è solo di presentare, in forma didattica, alcuni dei più comuni tipi di problemi formulabili come modelli di PL.

2.7.1 Modelli di allocazione ottima di risorse

Si tratta di problemi in cui disponendo di determinate *risorse* (disponibilità di macchinari, materie prime, mano d'opera, energia, etc.), si deve pianificare la produzione di un certo numero di prodotti finiti, cercando di massimizzare il profitto ricavato dalla loro vendita.

Esempi introduttivi

Esempio 2.7.1 *Un colorificio produce due tipi di coloranti **C1** e **C2** utilizzando 3 preparati base **P1**, **P2**, **P3**. Tali preparati devono essere acquistati e subiscono un processo di raffinazione prima di essere utilizzati. Le quantità (in litri) di preparati base da acquistare per produrre un litro di colorante di ciascuno dei due tipi è riportato nella seguente tabella*

	C1	C2
P1	1	1
P2	1	2
P3	-	1

Ogni mese la quantità di preparati base (in litri) che possono essere acquistati è la seguente

P1	P2	P3
750	1000	400

*Il prezzo di vendita del colorante **C1** è di 7000 lire al litro, mentre il colorante **C2** viene venduto a 10000 lire al litro. Determinare la strategia ottimale di produzione mensile in modo da massimizzare i ricavi ottenuti dalla vendita dei due coloranti.*

Formulazione.

Si vuole costruire il modello di Programmazione Lineare che rappresenti il problema in analisi considerando le limitazioni date dalle produzioni effettivamente realizzabili.

È immediato associare le variabili di decisione ai quantitativi di coloranti prodotti. Siano, quindi, rispettivamente x_1 e x_2 i quantitativi (in litri) da produrre mensilmente dei due coloranti.

Nel formulare il modello di Programmazione Lineare si deve verificare che siano soddisfatte le ipotesi fondamentali:

- *Proporzionalità.*

I consumi dei preparati base e i ricavi ottenibili sono proporzionali ai quantitativi di coloranti prodotti. Ad esempio, per produrre una quantità x_2 di colorante **C2** si consumano $2x_2$ litri di **P2** e dalla vendita di x_2 litri di **C2** si ricavano $10x_2$ migliaia di lire indipendentemente dalla quantità prodotta e venduta dell'altro tipo di colorante.

- *Additività.*

I consumi dei preparati base e i ricavi rispettivamente associati alla produzione dei due coloranti sono additivi, nel senso che per produrre x_1 litri di colorante **C1** e x_2 di **C2** si consumano $x_1 + 2x_2$ litri di preparato di base **P2** e si ricavano $7x_1 + 10x_2$ migliaia di lire.

- *Continuità.*

Ogni variabile introdotta nel modello può assumere tutti i valori reali nell'intervallo di ammissibilità.

Queste ipotesi devono valere in generale e quindi dovranno essere sempre verificate nella costruzione di un modello di (PL).

– *Variabili.* Come già detto, prendiamo come variabili di decisione x_1 e x_2 , rispettivamente i quantitativi (in litri) di colorante **C1** e **C2** da produrre mensilmente.

– *Funzione obiettivo.* È rappresentata dal profitto totale che per le ipotesi fatte è dato (in migliaia di lire) da $7x_1 + 10x_2$.

– *Vincoli.* Poiché il consumo di preparati base non può essere superiore alla disponibilità si deve avere

$$\begin{aligned} x_1 + x_2 &\leq 750 \\ x_1 + 2x_2 &\leq 1000 \\ x_2 &\leq 400. \end{aligned}$$

Inoltre si deve esplicitare il vincolo di non negatività sulle variabili

$$x_1 \geq 0, x_2 \geq 0.$$

Quindi la formulazione finale è

$$\begin{cases} \max (7x_1 + 10x_2) \\ x_1 + x_2 \leq 750 \\ x_1 + 2x_2 \leq 1000 \\ x_2 \leq 400 \\ x_1 \geq 0, x_2 \geq 0. \end{cases}$$

□

Esempio 2.7.2 *Un'industria manifatturiera può fabbricare 5 tipi di prodotti che indichiamo genericamente con **P1**, **P2**, **P3**, **P4**, **P5** usando 2 processi di produzione che avvengono mediante l'uso di due macchine che indichiamo con **M1** e **M2**. Dopo aver dedotto il costo del materiale grezzo, ciascuna unità di prodotto dà i seguenti profitti (in migliaia di lire)*

P1	P2	P3	P4	P5
250	300	500	450	180

Ciascuna unità di prodotto richiede un certo tempo di ciascuno dei 2 processi; la tabella che segue riporta i tempi (in ore) di lavorazione di ciascuna macchina per ottenere una unità di ciascuno dei prodotti finiti

	P1	P2	P3	P4	P5
M1	10	15	7	18	—
M2	9	13	—	—	20

Inoltre, l'assemblaggio finale per ciascuna unità di ciascun prodotto richiede 18 ore di lavoro di un operaio. La fabbrica possiede 4 macchine **M1** e 3 macchine **M2** che sono in funzione 5 giorni alla settimana per 2 turni di 8 ore al giorno. Gli operai impiegati nell'assemblaggio sono 10 e ciascuno di essi lavora 5 giorni alla settimana per un turno di 8 ore al giorno. Trovare la quantità che conviene produrre di ciascun prodotto per massimizzare il profitto totale.

Formulazione.

Costruiamo un modello di Programmazione Matematica rappresentante il problema in analisi supponendo di voler pianificare la produzione settimanale. È immediato verificare che anche in questo caso le ipotesi fondamentali della Programmazione Lineare sono soddisfatte.

- *Variabili di decisione.* È naturale introdurre le variabili reali x_1, x_2, x_3, x_4, x_5 rappresentanti rispettivamente le quantità di prodotto **P1**, **P2**, **P3**, **P4**, **P5** che conviene fabbricare in una settimana.
- *Funzione Obiettivo.* Ciascuna unità di prodotto finito contribuisce al profitto totale secondo la tabella data. Quindi il profitto totale sarà

$$250x_1 + 300x_2 + 500x_3 + 450x_4 + 180x_5. \quad (2.2)$$

L'obiettivo della fabbrica sarà quello di scegliere le variabili x_1, x_2, x_3, x_4, x_5 in modo che l'espressione (2.2) del profitto sia massimizzata. La (2.2) rappresenta la funzione obiettivo.

- *Vincoli.* Ovviamente la capacità produttiva della fabbrica sia dal punto di vista delle macchine, sia dal punto di vista degli operai, limita i valori che possono assumere le variabili x_j , $j = 1, \dots, 5$. Si hanno solo 4 macchine **M1** che lavorano per un totale di 320 ore settimanali e poiché ciascuna unità di prodotto **P1** usa per 10 ore la macchina **M1**, ciascuna unità di prodotto **P2** la usa per 15 ore e così via per gli altri prodotti, si dovrà avere

$$10x_1 + 15x_2 + 7x_3 + 18x_4 \leq 320. \quad (2.3)$$

Ragionando in modo analogo per la macchina **M2** si ottiene

$$9x_1 + 13x_2 + 20x_5 \leq 240. \quad (2.4)$$

Inoltre si hanno solo 10 uomini per l'assemblaggio, ciascuno dei quali lavora 40 ore a settimana e quindi si ha una capacità lavorativa settimanale di 400 ore. Poiché ciascuna unità di prodotto prevede 18 ore di lavoro di assemblaggio si dovrà avere

$$18x_1 + 18x_2 + 18x_3 + 18x_4 + 18x_5 \leq 400. \quad (2.5)$$

Le espressioni (2.3), (2.4) e (2.5) costituiscono i vincoli del modello. Ci sono inoltre vincoli impliciti dovuti al fatto che le variabili x_j , $j = 1, \dots, 5$ rappresentando quantità di prodotto non possono essere negative. Questa limitazione va esplicitata e quindi vanno aggiunti i vincoli

$$x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0, x_5 \geq 0.$$

La formulazione finale sarà quindi

$$\begin{cases} \max (250x_1 + 300x_2 + 500x_3 + 450x_4 + 180x_5) \\ 10x_1 + 15x_2 + 7x_3 + 18x_4 \leq 320 \\ 9x_1 + 13x_2 + 20x_5 \leq 240 \\ 18x_1 + 18x_2 + 18x_3 + 18x_4 + 18x_5 \leq 400 \\ x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0, x_5 \geq 0. \end{cases}$$

Questa formulazione è un problema matematico ben definito e costituisce il modello di Programmazione Matematica rappresentante il problema di pianificazione della produzione industriale in analisi. \square

Formulazione generale di un problema di allocazione ottima di risorse

Per costruire uno schema generale di formulazione per questo tipo di problemi si assuma di disporre di m risorse $\mathbf{R}_1, \mathbf{R}_2, \dots, \mathbf{R}_m$ e di voler fabbricare n diversi prodotti $\mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_n$.

Le risorse possono essere sia umane (mano d'opera) sia materiali (disponibilità di macchinari o di materie prime). Il problema della pianificazione delle risorse consiste nel determinare le quantità da fabbricare di ciascun prodotto $\mathbf{P}_1, \dots, \mathbf{P}_n$ in modo da massimizzare il profitto rispettando i vincoli sulle risorse disponibili o sui livelli di produzione richiesti.

Si indichi con a_{ij} , $i = 1, \dots, m$, $j = 1, \dots, n$ la quantità della risorsa \mathbf{R}_i necessaria per fabbricare una unità del prodotto \mathbf{P}_j . Si può così costruire la seguente tabella

	\mathbf{P}_1	\dots	\mathbf{P}_j	\dots	\mathbf{P}_n
\mathbf{R}_1	a_{11}	\dots	a_{1j}	\dots	a_{1n}
\vdots	\vdots		\vdots		\vdots
\mathbf{R}_i	a_{i1}	\dots	a_{ij}	\dots	a_{in}
\vdots	\vdots		\vdots		\vdots
\mathbf{R}_m	a_{m1}	\dots	a_{mj}	\dots	a_{mn}

Supponiamo che ciascuna risorsa \mathbf{R}_i non possa superare un valore prefissato b_i , $i = 1, \dots, m$

$$\begin{array}{cccc} \mathbf{R}_1 & \mathbf{R}_2 & \dots & \mathbf{R}_m \\ b_1 & b_2 & \dots & b_m \end{array}$$

e che nella vendita di ciascuna unità di prodotto \mathbf{P}_j si ricavi un profitto netto c_j , $j = 1, \dots, n$

$$\begin{array}{cccc} \mathbf{P}_1 & \mathbf{P}_2 & \dots & \mathbf{P}_n \\ c_1 & c_2 & \dots & c_n. \end{array}$$

Formulazione.

È utile ribadire le ipotesi già esposte in precedenza le quali devono valere in generale per la costruzione di modelli di Programmazione Lineare: *proporzionalità*, *additività*, *continuità* cioè i consumi delle risorse e i ricavi ottenibili sono proporzionali ai quantitativi di prodotto fabbricati; i consumi globali di risorse e i ricavi totali si ottengono come somma dei consumi e dei ricavi marginali; le variabili possono assumere valori frazionari.

– *Variabili di decisione.* Si introducono le variabili di decisione x_1, x_2, \dots, x_n rappresentanti (in un'opportuna unità di misura) la quantità di ciascun prodotto $\mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_n$. Queste saranno le incognite del problema. Tali variabili di decisione sono i cosiddetti *livelli di attività*. Introducendo come spazio delle variabili lo spazio delle n -uple reali \mathbb{R}^n si può considerare un $x \in \mathbb{R}^n$ definendo $x = (x_1, \dots, x_n)^T$.

– *Funzione obiettivo.* Per le ipotesi fatte la funzione obiettivo (da massimizzare) può essere scritta

$$z = c_1x_1 + \dots + c_nx_n = \sum_{j=1}^n c_jx_j.$$

Introducendo $c \in \mathbb{R}^n$, definito $c = (c_1, \dots, c_n)^T$ la funzione obiettivo può essere scritta in notazione vettoriale

$$z = c^T x.$$

– *Vincoli*. Si devono introdurre i seguenti vincoli:

- Vincoli di capacità produttiva:

tenendo conto delle limitazioni delle risorse si hanno i seguenti m vincoli

$$\begin{array}{llll} a_{11}x_1 + & \dots & +a_{1n}x_n & \leq b_1 \\ a_{21}x_1 + & \dots & +a_{2n}x_n & \leq b_2 \\ \vdots & \dots & \vdots & \vdots \\ a_{m1}x_1 + & \dots & +a_{mn}x_n & \leq b_m. \end{array}$$

- Vincoli di non negatività:

le variabili devono essere non negative in quanto esse rappresentano livelli di produzione e quindi si hanno i vincoli

$$x_i \geq 0, \quad i = 1, \dots, n.$$

Introducendo la matrice $(m \times n)$

$$A = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \dots & a_{mn} \end{pmatrix}$$

e il vettore $b = (b_1, \dots, b_m)^T$ la formulazione completa del problema può essere scritta nella forma

$$\begin{cases} \max c^T x \\ Ax \leq b \\ x \geq 0. \end{cases}$$

È una formulazione generale (con solo vincoli di disegualanza e vincoli di non negatività) in cui si può porre un generico problema di allocazione ottima di risorse.

Nella pratica, potrebbe essere necessario imporre ulteriori vincoli:

- Vincoli di domanda

- limitazioni inferiori sulle variabili x_i cioè

$$x_i \geq l_i \quad i = 1, \dots, n$$

con $l_i \geq 0$ per assicurare che i prodotti siano fabbricati in quantità significative. In questo caso, per ogni indice i per il quale $l_i > 0$ il vincolo di non negatività $x_i \geq 0$ è ridondante.

- limitazioni superiori sulle variabili, cioè

$$x_i \leq u_i \quad i = 1, \dots, n$$

dovute ad eventuali possibilità limitate di assorbimento dei prodotti da parte del mercato.

Introducendo le notazioni vettoriali $l = (l_1, \dots, l_n)^T$ e $u = (u_1, \dots, u_n)^T$ questi vincoli possono essere scritti nella forma $l \leq x \leq u$, $x \in \mathbb{R}^n$.

- Vincoli di interezza.

Se inoltre non ha senso considerare i prodotti quantità divisibili allora si deve definire un modello di programmazione a numeri interi. Cioè nel caso in cui non si possa supporre che i livelli di attività siano frazionari (ad es. se i prodotti sono quantità indivisibili come motori, lavatrici etc.), allora si deve aggiungere il vincolo che le quantità x_i siano intere.

Altri esempi di modelli di allocazione ottima di risorse

Verranno ora di seguito esaminati alcuni esempi di modelli di allocazione ottima di risorse.

Esempio 2.7.3 Una azienda automobilistica produce tre diversi modelli di autovettura: un modello economico, uno normale ed uno di lusso. Ogni autovettura viene lavorata da tre robot: **A**, **B** e **C**. I tempi necessari alla lavorazione sono riportati, in minuti, nella tabella seguente insieme al profitto netto realizzato per autovettura

	Economica	Normale	Lusso
A	20	30	62
B	31	42	51
C	16	81	10
Prezzo	1000	1500	2200

I robot **A** e **B** sono disponibili per 8 ore al giorno mentre il robot **C** è disponibile per 5 ore al giorno. Il numero di autovetture di lusso prodotte non deve superare il 20% del totale mentre il numero di autovetture economiche deve costituire almeno il 40% della produzione complessiva. Supponendo che tutte le autovetture vengano vendute, formulare un problema di Programmazione Lineare che permetta di decidere le quantità giornaliere (non necessariamente intere) da produrre per ciascun modello in modo tale da massimizzare i profitti rispettando i vincoli di produzione.

Formulazione.

È un problema di allocazione ottima di risorse e può essere formulato in termini di Programmazione Lineare nel seguente modo.

- *Variabili.* Indichiamo con x_1, x_2, x_3 , rispettivamente il numero di autovetture del modello economico, normale e di lusso.
- *Funzione obiettivo.* È data dal profitto globale e quindi è data da

$$1000x_1 + 1500x_2 + 2200x_3.$$

- *Vincoli.* I vincoli sulla capacità produttiva sono

$$20x_1 + 30x_2 + 62x_3 \leq 480$$

$$31x_1 + 42x_2 + 51x_3 \leq 480$$

$$16x_1 + 81x_2 + 10x_3 \leq 300.$$

I vincoli sul numero totale dei singoli tipi di autovetture sono

$$x_3 \leq 0.2(x_1 + x_2 + x_3)$$

$$x_1 \geq 0.4(x_1 + x_2 + x_3).$$

Si devono inoltre esplicitare i vincoli di non negatività

$$x_1 \geq 0 \quad x_2 \geq 0 \quad x_3 \geq 0.$$

Quindi la formulazione completa può essere scritta

$$\begin{cases} \max (1000x_1 + 1500x_2 + 2200x_3) \\ 20x_1 + 30x_2 + 62x_3 \leq 480 \\ 31x_1 + 42x_2 + 51x_3 \leq 480 \\ 16x_1 + 81x_2 + 10x_3 \leq 300 \\ x_3 \leq 0.2(x_1 + x_2 + x_3) \\ x_1 \geq 0.4(x_1 + x_2 + x_3) \\ x_1 \geq 0 \quad x_2 \geq 0 \quad x_3 \geq 0. \end{cases}$$

□

Esempio 2.7.4 Un'industria produce 4 tipi di elettrodomestici **E1**, **E2**, **E3**, **E4** ed è divisa in 3 reparti. Ciascun reparto può fabbricare ciascuno tipo di elettrodomestico. Questa industria dispone di 100 operai così ripartiti: 40 nel reparto 1, 35 nel reparto 2 e 25 nel reparto 3. Ciascun operaio lavora 5 giorni la settimana, per 8 ore al giorno. La tabella che segue riporta, per ciascun tipo di elettrodomestico e per ciascun reparto, il tempo di lavorazione (in ore) necessario per ottenere un elettrodomestico pronto per la vendita, insieme al prezzo di vendita unitario in migliaia di lire.

	E1	E2	E3	E4
Reparto 1	1	1.5	0.5	1.6
Reparto 2	1.2	1.3	0.6	1
Reparto 3	0.8	1.7	0.7	1.3
prezzo di vendita	800	1200	950	1100

Questa industria deve pianificare la sua produzione settimanale, deve cioè determinare il numero di ciascuno degli elettrodomestici che deve essere fabbricato da ciascun reparto in modo da soddisfare un ordine di almeno 1000, 600, 300, 200 elettrodomestici rispettivamente del tipo **E1**, **E2**, **E3**, **E4** e in modo da massimizzare il profitto complessivo ricavato dalla vendita.

Formulazione.

È un problema di allocazione ottima di risorse con la differenza rispetto a quelli fino ad ora esaminati, che ciascun reparto è in grado di fornire prodotti finiti pronti per la vendita, cioè non è necessaria la lavorazione su tutti i reparti per ottenere un prodotto finito. Questa differenza è di fondamentale importanza nella scelta delle variabili di decisione.

– *Variabili.* Si devono distinguere il numero di elettrodomestici prodotti in ciascun reparto e quindi una naturale associazione delle variabili di decisione è la seguente: si indica con x_{ij} , $i = 1, \dots, 4$, $j = 1, 2, 3$, il numero di elettrodomestici del tipo **Ei** da produrre settimanalmente nel reparto j -esimo.

– *Funzione obiettivo.* Sarà data dal profitto complessivo ricavato dalla vendita e quindi è

$$800(x_{11} + x_{12} + x_{13}) + 1200(x_{21} + x_{22} + x_{23}) + 950(x_{31} + x_{32} + x_{33}) + 1100(x_{41} + x_{42} + x_{43})$$

– *Vincoli.* Si devono considerare i vincoli dovuti alla limitata disponibilità settimanale di ore lavorative; in particolare, vista la distribuzione degli operai nei reparti si hanno al più le seguenti disponibilità orarie: 1600 ore nel reparto 1, 1400 ore nel reparto 2 e 1000 ore nel reparto 3. Pertanto in base ai tempi di lavorazione riportati nella tabella i vincoli da considerare sono:

$$x_{11} + 1.5x_{21} + 0.5x_{31} + 1.6x_{41} \leq 1600$$

$$\begin{aligned} 1.2x_{12} + 1.3x_{22} + 0.6x_{32} + x_{42} &\leq 1400 \\ 0.8x_{13} + 1.7x_{23} + 0.7x_{33} + 1.3x_{43} &\leq 1000. \end{aligned}$$

Inoltre si devono considerare dovuti all'ordine da soddisfare che possono essere scritti nella forma

$$\begin{aligned} x_{11} + x_{12} + x_{13} &\geq 1000 \\ x_{21} + x_{22} + x_{23} &\geq 600 \\ x_{31} + x_{32} + x_{33} &\geq 300 \\ x_{41} + x_{42} + x_{43} &\geq 200. \end{aligned}$$

Infine devono essere esplicitati i vincoli di

- non negatività delle variabili $x_{ij} \geq 0$, $i = 1, \dots, 4$, $j = 1, 2, 3$
- interezza delle variabili $x_{ij} \in \mathbf{Z}$, $i = 1, \dots, 4$, $j = 1, 2, 3$.

Quindi la formulazione completa è:

$$\begin{cases} \max (800(x_{11} + x_{12} + x_{13}) + 1200(x_{21} + x_{22} + x_{23}) + 950(x_{31} + x_{32} + x_{33}) + 1100(x_{41} + x_{42} + x_{43})) \\ x_{11} + 1.5x_{21} + 0.5x_{31} + 1.6x_{41} \leq 1600 \\ 1.2x_{12} + 1.3x_{22} + 0.6x_{32} + x_{42} \leq 1400 \\ 0.8x_{13} + 1.7x_{23} + 0.7x_{33} + 1.3x_{43} \leq 1000 \\ x_{11} + x_{12} + x_{13} \geq 1000 \\ x_{21} + x_{22} + x_{23} \geq 600 \\ x_{31} + x_{32} + x_{33} \geq 300 \\ x_{41} + x_{42} + x_{43} \geq 200 \\ x_{ij} \geq 0, \quad x_{ij} \in \mathbf{Z}, \quad i = 1, \dots, 4, \quad j = 1, 2, 3. \end{cases}$$

□

Modelli di pianificazione della produzione multiperiodo

Si tratta di problemi di allocazione ottima di risorse limitate analoghi a quelli già trattati, ma dove la pianificazione è effettuata su un orizzonte temporale composto da più periodi elementari; si richiede, ad esempio, di estendere la programmazione mensile della produzione di un'azienda in modo da ottenere un piano di produzione semestrale con possibilità di giacenze al termine di ciascun mese. L'esempio che segue riporta una semplice situazione di questo tipo.

Esempio 2.7.5 Una fabbrica produce due tipi di pneumatici A e B ed ha una gestione trimestrale della produzione. Per i prossimi tre mesi deve soddisfare il seguente ordine (espresso in numero di pneumatici richiesti ciascun mese)

	tipo A	tipo B
ottobre	4500	3000
novembre	5000	3500
dicembre	2000	8000

Per la produzione di questi pneumatici la fabbrica dispone di due macchine **M1** e **M2**. Per avere un pneumatico finito e pronto per essere venduto, è necessaria la lavorazione di materiale grezzo su solo una delle due macchine. Il numero di ore in cui le macchine sono disponibili ciascun mese sono riportate nella seguente tabella

	M1	M2
ottobre	1000	800
novembre	1500	1800
dicembre	600	1100

I tempi necessari per produrre questi pneumatici varia a seconda del tipo e della macchina usata. Tali tempi sono riportati nella seguente tabella (in ore)

	M1	M2
tipo A	0.10	0.12
tipo B	0.18	0.15

Il costo di lavorazione per ogni ora di macchina usata è uguale per entrambe le macchine ed è pari a lire 6000. Il costo del materiale grezzo necessario per produrre ciascun pneumatico è di lire 2500 per il tipo A e di lire 4000 per il tipo B.

Nel primo e nel secondo mese del trimestre è possibile produrre più di quanto richiesto nello stesso mese; la produzione in eccesso deve essere immagazzinata per essere usata nel mese successivo. Ogni mese, il costo di tale immagazzinamento è di lire 350 per ciascun pneumatico immagazzinato. Si assuma che all'inizio del trimestre non ci sia nessun prodotto immagazzinato e analogamente alla fine del trimestre non rimanga nessun prodotto immagazzinato.

Costruire un modello lineare che permetta di pianificare la produzione trimestrale minimizzando il costo complessivo trascurando l'interesse dei prodotti.

Formulazione.

Si tratta di un problema di allocazione ottima di risorse nel quale si deve tenere presente la possibilità dell'immagazzinamento del prodotto in eccesso.

– *Variabili.* Si introducono le variabili A_{Mi}^{ott} , A_{Mi}^{nov} , A_{Mi}^{dic} che indicano la quantità di pneumatici di tipo A prodotti dalla i –esima macchina ($i = 1, 2$) rispettivamente nei mesi di ottobre, novembre e dicembre. Analogamente B_{Mi}^{ott} , B_{Mi}^{nov} , B_{Mi}^{dic} indicheranno le quantità di pneumatici di tipo B prodotti dalla i –esima macchina ($i = 1, 2$) rispettivamente nei mesi di ottobre, novembre e dicembre. Si indichino inoltre con A_{im}^{ott} , A_{im}^{nov} , B_{im}^{ott} , B_{im}^{nov} le quantità di pneumatici di tipo A e B da immagazzinare nei mesi di ottobre e novembre.

– *Funzione obiettivo.* La funzione obiettivo da minimizzare è data dal costo complessivo di produzione. Poiché un'ora di lavorazione su una macchina costa lire 6000, e poiché i tempi di lavorazione cambiano a seconda delle macchine usate, per produrre ciascun pneumatico di tipo A si spende lire 600 se si utilizza la macchina 1, lire 720 se si utilizza la macchina 2. Analogamente, il costo di ciascun pneumatico del tipo B è di lire 1080 se si utilizza la macchina 1, e di lire 900 se si utilizza la macchina 2. Quindi tenendo conto del costo del materiale grezzo e dell'immagazzinamento, il costo complessivo sarà

$$\begin{aligned}
& 600(A_{M1}^{ott} + A_{M1}^{nov} + A_{M1}^{dic}) + 720(A_{M2}^{ott} + A_{M2}^{nov} + A_{M2}^{dic}) + \\
& + 1080(B_{M1}^{ott} + B_{M1}^{nov} + B_{M1}^{dic}) + 900(B_{M2}^{ott} + B_{M2}^{nov} + B_{M2}^{dic}) + \\
& + 2500((A_{M1}^{ott} + A_{M1}^{nov} + A_{M1}^{dic} + A_{M2}^{ott} + A_{M2}^{nov} + A_{M2}^{dic}) + \\
& + 4000(B_{M1}^{ott} + B_{M1}^{nov} + B_{M1}^{dic} + B_{M2}^{ott} + B_{M2}^{nov} + B_{M2}^{dic}) + \\
& + 350(A_{im}^{ott} + A_{im}^{nov} + B_{im}^{ott} + B_{im}^{nov})).
\end{aligned}$$

– *Vincoli.* I vincoli dovuti alla disponibilità limitata delle macchine sono

$$\begin{aligned}
0.10A_{M1}^{ott} + 0.18B_{M1}^{ott} & \leq 1000 \\
0.10A_{M1}^{nov} + 0.18B_{M1}^{nov} & \leq 1500 \\
0.10A_{M1}^{dic} + 0.18B_{M1}^{dic} & \leq 600
\end{aligned}$$

$$\begin{aligned}
0.12A_{M2}^{ott} + 0.15B_{M2}^{ott} &\leq 800 \\
0.12A_{M2}^{nov} + 0.15B_{M2}^{nov} &\leq 1800 \\
0.12A_{M2}^{dic} + 0.15B_{M2}^{dic} &\leq 1100.
\end{aligned}$$

Si hanno inoltre i seguenti vincoli dovuti alla richiesta e all'immagazzinamento:

$$\begin{aligned}
A_{M1}^{ott} + A_{M2}^{ott} &= 4500 + A_{im}^{ott} \\
A_{M1}^{nov} + A_{M2}^{nov} + A_{im}^{ott} &= 5000 + A_{im}^{nov} \\
A_{M1}^{dic} + A_{M2}^{dic} + A_{im}^{nov} &= 2000
\end{aligned}$$

$$\begin{aligned}
B_{M1}^{ott} + B_{M2}^{ott} &= 3000 + B_{im}^{ott} \\
B_{M1}^{nov} + B_{M2}^{nov} + B_{im}^{ott} &= 3500 + B_{im}^{nov} \\
B_{M1}^{dic} + B_{M2}^{dic} + B_{im}^{nov} &= 8000.
\end{aligned}$$

Si hanno inoltre i vincoli di non negatività sulle variabili.

Quindi il modello finale è:

$$\begin{cases}
\min \left(3100(A_{M1}^{ott} + A_{M1}^{nov} + A_{M1}^{dic}) + 3220(A_{M2}^{ott} + A_{M2}^{nov} + A_{M2}^{dic}) + \right. \\
\quad + 5080(B_{M1}^{ott} + B_{M1}^{nov} + B_{M1}^{dic}) + 4900(B_{M2}^{ott} + B_{M2}^{nov} + B_{M2}^{dic}) + \\
\quad + 350(A_{im}^{ott} + A_{im}^{nov} + B_{im}^{ott} + B_{im}^{nov}) \left. \right) \\
0.10A_{M1}^{ott} + 0.18B_{M1}^{ott} \leq 1000 \\
0.10A_{M1}^{nov} + 0.18B_{M1}^{nov} \leq 1500 \\
0.10A_{M1}^{dic} + 0.18B_{M1}^{dic} \leq 600 \\
0.12A_{M2}^{ott} + 0.15B_{M2}^{ott} \leq 800 \\
0.12A_{M2}^{nov} + 0.15B_{M2}^{nov} \leq 1800 \\
0.12A_{M2}^{dic} + 0.15B_{M2}^{dic} \leq 1100 \\
A_{M1}^{ott} + A_{M2}^{ott} = 4500 + A_{im}^{ott} \\
A_{M1}^{nov} + A_{M2}^{nov} + A_{im}^{ott} = 5000 + A_{im}^{nov} \\
A_{M1}^{dic} + A_{M2}^{dic} + A_{im}^{nov} = 2000 \\
B_{M1}^{ott} + B_{M2}^{ott} = 3000 + B_{im}^{ott} \\
B_{M1}^{nov} + B_{M2}^{nov} + B_{im}^{ott} = 3500 + B_{im}^{nov} \\
B_{M1}^{dic} + B_{M2}^{dic} + B_{im}^{nov} = 8000 \\
A_{Mi}^{ott} \geq 0, A_{Mi}^{nov} \geq 0, A_{Mi}^{dic} \geq 0, i = 1, 2 \\
B_{Mi}^{ott} \geq 0, B_{Mi}^{nov} \geq 0, B_{Mi}^{dic} \geq 0, i = 1, 2.
\end{cases}$$

□

2.7.2 Modelli di miscelazione

Si tratta di problemi in cui si hanno a disposizione un certo numero di sostanze ciascuna delle quali ha un certo costo ed un determinato contenuto di componenti utili. Si vuole ottenere la miscela più economica di queste sostanze in modo che contenga una certa quantità di ciascuno dei componenti utili.

Esempi introduttivi

Esempio 2.7.6 *Un'industria conserviera deve produrre succhi di frutta mescolando polpa di frutta e dolcificante ottenendo un prodotto finale che deve soddisfare alcuni requisiti riguardanti il contenuto di vitamina C, di sali minerali e di calorie. La polpa di frutta e il dolcificante vengono acquistati al costo rispettivamente di Lire 400 e Lire 600 ogni ettagrammo. Inoltre dalle etichette si ricava che 100 grammi di polpa di frutta contengono 140 mg di vitamina C, 20 mg di sali minerali e 50 calorie mentre 100 grammi di dolcificante contengono 10 mg di sali minerali, 100 calorie e non contengono vitamina C. I requisiti che il prodotto finale (cioè il succo di frutta pronto per la vendita) deve avere sono i seguenti: il succo di frutta deve contenere almeno 70 mg di vitamina C, almeno 30 mg di sali minerali e un contenuto calorico di almeno 150 calorie. Si devono determinare le quantità di polpa di frutta e di dolcificante da utilizzare nella produzione del succo di frutta in modo da minimizzare il costo complessivo dell'acquisto dei due componenti base.*

Formulazione.

Si vuole costruire un modello di Programmazione Lineare che rappresenti il problema in analisi tenendo presente i requisiti di qualità richiesti. Si verifica facilmente che le ipotesi fondamentali di un modello di Programmazione Lineare sono soddisfatte.

- *Variabili.* È naturale associare le variabili di decisione alle quantità di polpa di frutta e di dolcificante da utilizzare per la produzione del succo di frutta. Quindi siano x_1 e x_2 rispettivamente le quantità espresse in ettagrammi di polpa di frutta e di dolcificante che devono essere utilizzate.
- *Funzione obiettivo.* È rappresentata dal costo complessivo dell'acquisto dei due componenti base e quindi è data da $400x_1 + 600x_2$. Questa espressione naturalmente deve essere minimizzata.
- *Vincoli.* Poiché un ettagrammo di polpa contiene 140 mg di vitamina C e il dolcificante non contiene vitamina C, il primo vincolo da considerare riguardante il contenuto di vitamina C del succo di frutta si può scrivere nella forma

$$140x_1 \geq 70.$$

Analogamente per rispettare il requisito sul contenuto di sali minerali del succo di frutta si dovrà imporre il vincolo

$$20x_1 + 10x_2 \geq 30.$$

Infine il vincolo sul contenuto calorico del succo di frutta si può esprimere nella forma

$$50x_1 + 100x_2 \geq 150.$$

Infine si deve esplicitare il vincolo di non negatività sulle variabili cioè

$$x_1 \geq 0, x_2 \geq 0.$$

Quindi la formulazione finale è

$$\begin{cases} \min(400x_1 + 600x_2) \\ 140x_1 \geq 70 \\ 20x_1 + 10x_2 \geq 30 \\ 50x_1 + 100x_2 \geq 150 \\ x_1 \geq 0, x_2 \geq 0 \end{cases}$$

Esempio 2.7.7 – IL PROBLEMA DELLA DIETA

Una dieta prescrive che giornalmente devono essere assimilate quantità predeterminate di calorie, proteine e calcio, intese come fabbisogni minimi giornalieri, disponendo di cinque alimenti base (pane, latte, uova, carne, dolce). Tali fabbisogni minimi sono di 2000 calorie, 50 g. di proteine, 700 mg. di calcio. Dalle tabelle dietetiche si ricavano i seguenti contenuti di calorie (in cal.), proteine (in g.), calcio (in mg.) per ogni singola porzione di ciascun alimento, intendendo come porzione una quantità espressa in grammi e quindi frazionabile.

	Pane	Latte	Uova	Carne	Dolce
calorie	110	160	180	260	420
proteine	4	8	13	14	4
calcio	2	285	54	80	22

I costi (in migliaia di lire) e il numero massimo di porzioni tollerate giornalmente sono i seguenti

	Pane	Latte	Uova	Carne	Dolce
costo	2	3	4	19	20
porz.	4	8	3	2	2

Determinare una dieta a costo minimo che soddisfi le prescrizioni richieste.

Formulazione.

Poiché si è supposto che le porzioni siano frazionabili ed inoltre valgono le ipotesi di linearità, si può costruire un modello di Programmazione Lineare per rappresentare il problema in analisi.

– *Variabili.* È ovvio introdurre le variabili x_1, x_2, x_3, x_4, x_5 indicanti le quantità di porzioni dei singoli alimenti da includere giornalmente nella dieta.

– *Funzione obiettivo.* È rappresentata dal costo complessivo ed è quindi data da

$$2x_1 + 3x_2 + 4x_3 + 19x_4 + 20x_5.$$

– *Vincoli.* Poiché sono prescritti i fabbisogni minimi giornalieri, si avranno i seguenti vincoli:

$$\begin{aligned} \text{calorie} &\longrightarrow 110x_1 + 160x_2 + 180x_3 + 260x_4 + 420x_5 \geq 2000 \\ \text{proteine} &\longrightarrow 4x_1 + 8x_2 + 13x_3 + 14x_4 + 4x_5 \geq 50 \\ \text{calcio} &\longrightarrow 2x_1 + 285x_2 + 54x_3 + 80x_4 + 22x_5 \geq 700 \end{aligned}$$

Inoltre i vincoli sul numero massimo di porzioni giornaliere di ciascun alimento e di non negatività

$$0 \leq x_1 \leq 4, 0 \leq x_2 \leq 2, 0 \leq x_3 \leq 3, 0 \leq x_4 \leq 2, 0 \leq x_5 \leq 2.$$

La formulazione completa sarà quindi

$$\begin{cases} \min (2x_1 + 3x_2 + 4x_3 + 19x_4 + 20x_5) \\ 110x_1 + 160x_2 + 180x_3 + 260x_4 + 420x_5 \geq 2000 \\ 4x_1 + 8x_2 + 13x_3 + 14x_4 + 4x_5 \geq 50 \\ 2x_1 + 285x_2 + 54x_3 + 80x_4 + 22x_5 \geq 700 \\ 0 \leq x_1 \leq 4, 0 \leq x_2 \leq 2, 0 \leq x_3 \leq 3, 0 \leq x_4 \leq 2, 0 \leq x_5 \leq 2. \end{cases}$$

Se inoltre si vuole supporre, ad esempio, che nella dieta sia presente almeno una porzione di dolce e due di latte si dovranno imporre i vincoli $x_5 \geq 1$ e $x_2 \geq 2$ da aggiungere alla precedente formulazione. In questo caso, i vincoli già presenti $x_5 \geq 0$ e $x_2 \geq 0$ sono ridondanti. \square

Formulazione generale di un problema di miscelazione

Formalmente, supponiamo di disporre di n sostanze diverse che indichiamo con $\mathbf{S}_1, \mathbf{S}_2, \dots, \mathbf{S}_n$ ciascuna delle quali contenga una certa quantità di ciascuno degli m componenti utili che indichiamo con $\mathbf{C}_1, \mathbf{C}_2, \dots, \mathbf{C}_m$. Supponendo che ogni sostanza \mathbf{S}_j abbia costo unitario c_j , $j = 1, \dots, n$

$$\begin{array}{cccc} \mathbf{S}_1 & \mathbf{S}_2 & \cdots & \mathbf{S}_n \\ c_1 & c_2 & \cdots & c_n \end{array}$$

si desidera ottenere la miscela più economica che soddisfi alcuni requisiti qualitativi, cioè contenga una quantità non inferiore a b_i di ciascun \mathbf{C}_i , $i = 1, \dots, m$

$$\begin{array}{cccc} \mathbf{C}_1 & \mathbf{C}_2 & \cdots & \mathbf{C}_m \\ b_1 & b_2 & \cdots & b_m. \end{array}$$

Si indichi con a_{ij} , $i = 1, \dots, m$, $j = 1, \dots, n$ la quantità di componente \mathbf{C}_i presente nella sostanza \mathbf{S}_j . Si può così costruire la seguente tabella

	\mathbf{S}_1	\cdots	\mathbf{S}_j	\cdots	\mathbf{S}_n
\mathbf{C}_1	a_{11}	\cdots	a_{1j}	\cdots	a_{1n}
\vdots	\vdots		\vdots		\vdots
\mathbf{C}_i	a_{i1}	\cdots	a_{ij}	\cdots	a_{in}
\vdots	\vdots		\vdots		\vdots
\mathbf{C}_m	a_{m1}	\cdots	a_{mj}	\cdots	a_{mn}

Formulazione.

Supponendo che valgano le ipotesi di proporzionalità, additività ed inoltre assumendo che le quantità di sostanze da utilizzare siano frazionabili, si può formulare questo problema in termini di un problema di Programmazione Lineare.

– *Variabili.* È naturale introdurre le variabili di decisione x_1, x_2, \dots, x_n rappresentanti la quantità di ciascuna sostanza $\mathbf{S}_1, \mathbf{S}_2, \dots, \mathbf{S}_n$ da utilizzare nella miscela. Queste saranno le incognite del problema. Introducendo come spazio delle variabili lo spazio delle n -uple reali \mathbb{R}^n si può considerare un $x \in \mathbb{R}^n$ definendo $x = (x_1, \dots, x_n)^T$.

– *Funzione obiettivo.* Per le ipotesi fatte, la funzione obiettivo può essere scritta

$$z = c_1 x_1 + \dots + c_n x_n = \sum_{j=1}^n c_j x_j.$$

Introducendo $c \in \mathbb{R}^n$, definito $c = (c_1, \dots, c_n)^T$, la funzione obiettivo può essere scritta in notazione vettoriale

$$z = c^T x.$$

– *Vincoli.* Si devono introdurre i seguenti vincoli:

- *Vincoli di qualità.*

Tenendo conto del fatto che la miscela deve contenere una quantità non inferiore a b_i di ciascun componente \mathbf{C}_i si dovrà avere

$$\sum_{j=1}^n a_{ij} x_j \geq b_i, \quad i = 1, \dots, m.$$

- Vincoli di non negatività.

Si devono infatti considerare i vincoli di non negatività sulle variabili cioè $x_j \geq 0$, $j = 1, \dots, n$.

Introducendo la matrice $(m \times n)$

$$A = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \dots & a_{mn} \end{pmatrix}$$

e il vettore $b = (b_1, \dots, b_m)^T$ la formulazione completa del problema può essere scritta nella forma

$$\begin{cases} \min c^T x \\ Ax \geq b \\ x \geq 0. \end{cases}$$

Nella pratica, potrebbe essere necessario introdurre ulteriori vincoli:

- possono essere presenti limitazioni superiori o inferiori sulle variabili cioè $x_j \geq L$, $x_j \leq M$, $j = 1, \dots, n$;
- se è richiesto anche che la miscela contenga una quantità non superiore ad un valore d_i di ciascun componente \mathbf{C}_i si dovrà aggiungere alla formulazione un altro vincolo di qualità:

$$\sum_{j=1}^n a_{ij} x_j \leq d_i, \quad i = 1, \dots, m;$$

- in alcuni casi si richiede che una certa sostanza appartenga alla miscela solo se un'altra sostanza vi appartiene (o non vi appartiene). Questi vincoli richiedono l'uso di variabili booleane come descritto in seguito.

□

Altri esempi di modelli di miscelazione

Esempio 2.7.8 Il prodotto finale di una fabbrica è ottenuto raffinando materie prime grezze e miscelandole insieme. Queste materie prime possono essere di due categorie: naturali e sintetizzate. In particolare, sono disponibili tre materie prime naturali (**N1**, **N2**, **N3**) e due materie prime sintetizzate (**S1**, **S2**). Le materie prime naturali e quelle sintetizzate richiedono differenti linee di produzione. Ogni settimana è possibile raffinare non più di 500 quintali di materie prime naturali e non più di 300 quintali di materie prime sintetizzate. Si assume che non ci sia perdita di peso durante la raffinazione e che si possa trascurare il costo di raffinazione. Inoltre esiste una restrizione tecnologica sulla gradazione del prodotto finale: nell'unità di misura in cui questa gradazione è misurata, essa deve essere tra 2 e 7; si assume che tale gradazione nella miscela finale dipenda linearmente dalle singole gradazioni delle materie prime componenti. Nella tabella che segue è riportato il costo (in migliaia di lire) per quintale e la gradazione delle materie prime grezze.

	N1	N2	N3	S1	S2
costo	300	190	250	200	230
grad.	6.0	1.9	8.5	5.0	3.5

Il prodotto finale viene venduto a 350 migliaia di lire per quintale. Determinare come va pianificata la produzione settimanale per massimizzare il profitto netto.

Formulazione.

– *Variabili.* Introduciamo le variabili di decisione x_1, x_2, x_3, x_4, x_5 rappresentanti le quantità (in quintali) di **N1**, **N2**, **N3**, **S1**, **S2** che devono essere comprate e raffinate in una settimana. Inoltre introduciamo una ulteriore variabile y che indica la quantità di prodotto finale che deve essere fabbricato.

– *Funzione obiettivo.* La funzione obiettivo da massimizzare sarà data dal profitto netto cioè da

$$350y - 300x_1 - 190x_2 - 250x_3 - 200x_4 - 230x_5.$$

– *Vincoli.* Sono presenti tre tipi di vincoli

- capacità di raffinamento

$$x_1 + x_2 + x_3 \leq 500$$

$$x_4 + x_5 \leq 300;$$

- limitazioni sulla gradazione

$$6.0x_1 + 1.9x_2 + 8.5x_3 + 5.0x_4 + 3.5x_5 \leq 7y$$

$$6.0x_1 + 1.9x_2 + 8.5x_3 + 5.0x_4 + 3.5x_5 \geq 2y;$$

- vincolo di continuità

$$x_1 + x_2 + x_3 + x_4 + x_5 = y.$$

Questo vincolo di continuità esprime il fatto che il peso finale del prodotto deve essere uguale alla somma dei pesi degli ingredienti.

Inoltre si devono esplicitare i vincoli di non negatività delle variabili.

La formulazione finale risulta quindi

$$\begin{cases} \max (-300x_1 - 190x_2 - 250x_3 - 200x_4 - 230x_5 + 350y) \\ x_1 + x_2 + x_3 \leq 500 \\ x_4 + x_5 \leq 300 \\ 6.0x_1 + 1.9x_2 + 8.5x_3 + 5.0x_4 + 3.5x_5 - 7y \leq 0 \\ 6.0x_1 + 1.9x_2 + 8.5x_3 + 5.0x_4 + 3.5x_5 - 2y \geq 0 \\ x_1 + x_2 + x_3 + x_4 + x_5 - y = 0 \\ x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0, x_5 \geq 0, y \geq 0 \end{cases}$$

□

Osservazione 2.7.9 Un errore comune è quello di scrivere i vincoli sulla gradazione

$$6.0x_1 + 1.9x_2 + 8.5x_3 + 5.0x_4 + 3.5x_5 \leq 7$$

$$6.0x_1 + 1.9x_2 + 8.5x_3 + 5.0x_4 + 3.5x_5 \geq 2.$$

Queste relazioni sono evidentemente dimensionalmente errate: il primo membro ha le dimensioni di *gradazione* \times *quantità* mentre il secondo membro ha le dimensioni della *gradazione*. Tuttavia, invece delle variabili x_i in queste due diseguaglianze si potevano usare le variabili x_i/y per rappresentare le *proporzioni degli ingredienti*, piuttosto che le *quantità assolute* x_i ; ovviamente, in questo caso si dovevano modificare anche le altre espressioni. Comunque, l'uso delle variabili x_i/y è ovviamente possibile solo nel caso in cui la quantità di prodotto fabbricato è non nulla, cioè $y \neq 0$.

Esempio 2.7.10 Una raffineria produce quattro tipi di benzine grezze ($\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3, \mathbf{B}_4$) e le miscola allo scopo di ottenere carburanti di due diverse qualità ($\mathbf{C}_1, \mathbf{C}_2$). Le quantità di benzine grezze non utilizzate nella produzione delle miscele possono essere vendute direttamente. La seguente tabella riassume i dati delle benzine grezze, cioè il numero di ottani, la quantità (in ettolitri) che si può produrre al giorno e il costo (in migliaia di lire) di un ettolitro di ciascuna benzina.

	\mathbf{B}_1	\mathbf{B}_2	\mathbf{B}_3	\mathbf{B}_4
n. ottani	90	73	79	86
ettolitri	3500	6000	4500	5200
costo	260	210	190	220

Nella seguente tabella sono riportate le caratteristiche che devono avere le miscele cioè il minimo numero di ottani e il prezzo di vendita di un ettolitro di carburante (in migliaia di lire)

	\mathbf{C}_1	\mathbf{C}_2
min. n. ottani	80	85
prezzo	350	520

Inoltre il mercato è in grado di assorbire non più di 25000 ettolitri al giorno del carburante \mathbf{C}_1 , mentre richiede almeno 10000 ettolitri al giorno di carburante \mathbf{C}_2 . Infine, i quantitativi di benzine grezze prodotti ma non utilizzati nella preparazione delle miscele sono rivenduti al prezzo di 280 migliaia di lire per ettolitro se il numero di ottani è non inferiore a 80, e a 250 migliaia di lire per ettolitro altrimenti. Occorre pianificare la produzione giornaliera della raffineria, cioè le quantità e le composizioni delle due miscele, massimizzando il profitto ottenuto dalla vendita dei prodotti. Assumere che il numero di ottani di ciascuna miscela dipenda in modo lineare dalle gradazioni delle benzine componenti.

Formulazione.

– *Variabili.* È conveniente scegliere come variabili di decisione le quantità (in barili) di benzina grezza \mathbf{B}_i utilizzate nella miscela \mathbf{C}_j che indichiamo con x_{ij} , $i = 1, \dots, 4$, $j = 1, 2$. Inoltre denotiamo con y_i , $i = 1, \dots, 4$ le quantità di benzine grezze prodotte ma non utilizzate nelle miscele.

– *Funzione obiettivo.* La funzione obiettivo da massimizzare è data dal ricavo ottenuto dalla vendita dei prodotti sottratto dei costi di produzione. Quindi è data da

$$\begin{aligned}
 z = & 350(x_{11} + x_{21} + x_{31} + x_{41}) + 520(x_{12} + x_{22} + x_{32} + x_{42}) + \\
 & + 280(y_1 + y_4) + 250(y_2 + y_3) + \\
 & - 260(x_{11} + x_{12} + y_1) - 210(x_{21} + x_{22} + y_2) + \\
 & - 190(x_{31} + x_{32} + y_3) - 220(x_{41} + x_{42} + y_4).
 \end{aligned}$$

– *Vincoli.* I vincoli sulla capacità produttiva sono

$$x_{11} + x_{12} + y_1 \leq 3500$$

per quanto riguarda la benzina \mathbf{B}_1 e analoghi per le altre benzine. Tuttavia tali vincoli possono essere imposti, senza perdere generalità, come vincoli di uguaglianza considerando che non vi è alcuna convenienza a sotto-utilizzare le capacità produttive della raffineria:

$$x_{11} + x_{12} + y_1 = 3500$$

$$x_{21} + x_{22} + y_2 = 6000$$

$$x_{31} + x_{32} + y_3 = 4500$$

$$x_{41} + x_{42} + y_4 = 5200.$$

Analizziamo ora i vincoli dovuti al minimo numero di ottani che devono avere le miscele; essi sono dati da

$$\begin{aligned} 90x_{11} + 73x_{21} + 79x_{31} + 86x_{41} &\geq 80(x_{11} + x_{21} + x_{31} + x_{41}) \\ 90x_{12} + 73x_{22} + 79x_{32} + 86x_{42} &\geq 85(x_{12} + x_{22} + x_{32} + x_{42}). \end{aligned} \quad (2.6)$$

A queste espressioni si può anche arrivare considerando che il numero degli ottani di ciascuna miscela, per ipotesi, dipende linearmente dalle gradazioni delle benzine componenti e quindi è dato dalla media pesata dei numeri di ottani delle benzine componenti, con pesi costituiti dalle quantità di ciascun componente; quindi il numero di ottani della j -esima miscela è dato da

$$\frac{90x_{1j} + 73x_{2j} + 79x_{3j} + 86x_{4j}}{x_{1j} + x_{2j} + x_{3j} + x_{4j}}.$$

Questa espressione è valida solamente se la miscela è prodotta in quantità non nulla, perché in questo caso il denominatore è non nullo. Esprimendo con una disequazione la richiesta che il numero di ottani di tale miscela sia non inferiore al rispettivo limite minimo e moltiplicando entrambi i membri della disequazione per il denominatore della frazione si ottengono i vincoli richiesti nella forma data dalle (2.6) che valgono anche nel caso di produzione nulla delle miscele. Svolgendo i calcoli nelle (2.6) si ottengono i seguenti vincoli

$$10x_{11} - 7x_{21} - x_{31} + 6x_{41} \geq 0$$

$$5x_{12} - 12x_{22} - 6x_{32} + x_{42} \geq 0.$$

Si devono inoltre esprimere vincoli di mercato, cioè

$$x_{11} + x_{21} + x_{31} + x_{41} \leq 25000$$

$$x_{12} + x_{22} + x_{32} + x_{42} \geq 10000$$

e i vincoli di non negatività sulle variabili

$$x_{ij} \geq 0, \quad y_i \geq 0, \quad i = 1, \dots, 4; \quad j = 1, \dots, 3.$$

Quindi, la formulazione completa è

$$\left\{ \begin{array}{l} \max(90x_{11} + 140x_{21} + 160x_{31} + 130x_{41} + 260x_{12} + 310x_{22} + \\ \quad + 330x_{32} + 300x_{42} + 20y_1 + 40y_2 + 60y_3 + 60y_4) \\ x_{11} + x_{12} + y_1 = 3500 \\ x_{21} + x_{22} + y_2 = 6000 \\ x_{31} + x_{32} + y_3 = 4500 \\ x_{41} + x_{42} + y_4 = 5200 \\ 10x_{11} - 7x_{21} - x_{31} + 6x_{41} \geq 0 \\ 5x_{12} - 12x_{22} - 6x_{32} + x_{42} \geq 0 \\ x_{11} + x_{21} + x_{31} + x_{41} \leq 25000 \\ x_{12} + x_{22} + x_{32} + x_{42} \geq 10000 \\ x_{ij} \geq 0, \quad y_i \geq 0, \quad i = 1, \dots, 4; \quad j = 1, 2. \end{array} \right.$$

□

2.7.3 Modelli di trasporto

Si tratta di problemi in cui si hanno un certo numero di località (origini) ciascuna delle quali ha una quantità fissata di merce disponibile e un certo numero di clienti residenti in altre località (destinazioni) i quali richiedono quantitativi precisi di merce. Quindi conoscendo il costo unitario del trasporto della merce da ciascuna località origine a ciascuna località destinazione è necessario pianificare i trasporti, cioè la quantità di merce che deve essere trasportata da ciascuna località origine a ciascuna località destinazione in modo da soddisfare l'ordine dei clienti minimizzando il costo complessivo derivante dai trasporti.

Esempio introduttivo

Esempio 2.7.11 *Un'industria dell'acciaio dispone di due miniere \mathbf{M}_1 e \mathbf{M}_2 e di tre impianti di produzione \mathbf{P}_1 \mathbf{P}_2 \mathbf{P}_3 . Il minerale estratto deve essere giornalmente trasportato agli impianti di produzione soddisfacendo le rispettive richieste. Le miniere \mathbf{M}_1 e \mathbf{M}_2 producono giornalmente rispettivamente 130 e 200 tonnellate di minerale. Gli impianti richiedono giornalmente le seguenti quantità (in tonnellate) di minerale*

\mathbf{P}_1	\mathbf{P}_2	\mathbf{P}_3
80	100	150

Il costo (in migliaia di lire) del trasporto da ciascuna miniera a ciascun impianto di produzione è riportato nella seguente tabella

	\mathbf{P}_1	\mathbf{P}_2	\mathbf{P}_3
\mathbf{M}_1	10	8	21
\mathbf{M}_2	12	20	14

Formulare un modello che descriva il trasporto dalle miniere agli impianti di produzione in modo da minimizzare il costo globale del trasporto.

Analisi del problema.

È un problema di trasporti con 2 origini (\mathbf{M}_1 , \mathbf{M}_2) e 3 destinazioni (\mathbf{P}_1 \mathbf{P}_2 \mathbf{P}_3). Poiché risulta $a_1 + a_2 = 130 + 200 = 330$ e $b_1 + b_2 + b_3 = 80 + 100 + 150 = 330$, è soddisfatta l'ipotesi di ammissibilità richiesta dal Teorema 2.7.14

Formulazione.

– *Variabili.* Associamo le variabili di decisione alle quantità di minerale che deve essere trasportato; indichiamo con x_{ij} $i = 1, 2$, $j = 1, 2, 3$, le quantità (in tonnellate) di minerale da trasportare giornalmente da ciascuna miniera \mathbf{M}_i a ciascun impianto di produzione \mathbf{P}_j .

– *Funzione obiettivo.* La funzione obiettivo da minimizzare è data dalla somma dei costi dei trasporti cioè da

$$z = 10x_{11} + 8x_{12} + 21x_{13} + 12x_{21} + 20x_{22} + 14x_{23}.$$

– *Vincoli.* I vincoli di origine esprimono il fatto che la somma della quantità di minerale trasportato dalla miniera \mathbf{M}_i deve essere uguale alla disponibilità giornaliera della miniera stessa:

$$x_{11} + x_{12} + x_{13} = 130$$

$$x_{21} + x_{22} + x_{23} = 200.$$

I vincoli di destinazione esprimono il fatto che la somma delle quantità di minerale trasportato all'impianto di produzione \mathbf{P}_j deve essere pari alla richiesta giornaliera di tale impianto:

$$\begin{aligned} x_{11} + x_{21} &= 80 \\ x_{12} + x_{22} &= 100 \\ x_{13} + x_{23} &= 150. \end{aligned}$$

Infine si devono considerare i vincoli di non negatività $x_{ij} \geq 0$, $i = 1, 2$, $j = 1, 2, 3$.

La formulazione completa è quindi

$$\begin{cases} \min (10x_{11} + 8x_{12} + 21x_{13} + 12x_{21} + 20x_{22} + 14x_{23}) \\ x_{11} + x_{12} + x_{13} = 130 \\ x_{21} + x_{22} + x_{23} = 200 \\ x_{11} + x_{21} = 80 \\ x_{12} + x_{22} = 100 \\ x_{13} + x_{23} = 150 \\ x_{ij} \geq 0, \quad i = 1, 2, \quad j = 1, 2, 3. \end{cases}$$

□

Formulazione generale di un problema di trasporti

Sono definite m località *origini* indicate con $\mathbf{O}_1, \dots, \mathbf{O}_m$, e n località *destinazioni* indicate con $\mathbf{D}_1, \dots, \mathbf{D}_n$. Ogni origine \mathbf{O}_i , ($i = 1, \dots, m$) può fornire una certa disponibilità $a_i \geq 0$ di merce che deve essere trasferita dalle origini alle destinazioni

$$\begin{array}{ccc} \mathbf{O}_1 & \dots & \mathbf{O}_m \\ a_1 & \dots & a_m. \end{array}$$

Ad ogni destinazione \mathbf{D}_j , ($j = 1, \dots, n$) è richiesta una quantità $b_j \geq 0$ di merce.

$$\begin{array}{ccc} \mathbf{D}_1 & \dots & \mathbf{D}_n \\ b_1 & \dots & b_n. \end{array}$$

Supponiamo che il costo del trasporto di una unità di merce da \mathbf{O}_i a \mathbf{D}_j sia pari a c_{ij} . Tali costi nella realtà sono spesso collegati alle distanze tra origini e destinazioni.

Il problema consiste nel pianificare i trasporti in modo da soddisfare le richieste delle destinazioni minimizzando il costo del trasporto complessivo nella seguente ipotesi:

- la disponibilità complessiva uguaglia la richiesta complessiva, cioè

$$\sum_{i=1}^m a_i = \sum_{j=1}^n b_j; \quad (2.7)$$

si escludono possibilità di giacenze nelle origini, cioè tutta la merce prodotta in una origine deve essere trasportata in una delle destinazioni; si escludono possibilità di giacenze nelle destinazioni, cioè la quantità totale che arriva in una destinazione \mathbf{D}_j deve uguagliare la richiesta b_j .

Formulazione.

Si vuole dare una formulazione del problema in esame in termini di un problema di programmazione lineare supponendo quindi che siano verificate le ipotesi di linearità e continuità.

– *Variabili.* Per ogni coppia di origine e destinazione \mathbf{O}_i , \mathbf{D}_j si introducono le variabili di decisione x_{ij} rappresentanti la quantità di merce da trasportare da \mathbf{O}_i , a \mathbf{D}_j . Si tratta di mn variabili

	D₁	...	D_j	...	D_n
O₁	x_{11}	...	x_{1j}	...	x_{1n}
\vdots	\vdots		\vdots		\vdots
O_i	x_{i1}	...	x_{ij}	...	x_{in}
\vdots	\vdots		\vdots		\vdots
O_m	x_{m1}	...	x_{mj}	...	x_{mn}

– *Funzione obiettivo.* La funzione obiettivo da minimizzare sarà data da costo totale del trasporto e quindi da

$$z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}.$$

– *Vincoli.* Per le ipotesi fatte, si avranno due tipi di vincoli:

- vincoli di origine

$$\sum_{j=1}^n x_{ij} = a_i \quad i = 1, \dots, m; \quad (2.8)$$

impongono che tutta la merce prodotta in una origine sia trasportata alle destinazioni; si tratta di m vincoli;

- vincoli di destinazione

$$\sum_{i=1}^m x_{ij} = b_j \quad j = 1, \dots, n; \quad (2.9)$$

impongono che la quantità totale di merce che arriva in ciascuna delle destinazioni uguaglia la richiesta; si tratta di n vincoli.

Si devono infine considerare i vincoli di non negatività delle variabili

$$x_{ij} \geq 0 \quad i = 1, \dots, n; \quad j = 1, \dots, m.$$

Si è così ottenuta una formulazione del problema dei trasporti con mn variabili e $m + n + mn$ vincoli:

$$\begin{cases} \min \left(\sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \right) \\ \sum_{j=1}^n x_{ij} = a_i \quad i = 1, \dots, m \\ \sum_{i=1}^m x_{ij} = b_j \quad j = 1, \dots, n \\ x_{ij} \geq 0 \quad i = 1, \dots, n; \quad j = 1, \dots, m. \end{cases} \quad (2.10)$$

Osservazione 2.7.12 È chiaro che per le ipotesi fatte dovrà risultare

$$\sum_{i=1}^m \sum_{j=1}^n x_{ij} = \sum_{j=1}^n \sum_{i=1}^m x_{ij} = \sum_{i=1}^m a_i = \sum_{j=1}^n b_j.$$

Osservazione 2.7.13 Sempre nell'ipotesi (2.7), una equazione dei vincoli di bilanciamento è ridondante e si può eliminare. Si può facilmente comprendere ciò esaminando direttamente un esempio particolare con $m = 3$ ed $n = 4$; in questo caso i vincoli possono essere scritti

$$x_{11} + x_{12} + x_{13} + x_{14} = a_1 \quad (A)$$

$$x_{21} + x_{22} + x_{23} + x_{24} = a_2 \quad (B)$$

$$x_{31} + x_{32} + x_{33} + x_{34} = a_3 \quad (C)$$

$$x_{11} + x_{21} + x_{31} = b_1 \quad (D)$$

$$x_{12} + x_{22} + x_{32} = b_2 \quad (E)$$

$$x_{13} + x_{23} + x_{33} = b_3 \quad (F)$$

$$x_{14} + x_{24} + x_{34} = b_4. \quad (G)$$

È facile osservare come sommando le equazioni (D), (E), (F), (G) e sottraneando da questa somma la somma delle equazioni (B) e (C) e tenendo presente che per ipotesi $a_1 + a_2 + a_3 = b_1 + b_2 + b_3 + b_4$, si ottiene l'equazione (A); quindi l'equazione (A) è ridondante e può essere eliminata dalla formulazione.

Ciò vale in generale e quindi gli $m + n$ vincoli di origine e di destinazione per un generico problema di trasporti possono essere ridotti a $m + n - 1$ equazioni indipendenti in mn variabili; infatti analogamente al caso particolare ora analizzato si può considerare la formulazione generale; sommando la (2.9) per $j = 1, \dots, n$ si ottiene

$$\sum_{j=1}^n \sum_{i=1}^m x_{ij} = \sum_{j=1}^n b_j \quad (2.11)$$

e sommando la (2.8) per $i = 1, \dots, m$ con $i \neq i_o$, si ottiene

$$\sum_{\substack{i=1 \\ i \neq i_o}}^m \sum_{j=1}^n x_{ij} = \sum_{\substack{i=1 \\ i \neq i_o}}^m a_i. \quad (2.12)$$

Sottraendo membro a membro la (2.11) e la (2.12) e tenendo conto che per ipotesi vale $\sum_{i=1}^m a_i = \sum_{j=1}^n b_j$ si ottiene l'equazione

$$\sum_{j=1}^n x_{i_o j} = a_{i_o}$$

che quindi è ridondante. □

Esaminiamo, ora, un risultato che è una condizione necessaria e sufficiente affinché un generico problema dei trasporti scritto nella forma (2.10) con $a_i \geq 0$ e $b_j \geq 0$ abbia soluzione; tale risultato chiarisce perché nella formulazione classica del problema dei trasporti si adotta l'ipotesi (2.7) cioè che la disponibilità complessiva uguagli la richiesta complessiva.

Teorema 2.7.14 Condizione necessaria e sufficiente affinché il problema (2.10) ammetta soluzione, cioè che esista una soluzione ammissibile, è che risulti

$$\sum_{i=1}^m a_i = \sum_{j=1}^n b_j. \quad (2.13)$$

Dim.: Dimostriamo innanzitutto la necessità, cioè che se esiste una soluzione ammissibile x_{ij} allora la condizione (2.13) deve essere verificata; poiché x_{ij} deve soddisfare i vincoli, dalle equazioni dei vincoli nella (2.10) si ottiene

$$\sum_{i=1}^m \sum_{j=1}^n x_{ij} = \sum_{i=1}^m a_i$$

$$\sum_{j=1}^n \sum_{i=1}^m x_{ij} = \sum_{j=1}^n b_i,$$

e sottraendo membro a membro si ha

$$\sum_{i=1}^m a_i - \sum_{j=1}^n b_j = 0$$

che è la (2.13).

Dimostriamo ora la sufficienza; supponiamo quindi che valga la (2.13) e poniamo

$$\sum_{i=1}^m a_i = \sum_{j=1}^n b_j = A.$$

Si vuole allora dimostrare che esiste una soluzione ammissibile; infatti, sia $x_{ij} := \frac{a_i b_j}{A}$; allora x_{ij} ora definito è una soluzione ammissibile per il problema dei trasporti. Infatti risulta innanzitutto $x_{ij} \geq 0$ per ogni $i = 1, \dots, m$ e $j = 1, \dots, n$ per la non negatività degli a_i e dei b_j ; inoltre

$$\sum_{j=1}^n x_{ij} = \sum_{j=1}^n \frac{a_i b_j}{A} = \frac{a_i \sum_{j=1}^n b_j}{A} = a_i$$

$$\sum_{i=1}^m x_{ij} = \sum_{i=1}^m \frac{a_i b_j}{A} = \frac{b_j \sum_{i=1}^m a_i}{A} = b_j$$

e quindi x_{ij} soddisfando i vincoli del problema è una soluzione ammissibile. \square

Il teorema appena dimostrato garantisce quindi che, se è soddisfatta l'ipotesi (2.7) allora il problema dei trasporti ammette sempre soluzione.

Osservazione 2.7.15 La soluzione ammissibile del teorema, ovviamente, *non è l'unica soluzione* del problema.

Riportiamo di seguito, senza dimostrazione, un altro risultato di fondamentale importanza nella trattazione del problema dei trasporti.

Teorema 2.7.16 *Se nel problema dei trasporti le a_i , $i = 1, \dots, m$ e le b_j , $j = 1, \dots, n$ sono intere e se il problema ammette soluzione ottima, allora ha una soluzione ottima intera.*

Passiamo, ora, ad analizzare alcune varianti della formulazione classica del problema dei trasporti; può infatti accadere che non tutte le rotte di trasporto siano disponibili: se non è possibile il trasporto da una certa origine \mathbf{O}_i ad una destinazione \mathbf{D}_j si pone, per convenzione, $c_{ij} = \infty$. Oppure possono esistere rotte di trasporto in cui vi sono limitazioni sulle quantità massima di merci trasportabili.

Infine, si può supporre che la disponibilità complessiva possa essere superiore alla domanda cioè

$$\sum_{i=1}^m a_i \geq \sum_{j=1}^n b_j. \quad (2.14)$$

In tal caso, possono essere ammesse giacenze nelle origini e/o nelle destinazioni; se si accetta di avere giacenze nelle origini, allora i vincoli di origine diventano

$$\sum_{j=1}^n x_{ij} \leq a_i \quad i = 1, \dots, m;$$

se si accetta di avere giacenze nelle destinazioni, allora i vincoli di destinazione diventano

$$\sum_{i=1}^m x_{ij} \geq b_j \quad j = 1, \dots, n.$$

nel caso in cui vale la (2.14), per porre il problema dei trasporti nella sua formulazione classica, cioè con vincoli di uguaglianza, si può introdurre una destinazione fittizia che abbia una richiesta pari a

$$\sum_{i=1}^m a_i - \sum_{j=1}^n b_j$$

ponendo uguale a zero il costo per raggiungere questa destinazione fittizia da qualsiasi origine.

Altri esempi di modelli di trasporto

Esempio 2.7.17 Un'industria produce un preparato chimico utilizzando due impianti di produzione **I1**, **I2**. Da questi impianti tutto il preparato chimico prodotto viene trasportato in due magazzini **M1**, **M2** che si trovano in differenti località. In questi magazzini una parte del preparato è venduta all'ingrosso direttamente, un'altra parte viene inviata a quattro centri di distribuzione **D1**, **D2**, **D3**, **D4** che effettuano la vendita al minuto. Questi centri necessitano ripetutamente di almeno 150, 190, 220, 170 quintali di preparato chimico che vendono rispettivamente a Lit.350000, 280000, 200000, 270000 al quintale. La tabella che segue riporta i costi (in migliaia di lire) necessari per trasportare un quintale di preparato da ciascun impianto a ciascun magazzino.

	M1	M2
I1	21	25
I2	27	22

Nella seguente tabella si riportano i costi (in migliaia di lire) necessari per trasportare un quintale di preparato chimico da ciascun magazzino a ciascun centro di distribuzione.

	D1	D2	D3	D4
M1	33	31	36	30
M2	27	30	28	31

L'impianto di produzione **I1** può fabbricare al più 3000 quintali di preparato, l'impianto **I2** può fabbricare al più 2000 quintali di preparato. I prezzi di vendita all'ingrosso effettuati presso i magazzini **M1** e

M2 sono rispettivamente di Lit. 150000 e 170000 al quintale. Per ragioni commerciali i quantitativi di preparato chimico venduti all'ingrosso in ciascun magazzino devono essere pari ad almeno 500 quintali ed inoltre tutto il preparato contenuto nei magazzini dovrà essere o venduto o trasportato ai centri di distribuzione per non avere rimanenze non vendute. Costruire un modello lineare che permetta di determinare le quantità di preparato chimico che devono essere prodotte in ciascun impianto e come esse devono essere ripartite tra i magazzini e i centri di distribuzione in modo da massimizzare il profitto netto complessivo.

Formulazione.

Si tratta di un problema di pianificazione industriale che unisce ad un problema di trasporti dagli impianti di produzione ai magazzini e dai magazzini ai centri di distribuzione, un problema di allocazione ottima di risorse.

– *Variabili.* Si introducono le variabili x_{ij} , $i, j = 1, 2$ per rappresentare la quantità (in quintali) di preparato chimico da produrre e quindi da trasportare dall'impianto **Ii** al magazzino **Mj**. Inoltre si introducono le variabili y_{kh} , $k = 1, 2$, $h = 1, 2, 3, 4$, per rappresentare la quantità (in quintali) di preparato chimico da trasportare dal magazzino **Mk** al centro di distribuzione **Dh**. Infine si devono introdurre due variabili z_1, z_2 per rappresentare la quantità (in quintali) di preparato chimico venduto all'ingrosso rispettivamente nel magazzino **M1** e **M2**.

– *Funzione obiettivo.* È data dal profitto netto, quindi dalla differenza tra ricavo ottenuto dalla vendita (presso i centri di distribuzione e all'ingrosso presso i magazzini) e le spese complessive dei trasporti. Il ricavo è dato da

$$350(y_{11} + y_{21}) + 280(y_{12} + y_{22}) + 200(y_{13} + y_{23}) + 270(y_{14} + y_{24}) + 150z_1 + 170z_2$$

mentre la spesa complessiva dei trasporti (dagli impianti ai magazzini e dai magazzini ai centri di distribuzione) è data

$$21x_{11} + 25x_{12} + 27x_{21} + 22x_{22} + 33y_{11} + 31y_{12} + 36y_{13} + 30y_{14} + 27y_{21} + 30y_{22} + 28y_{23} + 31y_{24}.$$

La funzione obiettivo sarà quindi data dalla differenza di queste due espressioni.

– *Vincoli.* Si hanno i seguenti vincoli

- vincoli dovuti alla capacità massima produttiva dei due impianti

$$\begin{aligned} x_{11} + x_{12} &\leq 3000 \\ x_{21} + x_{22} &\leq 2000 \end{aligned}$$

- vincoli dovuti alle richieste dei centri di distribuzione

$$\begin{aligned} y_{11} + y_{21} &\geq 150 \\ y_{12} + y_{22} &\geq 190 \\ y_{13} + y_{23} &\geq 220 \\ y_{14} + y_{24} &\geq 170 \end{aligned}$$

- vincoli derivanti dal quantitativo minimo di preparato che deve essere venduto all'ingrosso nei magazzini

$$z_1 \geq 500, \quad z_2 \geq 500$$

- vincoli dovuti al fatto che tutto il preparato contenuto nei magazzini dovrà essere o venduto all'ingrosso oppure trasportato ai centri di distribuzione in modo da non avere rimanenze non vendute; questi vincoli si esprimono imponendo che in ciascun magazzino la quantità di preparato chimico che arriva trasportato dagli impianti sia uguale alla somma del quantitativo di preparato trasportato dal magazzino ai centri di distribuzione e del quantitativo di preparato venduto all'ingrosso direttamente nel magazzino. Si hanno quindi i vincoli

$$\begin{aligned} x_{11} + x_{21} &= y_{11} + y_{12} + y_{13} + y_{14} + z_1 \\ x_{12} + x_{22} &= y_{21} + y_{22} + y_{23} + y_{24} + z_2 \end{aligned}$$

- vincoli di non negatività su tutte le variabili

$$x_{ij} \geq 0, \quad i, j = 1, 2, \quad y_{kh} \geq 0, \quad k = 1, 2, \quad h = 1, 2, 3, 4 \quad z_1 \geq 0, z_2 \geq 0.$$

Quindi la formulazione completa sarà

$$\left\{ \begin{array}{l} \max \left(350(y_{11} + y_{21}) + 280(y_{12} + y_{22}) + 200(y_{13} + y_{23}) + 270(y_{14} + y_{24}) + 150z_1 + 170z_2 + \right. \\ \left. - (21x_{11} + 25x_{12} + 27x_{21} + 22x_{22} + 33y_{11} + 31y_{12} + 36y_{13} + 30y_{14} + 27y_{21} + 30y_{22} + 28y_{23} + 31y_{24}) \right) \\ x_{11} + x_{12} \leq 3000 \\ x_{21} + x_{22} \leq 2000 \\ y_{11} + y_{21} \geq 150 \\ y_{12} + y_{22} \geq 190 \\ y_{13} + y_{23} \geq 220 \\ y_{14} + y_{24} \geq 170 \\ x_{11} + x_{21} = y_{11} + y_{12} + y_{13} + y_{14} + z_1 \\ x_{12} + x_{22} = y_{21} + y_{22} + y_{23} + y_{24} + z_2 \\ z_1 \geq 500 \\ z_2 \geq 500 \\ x_{ij} \geq 0, \quad i, j = 1, 2, \\ y_{kh} \geq 0, \quad k = 1, 2, \quad h = 1, 2, 3, 4 \end{array} \right.$$

I vincoli di non negatività delle variabili z_i sono ovviamente implicati dai vincoli $z_1, z_2 \geq 500$.

Capitolo 3

La dualità nella Programmazione Lineare

In questo capitolo¹ verrà introdotto un concetto di fondamentale importanza sia per l'analisi dei problemi di Programmazione Lineare, sia per lo sviluppo di algoritmi risolutivi, sia per l'interpretazione dei risultati da essi prodotti. Si tratta del concetto di *dualità* che è legato alla possibilità di associare ad ogni problema di Programmazione Lineare un altro problema di Programmazione Lineare (chiamato *problema duale*) che ha la particolarità di permettere la deduzione di importanti proprietà del problema originario; quest'ultimo viene chiamato *problema primale*. In particolare, da un punto di vista computazionale, sarà possibile risolvere il problema duale al posto del problema primale e ottenere comunque le informazioni desiderate sulla soluzione ottima del problema primale.

Verrà inoltre esaminata un'interpretazione economica delle variabili del problema duale e verrà illustrato, attraverso gli esempi di un problema di allocazione ottima di risorse, di un problema di miscelazione e di un problema di trasporti, come il problema duale può avere interessanti interpretazioni applicative.

Si osservi che la teoria della dualità sarà esaminata in relazione a problemi di Programmazione Lineare, ma essa può essere estesa anche al caso di problemi di Programmazione Non Lineare.

3.1 Teoria della dualità

Un modo di introdurre il problema duale può essere quello di far riferimento alla possibilità di determinare delle stime inferiori del valore ottimo della funzione obiettivo di un problema di Programmazione Matematica (in forma di problema di minimizzazione). In particolare, dato un problema di Programmazione Lineare

$$\begin{cases} \min c^T x \\ Ax \geq b \end{cases} \quad (3.1)$$

(dove $c \in \mathbb{R}^n$, $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ e $b \in \mathbb{R}^m$) se si dispone di un punto ammissibile \tilde{x} può risultare molto interessante capire se esso rappresenta una buona stima di un punto di ottimo senza risolvere il problema esattamente. Infatti se si conoscesse una buona stima ϕ del valore ottimo potremmo capire la "bontà" del punto \tilde{x} confrontando il valore $c^T \tilde{x}$ con la stima ϕ .

A questo scopo, sia x^* una soluzione ottima del problema (3.1); allora per ogni $u \in \mathbb{R}^m$, $u \geq 0$, poiché $Ax^* \geq b$ risulta

$$\begin{aligned} c^T x^* &\geq c^T x^* + u^T (b - Ax^*) \\ &= b^T u + (c^T - u^T A)x^*. \end{aligned} \quad (3.2)$$

¹Questo capitolo è tratto da: Stefano Lucidi, Massimo Roma, "Lezioni di Ricerca Operativa", Università di Roma "La Sapienza", Corso di Laurea in Ingegneria Informatica, A.A.2000-2001.

Se si restringe ulteriormente la scelta del vettore u si può eliminare la presenza di x^* dal membro di destra della (3.2). Infatti, per ogni $u \in \mathbb{R}^n$, $u \geq 0$, tale che

$$A^T u = c,$$

la (3.2) si riduce a $b^T u$ e quindi risulta

$$c^T x^* \geq b^T u. \quad (3.3)$$

Per rendere quanto più possibile stringente la stima della limitazione inferiore del valore ottimo della funzione obiettivo $c^T x^*$ si può rendere quanto più possibile grande il termine di destra della diseguaglianza (3.3), cioè si può massimizzare la quantità $b^T u$ al variare del vettore $u \in \mathbb{R}^n$, tra tutti i vettori che soddisfano $A^T u = c$, $u \geq 0$. Più formalmente si ha

$$c^T x^* \geq b^T u^*$$

dove u^* è soluzione del problema

$$\begin{cases} \max b^T u \\ A^T u = c \\ u \geq 0 \end{cases} \quad (3.4)$$

Questo problema viene definito *problema duale* del problema dato (3.1) che viene denominato *problema primale*.

La possibilità di definire un problema duale non è legato al fatto che il problema dato abbia solo vincoli di diseguaglianza e variabili non vincolate in segno come è il problema (3.1). Infatti, si consideri un problema Programmazione Lineare scritto nella forma più generale possibile cioè nella forma

$$(P) \quad \begin{cases} \min c^T x + d^T y \\ Cx + Dy = h \\ Ex + Fy \geq g \\ x \geq 0 \end{cases}$$

con $x \in \mathbb{R}^p$, $c \in \mathbb{R}^p$, $y \in \mathbb{R}^{n-p}$, $d \in \mathbb{R}^{n-p}$; C matrice $q \times p$, D matrice $q \times (n-p)$ e $h \in \mathbb{R}^q$; E matrice $(m-q) \times p$, F matrice $(m-q) \times (n-p)$ e $g \in \mathbb{R}^{m-q}$. La notazione in cui è scritto questo generico problema di Programmazione Lineare (P) è tale da evidenziare separatamente gli elementi che intervengono nella formulazione: le variabili sono partizionate nella variabili x vincolate in segno e y non vincolate in segno e corrispondentemente anche i coefficienti di costo della funzione obiettivo sono partizionati in c e d ; i vincoli sono scritti suddividendo quelli di uguaglianza e quelli di diseguaglianza (nella forma di maggiore o uguale).

Per costruire il problema duale del problema (P) è sufficiente ripetere i ragionamenti fatti in relazione al problema (3.1) dopo aver trasformato il problema (P) in un problema equivalente con soli vincoli di diseguaglianza cioè nella forma (3.1). A tale scopo riscriviamo il problema (P) nella forma equivalente

$$\begin{cases} \min c^T x + d^T y \\ Cx + Dy \geq h \\ -Cx - Dy \geq -h \\ Ex + Fy \geq g \\ I_p x \geq 0 \end{cases}$$

dove I_p è la matrice identità di ordine p . I vincoli di questo problema possono essere scritti in forma matriciale

$$\begin{pmatrix} C & D \\ -C & -D \\ E & F \\ I_p & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \geq \begin{pmatrix} h \\ -h \\ g \\ 0 \end{pmatrix},$$

quindi il problema (P) è stato ricondotto nella forma (3.1). Siamo quindi in grado di scrivere il duale di questo problema nella forma (3.4) cioè:

$$\begin{cases} \max h^T t - h^T w + g^T v \\ C^T t - C^T w + E^T v + I_p z = c \\ D^T t - D^T w + F^T v = d \\ t \geq 0, w \geq 0, v \geq 0, z \geq 0 \end{cases} \quad \text{che è equivalente a} \quad \begin{cases} \max h^T(t-w) + g^T v \\ C^T(t-w) + E^T v + I_p z = c \\ D^T(t-w) + F^T v = d \\ t \geq 0, w \geq 0, v \geq 0, z \geq 0. \end{cases}$$

Eliminando la variabile z ed effettuando il cambio di variabili $t - w = u$ si ottiene il seguente problema nelle variabili (u, v) , con u non vincolata in segno e $v \geq 0$:

$$(D) \quad \begin{cases} \max h^T u + g^T v \\ C^T u + E^T v \leq c \\ D^T u + F^T v = d \\ v \geq 0 \end{cases}$$

con $u \in \mathbb{R}^q$ e $v \in \mathbb{R}^{m-q}$.

Il problema (D) è il *problema duale* del problema (P) che viene detto *problema primale*. Le variabili (x, y) sono dette *variabili primali*; le variabili (u, v) sono dette *variabili duali*. I due problemi (P) e (D) vengono chiamati *coppia primale-duale*. In maniera del tutto simmetrica, il problema (P) risulterà il problema duale del problema (D).

Dall'osservazione dei due problemi (P) e (D) si deducono facilmente le proprietà fondamentali di una coppia primale-duale; innanzitutto un problema è di minimizzazione mentre l'altro è di massimizzazione. Inoltre poiché la matrice dei coefficienti dei vincoli di un problema si ottiene trasponendo quella dell'altro, si ha che ad ogni variabile di un problema corrisponde un vincolo nell'altro. Si osserva inoltre uno scambio tra i termini noti di un problema e i coefficienti della funzione obiettivo dell'altro.

Queste proprietà possono essere così schematicamente riassunte:

- il problema duale di un problema di minimizzazione è un problema di massimizzazione e simmetricamente, il problema duale di un problema di massimizzazione è un problema di minimizzazione;
- ad ogni vincolo di uguaglianza del problema primale è associata una variabile nel problema duale non vincolata in segno che ha come coefficiente nella funzione obiettivo duale il termine noto del vincolo primale associato;
- ad ogni vincolo di diseguaglianza (di maggiore o uguale) del problema primale è associata una variabile nel problema duale vincolata in segno che ha come coefficiente nella funzione obiettivo duale il termine noto del vincolo primale associato;
- ad ogni variabile vincolata in segno del problema primale è associato un vincolo di diseguaglianza (di minore o uguale) del problema duale il cui termine noto è dato dal coefficiente della funzione obiettivo primale;
- ad ogni variabile non vincolata in segno del problema primale è associato un vincolo di uguaglianza del problema duale il cui termine noto è dato dal coefficiente della funzione obiettivo primale.

Queste corrispondenze possono essere riassunte nella tabella che segue dove gli insieme I, J, M e N sono insiemi di indici:

	PRIMALE	DUALE	
	$\min c^T x$	$\max b^T u$	
VINCOLI	$= b_i, \ i \in I$ $\geq b_i, \ i \in J$	$u_i, \ i \in I, \text{ libere}$ $u_i, \ i \in J, u_i \geq 0$	VARIABILI
VARIABILI	$x_j \geq 0, \ j \in M$ $x_j, \ j \in N \text{ libere}$	$\leq c_j, \ j \in M$ $= c_j, \ j \in N$	VINCOLI

Un importante caso di coppia primale–duale è ottenuto considerando nel problema primale solamente i vincoli di diseguaglianza e solamente variabili vincolate in segno. In questo caso si ottiene la seguente coppia di problemi:

$$(P) \quad \begin{cases} \min c^T x \\ Ax \geq b \\ x \geq 0 \end{cases} \quad (D) \quad \begin{cases} \max b^T u \\ A^T u \leq c \\ u \geq 0 \end{cases}$$

con $x \in \mathbb{R}^n$, $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $u \in \mathbb{R}^m$ e $A \in \mathbb{R}^{m \times n}$. Tale coppia di problemi viene detta *coppia primale–duale simmetrica*. A tale coppia particolare primale–duale si farà spesso riferimento nel seguito anche in relazione a considerazioni riguardanti l’interpretazione della dualità per problemi di Programmazione Lineare che sono formulazioni di importanti classi di modelli lineari.

Seguono ora alcuni esempi di costruzione del problema duale di un problema di Programmazione Lineare assegnato.

Esempio 3.1.1 *Si consideri il seguente problema di Programmazione Lineare*

$$\begin{cases} \min 2x_1 + 3x_2 + 4x_3 + x_4 \\ x_1 - 5x_3 + 2x_4 \geq 7 \\ 2x_1 + 4x_2 - 6x_3 \geq 9. \end{cases}$$

Il problema duale associato è

$$\begin{cases} \max 7u_1 + 9u_2 \\ u_1 + 2u_2 = 2 \\ 4u_2 = 3 \\ -5u_1 - 6u_2 = 4 \\ 2u_1 = 1 \\ u_1 \geq 0, u_2 \geq 0. \end{cases}$$

Esempio 3.1.2 *Si consideri il seguente problema di Programmazione Lineare*

$$\begin{cases} \max 4x_1 + 3x_2 + 2x_3 \\ x_1 + 2x_2 + 3x_3 \leq 8 \\ 2x_1 - x_3 \leq 7 \\ 3x_1 + 4x_2 - x_3 \leq 5 \\ x_2 + x_3 \leq 6 \\ x_2 \geq 0 \end{cases}$$

Il problema duale è il seguente problema di minimizzazione

$$\begin{cases} \min 8u_1 + 7u_2 + 5u_3 + 6u_4 \\ u_1 + 2u_2 + 3u_3 = 4 \\ 2u_1 + 4u_3 + u_4 \geq 3 \\ 3u_1 - u_2 - u_3 + u_4 = 2 \\ u_1 \geq 0, u_2 \geq 0, u_3 \geq 0, u_4 \geq 0. \end{cases}$$

Esempio 3.1.3 Si consideri il seguente problema di Programmazione Lineare

$$\begin{cases} \min 2x_1 - 3x_2 + x_3 \\ 3x_1 + x_2 + 5x_3 \geq 7 \\ x_1 + x_2 - 6x_3 \leq 9 \\ 4x_1 - x_2 - 2x_3 = 8 \\ x_1 \geq 0, x_2 \geq 0. \end{cases}$$

Dopo aver riscritto il secondo vincolo come $-x_1 - x_2 + 6x_3 \geq -9$ si può formulare facilmente il problema duale associato

$$\begin{cases} \max 7u_1 - 9u_2 + 8u_3 \\ 3u_1 - u_2 + 4u_3 \leq 2 \\ u_1 - u_2 - u_3 \leq -3 \\ 5u_1 + 6u_2 - 2u_3 = 1 \\ u_1 \geq 0, u_2 \geq 0 \end{cases}$$

3.1.1 Risultati fondamentali della teoria della dualità

Una coppia primale-duale di problemi di Programmazione Lineare gode di proprietà importanti sia dal punto di vista teorico sia da quello pratico. I risultati che verranno di seguito analizzati valgono per una generica coppia primale-duale, ma per brevità e semplicità di notazioni essi sono enunciati e dimostrati facendo riferimento alla coppia primale duale simmetrica

$$(P) \quad \begin{cases} \min c^T x \\ Ax \geq b \\ x \geq 0 \end{cases} \quad (D) \quad \begin{cases} \max b^T u \\ A^T u \leq c \\ u \geq 0 \end{cases} \quad (3.5)$$

con $x \in \mathbb{R}^n$, $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $u \in \mathbb{R}^m$ e $A \in \mathbb{R}^{m \times n}$.

Il primo risultato che si vuole analizzare discende in maniera naturale dal modo in cui è stato introdotto il problema duale.

Teorema 3.1.4 – TEOREMA DELLA DUALITÀ DEBOLE

Per ogni soluzione ammissibile \bar{x} del problema primale (P) ed ogni soluzione ammissibile \bar{u} del problema duale (D) si ha

$$b^T \bar{u} \leq c^T \bar{x}$$

cioè il valore della funzione obiettivo duale in \bar{u} è minore o uguale del valore della funzione obiettivo primale in \bar{x} .

Dim.: Poiché \bar{x} è soluzione ammissibile per il problema primale (P), deve risultare $A\bar{x} \geq b$; tale diseguaglianza può essere riscritta nella forma $\bar{x}^T A^T \geq b^T$. Effettuando il prodotto scalare di entrambi i membri di questa diseguaglianza per \bar{u} (che è non negativo perché \bar{u} è ammissibile per il problema duale) si ottiene

$$\bar{x}^T A^T \bar{u} \geq b^T \bar{u}. \quad (3.6)$$

In modo del tutto analogo, poiché \bar{u} è soluzione ammissibile per il problema duale (D), deve risultare $A^T \bar{u} \leq c$ che può essere riscritta nella forma $\bar{u}^T A \leq c^T$. Effettuando il prodotto scalare di entrambi i membri di questa diseguaglianza per \bar{x} (che è non negativo perché \bar{u} è ammissibile per il problema primale) si ottiene

$$\bar{u}^T A \bar{x} \leq c^T \bar{x}. \quad (3.7)$$

Dalla (3.6) e dalla (3.7) si ottiene

$$b^T \bar{u} \leq \bar{x}^T A^T \bar{u} = \bar{u}^T A \bar{x} \leq c^T \bar{u}$$

che è quanto si voleva dimostrare. \square

Da questo importante teorema discendono due conseguenze immediate che sono riportate nei corollari che seguono.

Corollario 3.1.5 *Se \bar{x} è una soluzione ammissibile del problema primale (P) e \bar{u} una soluzione ammissibile del problema duale (D) tali che*

$$c^T \bar{x} = b^T \bar{u} \quad (3.8)$$

allora \bar{x} e \bar{u} sono soluzioni ottime rispettivamente per il problema primale (P) e per il problema duale (D).

Dim.: La dimostrazione procede per assurdo. Infatti, supponiamo per assurdo che \bar{u} non sia soluzione ottima del problema duale (D). Questo significa che esiste una soluzione ammissibile del problema duale (D) che indichiamo con \tilde{u} tale che

$$b^T \tilde{u} > b^T \bar{u}.$$

Per la (3.8) si avrebbe

$$b^T \tilde{u} > b^T \bar{u} = c^T \bar{x}$$

contraddicendo il Teorema 3.1.4 (Teorema della Dualità debole).

Simmetricamente, si ottiene una contraddizione supponendo che \bar{x} non sia soluzione ottima del problema primale (P). \square

Corollario 3.1.6 *Se il problema primale (P) è illimitato (inferiormente) allora il problema duale (D) è inammissibile. Viceversa, se il problema duale è illimitato (superiormente) allora il problema primale è inammissibile.*

Dim.: Supponiamo che il problema primale (P) sia illimitato e che, per assurdo, il problema duale (D) non sia inammissibile, cioè che esista una soluzione ammissibile \bar{u} del problema duale (D). Per il Teorema 3.1.4 (Teorema della Dualità debole), per ogni soluzione ammissibile x del problema primale (P) deve valere

$$b^T \bar{u} \leq c^T x \quad (3.9)$$

e, poiché si è supposto che il problema primale (P) è illimitato inferiormente, con la (3.9) si ottiene una contraddizione al fatto che \bar{u} è una soluzione ammissibile del problema duale (D). In modo del tutto simmetrico si dimostra il viceversa. \square

Oltre alle proprietà fino ad ora esaminate, nel caso di problemi di Programmazione Lineare è vera anche un'altra proprietà che è solitamente chiamata *dualità forte* che fornisce una caratterizzazione importante nel caso in cui esista soluzione ottima di uno dei due problemi della coppia primale duale. Questa proprietà è descritta nel seguente teorema che si riporta senza dimostrazione in quanto la dimostrazione richiederebbe la conoscenza di ulteriori risultati preliminari la cui trattazione esula dallo scopo di queste note.

Teorema 3.1.7 – TEOREMA DELLA DUALITÀ FORTE

Se il problema primale (P) ammette una soluzione ottima x^ allora anche il problema duale (D) ammette una soluzione ottima u^* . Simmetricamente, se il problema duale (D) ammette una soluzione ottima u^* allora anche il problema primale (P) ammette una soluzione ottima x^* . Inoltre i valori delle funzioni obiettivo dei due problemi all'ottimo sono uguali cioè risulta*

$$c^T x^* = b^T u^*.$$

Siamo ora in grado di formulare delle *condizioni di ottimalità* in riferimento alla coppia primale-duale simmetrica; esse costituiscono una caratterizzazione analitica delle soluzioni ottime del problema primale e del problema duale; enunciamo queste condizioni nel seguente teorema che discende immediatamente dai risultati già ottenuti:

Teorema 3.1.8 – CONDIZIONI DI OTTIMALITÀ

Siano dati $\bar{x} \in \mathbb{R}^n$ e $\bar{u} \in \mathbb{R}^m$. Allora \bar{x} e \bar{u} sono soluzioni ottime rispettivamente per il problema primale (P) e per il problema duale (D) se e solo se valgono le seguenti condizioni

- (i) $A\bar{x} \geq b, \bar{x} \geq 0$ *(ammissibilità primale)*
- (ii) $A^T \bar{u} \leq c, \bar{u} \geq 0$ *(ammissibilità duale)*
- (iii) $c^T \bar{x} = b^T \bar{u}$. *(coincidenza dei valori delle funzioni obiettivo)*

Sulla base dei risultati fino ad ora esaminati si evince che data un coppia primale-duale di problemi di Programmazione Lineare possono verificarsi le seguenti situazioni: o entrambi ammettono soluzione

ottima, oppure se uno è illimitato l'altro è inammissibile, oppure sono entrambi inammissibili. Queste possibilità sono riportate schematicamente nella tabella che segue.

		DUALE		
		OTTIMO FINITO	ILLIMITATO SUPERIOR.	INAMMISSIBILE
PRIMALE	OTTIMO FINITO	SI	NO	NO
	ILLIMITATO INFERIOR.	NO	NO	SI
	INAMMISSIBILE	NO	SI	SI

Ribadiamo che tutti i risultati ottenuti nel caso di una coppia primale–duale simmetrica (3.5) si estendono facilmente ad una qualsiasi coppia primale–duale.

Seguono alcuni esempi che illustrano i risultati teorici ora esaminati.

Esempio 3.1.9 *Si consideri il problema di Programmazione Lineare*

$$\begin{cases} \min x_1 + 3x_2 \\ x_1 + 4x_2 \geq 24 \\ 5x_1 + x_2 \geq 25 \\ x_1 \geq 0, x_2 \geq 0 \end{cases}$$

Geometricamente si ricava facilmente che questo problema ammette soluzione ottima nel punto $(x_1, x_2) = (4, 5)$ e il valore ottimo della funzione obiettivo è pari a 19. Se si considera il problema duale

$$\begin{cases} \max 24u_1 + 25u_2 \\ u_1 + 5u_2 \leq 1 \\ 4u_1 + u_2 \leq 3 \\ u_1 \geq 0, u_2 \geq 0; \end{cases}$$

si ricava facilmente (geometricamente) che, in accordo con quanto previsto dal Teorema della Dualità Forte, anche questo problema ammette soluzione ottima — nel punto $(u_1, u_2) = \left(\frac{14}{19}, \frac{1}{19}\right)$ — e il valore ottimo della funzione obiettivo vale 19.

Esempio 3.1.10 *Si consideri il problema di Programmazione Lineare*

$$\begin{cases} \max 2x_1 + 3x_2 \\ -2x_1 + x_2 \leq 3 \\ -\frac{1}{2}x_1 + x_2 \leq 6 \\ x_1 \geq 0, x_2 \geq 0; \end{cases}$$

Geometricamente si ricava che il problema è illimitato superiormente. Quindi, per l'analisi teorica vista deve risultare che il suo duale è inammissibile. E infatti se si considera il problema duale associato

$$\begin{cases} \min 3u_1 + 6u_2 \\ -2u_1 - \frac{1}{2}u_2 \geq 2 \\ u_1 + u_2 \geq 3 \\ u_1 \geq 0, u_2 \geq 0 \end{cases}$$

si vede facilmente che questo problema non ammette soluzioni ammissibili.

Esercizio 3.1.11 Risolvere graficamente i problemi di Programmazione Lineare proposti nell'Esempio 3.1.9 e nell'Esempio 3.1.10 e verificare le conclusioni tratte negli esempi.

3.1.2 Condizioni di complementarità

Un'ulteriore proprietà della coppia primale-duale è la cosiddetta *complementarietà*. Tale proprietà è di fondamentale importanza anche negli sviluppi algoritmici in quanto è alla base dei cosiddetti *metodi primali duali* per soluzione dei problemi di Programmazione Lineare.

Anche in questo caso si farà riferimento alla coppia primale-duale simmetrica

$$(P) \quad \begin{cases} \min c^T x \\ Ax \geq b \\ x \geq 0 \end{cases} \quad (D) \quad \begin{cases} \max b^T u \\ A^T u \leq c \\ u \geq 0 \end{cases}$$

con $x \in \mathbb{R}^n$, $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $u \in \mathbb{R}^m$ e $A \in \mathbb{R}^{m \times n}$, ma tutti i risultati sono validi nel caso di una qualsiasi coppia primale-duale.

In particolare, riportiamo di seguito un teorema fondamentale che caratterizza ulteriormente le soluzioni ottime di una coppia primale-duale di problemi di Programmazione Lineare.

Teorema 3.1.12 Sia \bar{x} un punto ammissibile del problema primale (P) e sia \bar{u} un punto ammissibile del problema duale (D). Allora \bar{x} e \bar{u} sono soluzioni ottime rispettivamente del problema primale (P) e del problema duale (D) se e solo se soddisfano le seguenti condizioni:

$$\bar{u}^T(Ax - b) = 0 \quad (3.10)$$

$$\bar{x}^T(c - A^T\bar{u}) = 0. \quad (3.11)$$

Dim.: Iniziamo supponendo che \bar{x} e \bar{u} sono soluzioni ammissibili rispettivamente per il problema primale (P) e per il problema duale (D) e che valgano le (3.10) e (3.11), e mostrando che allora \bar{x} e \bar{u} sono soluzioni ottime rispettivamente del problema primale (P) e del problema duale (D). Infatti se valgono le (3.10) e (3.11) allora dalla (3.10) si ha $\bar{u}^T A \bar{x} = \bar{u}^T b$ e dalla (3.11) si ha $\bar{x}^T c = \bar{x}^T A^T \bar{u}$ da cui

$$b^T \bar{u} = \bar{u}^T b = \bar{u}^T A \bar{x} = \bar{x}^T A^T \bar{u} = \bar{x}^T c = c^T \bar{x}.$$

Quindi risulta $b^T \bar{u} = c^T \bar{x}$ e dal Teorema 3.1.8 si ha che \bar{x} e \bar{u} sono soluzioni ottime rispettivamente del problema primale (P) e del problema duale (D).

Supponiamo ora che \bar{x} e \bar{u} siano soluzioni ottime rispettivamente del problema primale (P) e del problema duale (D) e dimostriamo allora che devono valere le condizioni (3.10) e (3.11). Se \bar{x} e \bar{u} sono

soluzioni ottime per i rispettivi problemi, dal Teorema 3.1.8 si ha che deve valere $c^T \bar{x} = b^T \bar{u}$. Ora, poiché \bar{x} e \bar{u} sono ammissibili rispettivamente per il problema (P) e per il problema (D) si ha

$$A\bar{x} \geq b, \quad \bar{x} \geq 0 \quad (3.12)$$

$$A^T \bar{u} \leq c, \quad \bar{u} \geq 0. \quad (3.13)$$

Effettuando il prodotto scalare di entrambi i membri della (3.12) per \bar{u} e di entrambi i membri della (3.13) per \bar{x} si ha

$$\begin{aligned} \bar{u}^T A\bar{x} &\geq \bar{u}^T b \\ \bar{x}^T A^T \bar{u} &\leq \bar{x}^T c. \end{aligned}$$

Da queste due diseguaglianze segue che

$$\bar{u}^T b \leq \bar{u}^T A\bar{x} = \bar{x}^T A^T \bar{u} \leq \bar{x}^T c \quad (3.14)$$

e poiché stiamo supponendo $c^T \bar{x} = b^T \bar{u}$ allora la (3.14) diventa

$$\bar{u}^T b = \bar{u}^T A\bar{x} = \bar{x}^T A^T \bar{u} = \bar{x}^T c$$

dalla quale discende immediatamente

$$\begin{aligned} \bar{u}^T (A\bar{x} - b) &= 0 \\ \bar{x}^T (c - A^T) &= 0 \end{aligned}$$

che sono le condizioni (3.10) e (3.11). \square

Le condizioni (3.10) e (3.11) vengono chiamate *condizioni di complementarietà* e costituiscono, di fatto, delle condizioni di ottimalità per i problemi della coppia primale-duale.

Dal precedente risultato seguono facilmente i seguenti corollari.

Corollario 3.1.13 *Sia \bar{x} un punto ammissibile del problema primale (P) e sia \bar{u} un punto ammissibile del problema duale (D). Allora \bar{x} e \bar{u} sono soluzioni ottime rispettivamente del problema primale (P) e del problema duale (D) se e solo se soddisfano le seguenti condizioni:*

$$\bar{u}_j (A^T \bar{x} - b)_j = 0 \quad (3.15)$$

$$\bar{x}_i (c - A^T \bar{u})_i = 0. \quad (3.16)$$

$$i = 1, \dots, n, \quad j = 1, \dots, m.$$

Dim.: Per il Teorema 3.1.12 \bar{x} e \bar{u} sono soluzioni ottime dei rispettivi problemi se e solo se i prodotti scalari

$$\begin{aligned} \bar{u}^T (A\bar{x} - b) \\ \bar{x}^T (c - A^T \bar{u}) \end{aligned}$$

sono nulli. Ora, poiché per l'ammissibilità primale risulta $A\bar{x} - b \geq 0$ e $\bar{x} \geq 0$ e per l'ammissibilità duale risulta $c - A^T \bar{u} \geq 0$ e $\bar{u} \geq 0$, si ha che affinché ciascuno dei prodotti scalari risulti nullo, ogni termine dei due prodotti deve essere nullo e quindi il corollario è dimostrato. \square

Il Corollario (3.1.13) può essere formulato in maniera del tutto equivalente nella seguente forma:

Corollario 3.1.14 *Sia \bar{x} un punto ammissibile del problema primale (P) e sia \bar{u} un punto ammissibile del problema duale (D). Allora \bar{x} e \bar{u} sono soluzioni ottime rispettivamente del problema primale (P) e del problema duale (D) se e solo se soddisfano le seguenti proprietà:*

- (i) *per ogni variabile del problema (P) che assume valore non nullo il corrispondente vincolo del problema duale (D) deve essere soddisfatto all'uguaglianza;*
- (ii) *per ogni variabile del problema (D) che assume valore non nullo il corrispondente vincolo del problema primale (P) è soddisfatto all'uguaglianza.*

Il Teorema 3.1.12 e i successivi corollari forniscono delle *condizioni di ottimalità* per i problemi Programmazione Lineare; essi infatti forniscono delle condizioni necessarie e sufficienti che caratterizzano analiticamente la soluzione ottima di un problema di Programmazione Lineare. Per completezza riportiamo queste condizioni (già dimostrate) formalizzate nel seguente teorema.

Teorema 3.1.15 CONDIZIONI DI OTTIMALITÀ

Due punti $\bar{x} \in \mathbb{R}^n$ e $\bar{u} \in \mathbb{R}^m$ sono soluzioni ottime rispettivamente del problema (P) e (D) se e solo se valgono le seguenti condizioni:

- (i) $A\bar{x} \geq b, \quad \bar{x} \geq 0 \quad (\text{ammissibilità primale})$
- (ii) $A^T\bar{u} \leq c, \quad \bar{u} \geq 0 \quad (\text{ammissibilità duale})$
- (iii) $\bar{u}^T(A\bar{x} - b) = 0 \quad (\text{condizioni di complementarità})$
- (iv) $\bar{x}^T(c - A^T\bar{u}) = 0.$

Ribadiamo che tutti i risultati fino ad ora enunciati e dimostrati nel caso di una coppia primale-duale simmetrica, valgono in generale per ogni coppia primale-duale.

Osservazione 3.1.16 Si osservi che in relazione alla coppia primale-duale

$$\begin{cases} \min c^T x \\ Ax \geq b \end{cases} \quad \begin{cases} \max b^T u \\ A^T u = c \end{cases}$$

le condizioni di complementarità si riducono alla sola condizione

$$u^T(Ax - b) = 0,$$

mentre per una coppia primale-duale

$$\begin{cases} \min c^T x \\ Ax = b \\ x \geq 0 \end{cases} \quad \begin{cases} \max b^T u \\ A^T u \leq c \end{cases}$$

le condizioni di complementarità si riducono alla sola condizione

$$x^T(c - A^T u) = 0.$$

Seguono alcuni esempi di applicazione della complementarità.

Esempio 3.1.17 *Si consideri il problema di Programmazione Lineare*

$$\begin{cases} \min 3x_1 + 2x_2 + x_3 + 4x_4 \\ x_1 - 3x_3 + 2x_4 \geq 5 \\ 2x_1 + x_2 - x_3 \geq 3 \end{cases}.$$

Il problema duale associato è

$$\begin{cases} \max 5u_1 + 3u_2 \\ u_1 + 2u_2 = 3 \\ u_2 = 2 \\ -3u_1 - u_2 = 1 \\ 2u_1 = 4 \\ u_1 \geq 0, u_2 \geq 0 \end{cases}$$

Poiché il duale ha solo vincoli di uguaglianza, le condizioni di complementarità si riducono a $u^T(Ax - b) = 0$ che in questo caso sono

$$\begin{aligned} u_1(x_1 - 3x_3 + 2x_4 - 5) &= 0 \\ u_2(2x_1 + x_2 - x_3 - 3) &= 0. \end{aligned}$$

Esempio 3.1.18 *Dato il problema di programmazione lineare*

$$\begin{cases} \min 2x_1 + 3x_2 + x_3 + x_4 \\ x_1 + x_2 + x_3 = 2 \\ 2x_1 + 3x_4 = 1 \\ x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0 \end{cases} \quad (3.17)$$

si consideri il punto $\bar{x} = (0, 0, 2, 1/3)$ soluzione ammissibile per il problema (3.17) e il punto $\bar{u} = (1, 1/3)$ soluzione ammissibile per il problema duale associato a (3.17). Attraverso le condizioni di complementarità si vuole verificare se \bar{x} è una soluzione ottima del problema del problema (3.17). Innanzitutto scriviamo il problema duale del problema dato; esso è

$$\begin{cases} \max 2u_1 + u_2 \\ u_1 + 2u_2 \leq 2 \\ u_1 \leq 3 \\ u_1 \leq 1 \\ 3u_2 \leq 1. \end{cases}$$

Poiché il problema (3.17) presenta solo vincoli di uguaglianza, le condizioni di complementarità si riducono a $x^T(c - A^T u) = 0$ che in questo caso sono

$$\begin{aligned} x_1(2 - u_1 - 2u_2) &= 0 \\ x_2(3 - u_1) &= 0 \\ x_3(1 - u_1) &= 0 \\ x_4(1 - 3u_2) &= 0 \end{aligned}$$

Sostituendo i valori delle soluzioni ammissibili \bar{x} , \bar{u} rispettivamente per il primale ed il duale, le condizioni di complementarità risultano verificate. Quindi la soluzione \bar{x} è effettivamente ottima per il primale e \bar{u} è ottima per il duale.

Esempio 3.1.19 Si consideri il problema di Programmazione Lineare

$$\begin{cases} \min c_1 x_1 + c_2 x_2 + c_3 x_3 \\ x_1 + 2x_2 + 2x_3 \leq 2 \\ x_1 + 4x_2 + 2x_3 \leq 3 \\ x_1 \geq 0, x_2 \geq 0, x_3 \geq 0 \end{cases}$$

con $c_1 \in \mathbb{R}$, $c_2 \in \mathbb{R}$, $c_3 \in \mathbb{R}$. Utilizzando la teoria della dualità, si vuole stabilire se esistono valori (non tutti nulli) di c_1, c_2, c_3 tali che il punto $\bar{x} = (0, 0, 1/2)^T$ sia una soluzione ottima del problema.

Innanzitutto scriviamo il problema duale associato che è

$$\begin{cases} \max -2u_1 - 3u_3 \\ -u_1 - u_2 \leq c_1 \\ -2u_1 - 4u_2 \leq c_2 \\ -2u_1 - 2u_2 \leq c_3 \\ u_1 \geq 0, u_2 \geq 0. \end{cases}$$

e le condizioni di complementarità

$$\begin{aligned} u_1(-2 + x_1 + 2x_2 + 2x_3) &= 0 \\ u_2(-3 + x_1 + 4x_2 + 2x_3) &= 0 \\ x_1(c_1 + u_1 + u_2) &= 0 \\ x_2(c_2 + 2u_1 + 4u_2) &= 0 \\ x_3(c_3 + 2u_1 + 2u_2) &= 0 \end{aligned}$$

Sostituendo il punto \bar{x} affinché siano soddisfatte tutte le equazioni deve essere

$$\bar{u}_1 = 0, \quad \bar{u}_2 = 0, \quad \frac{1}{2}(c_3 + 2\bar{u}_1 + 2\bar{u}_2) = 0$$

e quindi $c_3 = 0$ (dove \bar{u} è soluzione ottima del problema duale). Quindi le condizioni di complementarità sono soddisfatte per qualunque c_1 e c_2 e $c_3 = 0$. Quindi il punto dato \bar{x} è soluzione ottima del problema per qualsiasi valore di c_1 e c_2 e $c_3 = 0$.

3.2 Interpretazione della Dualità

Nei modelli reali le variabili (primarie) possono rappresentare, ad esempio, livelli di produzione e i coefficienti di costo possono essere associati ai profitti ricavati dalla vendita dei prodotti. Quindi la funzione obiettivo di un problema primale indica direttamente come un aumento della produzione può influenzare il profitto. Sempre in relazione, ad esempio, ad un modello per la pianificazione della produzione, i vincoli di un problema (primale) possono rappresentare una limitazione dovuta alla limitata disponibilità delle risorse; ora, un aumento della disponibilità delle risorse può consentire un aumento della produzione e quindi anche del profitto, ma questa relazione tra aumento della disponibilità delle risorse e aumento del profitto non si deduce facilmente dal problema formulato (il problema primale). Uno dei possibili usi della dualità è quello di rendere esplicito l'effetto dei cambiamenti nei vincoli (ad esempio in quelli di disponibilità di risorse) sul valore della funzione obiettivo. Questo perché, come vedremo, le variabili duali possono essere anche interpretate come i cosiddetti *prezzi ombra* in quanto misurano i “costi” impliciti associati ai vincoli.

3.2.1 Interpretazione economica della dualità e prezzi ombra

Per introdurre il concetto delle variabili duali come prezzi ombra facciamo riferimento ad un semplice esempio di modello di pianificazione della produzione che brevemente descriviamo.

Esempio 3.2.1 Un'industria produce due tipi di prodotti: un tipo de luxe e un tipo standard. Per avere un prodotto finito di ciascuno dei due tipi sono necessari due ingredienti grezzi \mathbf{I}_1 e \mathbf{I}_2 e la lavorazione su una macchina. La tabella che segue riporta le quantità in Kg di ciascuno degli ingredienti e le ore di lavorazione sulla macchina necessarie per ottenere un prodotto finito di ciascuno dei due tipi.

	de luxe	standard
\mathbf{I}_1	3	2
\mathbf{I}_2	4	1
ore lavoraz.	2	1

Settimanalmente si hanno a disposizione al più 1200 Kg dell'ingrediente \mathbf{I}_1 e al più 1000 Kg dell'ingrediente \mathbf{I}_2 mentre la disponibilità massima settimanale di ore lavorative della macchina è pari a 700. Un prodotto de luxe è venduto a 24 Euro e un prodotto standard è venduto a 14 Euro. Si vuole pianificare la produzione settimanale in modo da massimizzare il profitto complessivo assumendo che i prodotti siano frazionabili.

Si tratta di un problema di allocazione ottima di risorse limitate che può essere formulato come problema di Programmazione Lineare nel seguente modo:

$$\begin{cases} \max 24x_1 + 14x_2 \\ 3x_1 + 2x_2 \leq 1200 \\ 4x_1 + x_2 \leq 1000 \\ 2x_1 + x_2 \leq 700 \\ x_1 \geq 0, x_2 \geq 0 \end{cases}$$

dove le variabili x_1 e x_2 rappresentano le quantità di prodotti ripetutivamente del tipo *de luxe* e del tipo *standard* da fabbricare settimanalmente.

Consideriamo, ora, il problema duale del problema ora formulato; esso è

$$\begin{cases} \min 1200u_1 + 1000u_2 + 700u_3 \\ 3u_1 + 4u_2 + 2u_3 \geq 24 \\ 2u_1 + u_2 + u_3 \geq 14 \\ u_1 \geq 0, u_2 \geq 0, u_3 \geq 0. \end{cases}$$

La soluzione ottima del primale è

$$x_1^* = 160, \quad x_2^* = 360$$

e il valore ottimo della funzione obiettivo primale è pari a 8880.

La soluzione ottima del duale è

$$u_1^* = 6.4, \quad u_2^* = 1.2, \quad u_3^* = 0$$

e il valore ottimo della funzione obiettivo duale è pari a 8880. (Si veda l'Esercitazione di Laboratorio n. 4) Quindi il Teorema della Dualità Forte è verificato.

Scriviamo, ora, le condizioni di complementarità:

$$\begin{aligned} x_1^*(3u_1^* + 4u_2^* - 2u_3^* - 24) &= 0 \\ x_2^*(2u_1^* + u_2^* + u_3^* - 14) &= 0 \\ u_1^*(1200 - 3x_1^* - 2x_2^*) &= 0 \\ u_2^*(1000 - 4x_1^* - x_2^*) &= 0 \\ u_3^*(700 - 2x_1^* - x_2^*) &= 0 \end{aligned}$$

Si verifica immediatamente che tali condizioni sono soddisfatte. Si osservi che tutte le equazioni tranne l'ultima sono verificate in quanto si annulla il secondo dei due fattori moltiplicativi. Questo significa,

in particolare, che il primo e il secondo vincolo del problema primale sono attivi nella soluzione ottima, cioè verificati all'uguaglianza. L'ultima equazione invece è verificata per il fatto che è nulla all'ottimo la variabile duale u_3^* mentre il vincolo corrispondente primale (cioè il terzo vincolo del problema primale) non è verificato all'uguaglianza. Infatti in corrispondenza della soluzione ottima il valore ottenuto è $2x_1^* + x_2^* = 680$. Poiché la disponibilità di ore lavorative è pari a 700 ore, si hanno ancora 20 ore disponibili (surplus). Quindi l'industria, per aumentare il profitto, potrebbe acquistare altre quantità di ingredienti grezzi e quindi aumentare la disponibilità settimanale di questi ingredienti e utilizzare le ore di lavorazione ancora rimaste disponibili. Poiché i valori all'ottimo della funzione obiettivo primale e della funzione obiettivo duale coincidono e poiché la funzione obiettivo duale è

$$1200u_1 + 1000u_2 + 700u_3,$$

essendo $u_1^* = 6.4$, $u_2^* = 1.2$, $u_3^* = 0$, l'aumento di 1 Kg della disponibilità di ingrediente **I₁** (da 1200 a 1201 Kg) porta ad un incremento di 6.4 Euro nel profitto complessivo. Analogamente per l'ingrediente **I₂**: un incremento di 1 Kg (da 1000 a 1001 Kg) porta ad un incremento del profitto complessivo di 1.2 Euro.

Questo è il motivo per cui le variabili duali sono anche chiamate *prezzi ombra* e determinano il *valore marginale* delle risorse. Ovviamente il fatto che $u_3^* = 0$ significa che l'aumento della disponibilità di ore lavorative non porta a nessun incremento del profitto, ma questo è ovvio in quanto ore lavorative inutilizzate sono già disponibili.

Nell'ipotesi che, ad esempio, si possa incrementare la disponibilità di una sola delle risorse, naturalmente esaminando i prezzi ombra, si deduce che conviene aumentare la disponibilità dell'ingrediente **I₁** che porta ad un maggiore incremento del profitto complessivo.

Si osservi che il fatto che ad un incremento pari a δ nel termine noto del primo vincolo corrisponda un incremento pari a 6.4δ nel valore ottimo della funzione obiettivo, è valido fin tanto che la variabile duale all'ottimo u_1^* associata al primo vincolo rimane pari al valore 6.4. Infatti, ovviamente la variazione del termine noto del vincolo corrispondente alla disponibilità dell'ingrediente **I₁** porta anche ad un cambiamento nella formulazione del problema duale: infatti un cambiamento nel termine noto di un vincolo primale corrisponde ad un cambiamento in un coefficiente della funzione obiettivo del problema duale. Pertanto c'è la possibilità che se la variazione è ampia, cambi il punto di ottimo del problema duale e quindi, in particolare, cambi il prezzo ombra u_1^* associato al primo vincolo. In questo caso, naturalmente, la variazione del valore della funzione obiettivo all'ottimo non può essere più proporzionale al valore 6.4 (si veda l'Esercitazione di Laboratorio n.4). \square

Quindi come visto nell'esempio, in generale, le variabili duali (i prezzi ombra) rappresentano l'effetto di cambiamenti nel termine noto dei vincoli. Si consideri, infatti un generico problema di Programmazione Lineare (in forma standard) (P), il suo duale (D) ed inoltre si consideri il problema (P_Δ) ottenuto modificando il termine noto da b a $b + \Delta$ (con $\Delta \in \mathbb{R}^m$) e il corrispondente problema duale (D_Δ) :

$$(P) \quad \begin{cases} \min c^T x \\ Ax = b \\ x \geq 0 \end{cases} \quad (D) \quad \begin{cases} \max b^T u \\ A^T u \leq c \end{cases}$$

$$(P_\Delta) \quad \begin{cases} \min c^T x \\ Ax = b + \Delta \\ x \geq 0 \end{cases} \quad (D_\Delta) \quad \begin{cases} \max(b + \Delta)^T u \\ A^T u \leq c \end{cases}$$

Siano x^* e u^* rispettivamente la soluzione ottima del problema (P) e del problema (D). Siano inoltre $x^*(\Delta)$ e $u^*(\Delta)$ rispettivamente la soluzione del problema (P_Δ) e del problema (D_Δ)

Dalle formulazioni di questi problemi si possono facilmente dedurre due osservazioni:

- la variazione del termine noto b nel problema primale si riflette in un cambiamento dei coefficienti della funzione obiettivo del problema duale;
- la regione ammissibile del problema (D) e del problema (D_Δ) sono uguali; da questo segue che se $u^* \in \mathbb{R}^m$ è soluzione ottima del problema (D) allora u^* è ammissibile per il problema (D_Δ) , ma non necessariamente è ottima per (D_Δ) .

Inoltre per il Teorema della dualità forte applicato alla coppia primale-duale $(P)-(D)$ deve essere

$$c^T x^* = b^T u^*, \quad (3.18)$$

mentre, sempre per il della dualità forte ma applicato alla coppia primale-duale $(P_\Delta)-(D_\Delta)$ deve essere

$$c^T x^*(\Delta) = (b + \Delta)^T u^*(\Delta). \quad (3.19)$$

Se la soluzione ottima x^* soddisfa un'opportuna ipotesi (cioè che in x^* non ci siano più di n vincoli attivi) e se il vettore Δ ha componenti “sufficientemente” piccole allora si può dimostrare che:

$$u^*(\Delta) = u^*. \quad (3.20)$$

Utilizzando la (3.18), la (3.19) e la (3.20) si ha:

$$c^T x^*(\Delta) = b^T u^* + \Delta^T u^* = c^T x^* + \Delta^T u^*, \quad (3.21)$$

che può essere riscritta nella seguente forma:

$$c^T x^*(\Delta) - c^T x^* = \Delta_1 u_1^* + \Delta_2 u_2^* + \dots + \Delta_m u_m^*, \quad (3.22)$$

dove $\Delta = (\Delta_1, \dots, \Delta_m)^T$.

Dalla precedente relazione segue che una possibile interpretazione della variabile duale u_i^* è quella di essere un prezzo associato ad un incremento unitario del termine noto b_i . Per questa ragione le variabili duali u_i^* , $i = 1, \dots, m$, vengono denominate *prezzi ombra* o *costi marginali*. Sebbene la (3.20) (e di conseguenza la (3.22)) valga solamente sotto opportune ipotesi, in molte situazioni pratiche, le variabili duali u_i^* , $i = 1, \dots, m$, forniscono delle utili indicazioni su quale componente b_i variare per migliorare il valore ottimo della funzione obiettivo.

Si consideri, ora (come nell'Esempio 3.2.1) la variazione del termine noto di un solo vincolo che si ottiene prendendo $\Delta = \delta e_i$ (dove $e_i \in \mathbb{R}^m$ è il vettore con l' i -esima componente uguale a 1 e le altre componenti nulle). In questo caso, naturalmente ad una variazione del termine noto dell' i -esimo vincolo corrisponde una variazione del valore della funzione obiettivo pari a δu_i^* . Nell'esempio precedente era stato infatti osservato come una variazione di δ effettuata nel termine noto del primo vincolo porta ad una variazione della funzione obiettivo pari a $u_1^* \delta = 6.4\delta$. Si deve tuttavia ribadire un fatto molto importante: l'interpretazione delle variabili duali come prezzi ombra e quindi come strumento per valutare la variazione del valore della funzione obiettivo al variare del termine noto di un vincolo a partire da una soluzione ottima è vera *solamente per piccole variazioni* del termine noto; esiste cioè un intervallo entro il quale δ deve rimanere.

Esula dallo scopo di queste note la motivazione teorica dettagliata della validità dell'interpretazione data delle variabili duali, a partire da una soluzione ottima, come prezzi ombra rappresentanti i *valori marginali* dei termini noti dei vincoli solo per piccole perturbazioni di questi termini noti ed anche la determinazione dell'intervallo $[\delta_l, \delta_u]$ in cui può variare δ rimanendo valida tale l'interpretazione.

Queste considerazioni fanno parte della cosiddetta *analisi post-ottimale*. Un altro scopo di questo tipo di analisi è quello di indagare la “sensibilità” del modello al variare dei del termine noto dei vincoli; questo rientra nella cosiddetta *analisi della sensibilità* che affronta lo studio di come varia la soluzione ottima di un problema al variare oltre che dei termini noti dei vincoli, anche al variare dei coefficienti di costo della funzione obiettivo, oppure aggiungendo nuove variabili o nuovi vincoli. Ovviamente una trattazione rigorosa di queste problematiche esula dallo scopo di queste note e perciò si rimanda ai testi di approfondimento specifici.

3.2.2 Il duale del problema di allocazione ottima di risorse

Si consideri nuovamente il semplice problema di allocazione ottima dell'Esempio 2.7.1 che è rappresentato dal seguente problema di Programazione Lineare:

$$\begin{cases} \max (7x_1 + 10x_2) \\ x_1 + x_2 \leq 750 \\ x_1 + 2x_2 \leq 1000 \\ x_2 \leq 400 \\ x_1 \geq 0, x_2 \geq 0. \end{cases} \quad (3.23)$$

Ricordiamo che le variabili x_1 e x_2 sono associate rispettivamente ai quantitativi di colorante **C1** e **C2** da produrre e che la produzione avviene utilizzando tre preparati base **P1**, **P2** e **P3** dei quali si ha una disponibilità massima rispettivamente pari a 750, 1000 e 400 ettagrammi. Supponiamo, ora di voler sottrarre preparati base dalla produzione dei coloranti per venderli direttamente. Indichiamo con u_1 , u_2 e u_3 i prezzi associati rispettivamente alla vendita diretta di un ettagrammo di preparato base **P1**, **P2** e **P3**. Supponendo di destinare tutti i preparati alla vendita diretta, il profitto che si otterrebbe sarebbe

$$750u_1 + 1000u_2 + 400u_3. \quad (3.24)$$

Naturalmente si vorrà fare in modo che questa operazione di sottrazione dei preparati base dalla produzione dei coloranti e vendita diretta risulti economicamente conveniente e quindi mentre si vuole minimizzare l'espressione (3.24) affinché i prezzi di vendita risultino competitivi sul mercato, si imporrà che il profitto ottenuto vendendo direttamente i quantitativi di preparato base necessario per ottenere un litro di colorante sia maggiore o uguale del profitto associato alla vendita di un litro di colorante stesso; quindi, utilizzando i dati del problema riportati nella tabella dell'Esempio 2.7.1, si deve imporre che risulti

$$u_1 + u_2 \geq 7$$

per quanto riguarda il colorante **C1** e

$$u_1 + 2u_2 + u_3 \geq 10$$

per quanto riguarda il colorante **C2** e naturalmente deve essere $u_1 \geq 0$, $u_2 \geq 0$ e $u_3 \geq 0$. Quindi il modello lineare che rappresenta l'operazione sopra descritta è il seguente:

$$\begin{cases} \min (750u_1 + 1000u_2 + 400u_3) \\ u_1 + u_2 \geq 7 \\ u_1 + 2u_2 + u_3 \geq 10 \\ u_1 \geq 0, u_2 \geq 0, u_3 \geq 0. \end{cases}$$

Esaminando questo problema si vede immediatamente che esso rappresenta il *problema duale* del problema dato (3.23).

In generale, se si considera un generico problema di allocazione ottima di m risorse \mathbf{R}_i , $i = 1, \dots, m$ con la possibilità di fabbricare n prodotti \mathbf{P}_j , $j = 1, \dots, n$, come abbiamo già esaminato nel capitolo precedente si può formulare questo problema come

$$\begin{cases} \max c^T x \\ Ax \leq b \\ x \geq 0 \end{cases} \quad (3.25)$$

dove ricordiamo $x \in \mathbb{R}^n$ è il vettore avente per componenti i livelli di produzione di ciascuno dei prodotti, $c \in \mathbb{R}^n$ il vettore dei profitti netti e $b \in \mathbb{R}^m$ il vettore delle disponibilità massima di ciascuna delle risorse.

Supponiamo ora di voler sottrarre risorse alla produzione per venderle direttamente e siano u_i , $i = 1, \dots, m$ i prezzi unitari associati alla vendita dell' i -esima risorsa. Supponendo che per ciascuna risorsa

si voglia destinare alla vendita una quantità pari alla disponibilità massima di quella risorsa, si ottiene un profitto pari a

$$b_1u_1 + b_2u_2 + \cdots + b_mu_m.$$

Per rendere competitivi sul mercato i prezzi unitari u_i da assegnare alle risorse vendute direttamente, si vogliono scegliere i valori più bassi possibile per le u_i , ma naturalmente, affinché questa operazione di vendita diretta in luogo della fabbricazione dei prodotti risulti conveniente si deve imporre che il profitto ottenuto vendendo direttamente le risorse necessarie per fabbricare un prodotto sia maggiore o uguale al profitto che si ricaverebbe dalla vendita del prodotto finito. Quindi per ogni prodotto, si deve imporre che valga

$$\begin{array}{lllll} a_{11}u_1 + & \dots & + a_{m1}u_m & \geq c_1 \\ a_{12}u_1 + & \dots & + a_{m2}u_m & \geq c_2 \\ \vdots & \dots & \vdots & \vdots \\ a_{1n}u_1 + & \dots & + a_{mn}u_m & \geq c_n \end{array}$$

con $u_i \geq 0$, $i = 1, \dots, m$ e dove le quantità a_{ij} rappresentano la quantità di risorsa \mathbf{R}_i necessaria per fabbricare una unità di prodotto \mathbf{P}_j .

Quindi il problema da risolvere può essere scritto nella forma

$$\begin{cases} \min b^T u \\ A^T u \geq c \\ u \geq 0 \end{cases}$$

che è il problema duale del problema (3.25).

3.2.3 Il duale del problema di miscelazione

Si consideri il problema di miscelazione dell'Esempio 2.7.6 che è rappresentato dal seguente problema di Programmazione Lineare:

$$\begin{cases} \min(400x_1 + 600x_2) \\ 140x_1 \geq 70 \\ 20x_1 + 10x_2 \geq 30 \\ 25x_1 + 50x_2 \geq 75 \\ x_1 \geq 0, x_2 \geq 0. \end{cases}$$

Ricordiamo che le variabili x_1 e x_2 rappresentano le quantità di polpa di frutta e di dolcificante da utilizzare nella produzione del succo di frutta che deve avere come requisito un contenuto minimo di 70 mg di vitamina C, 30 mg di sali minerali e 75 grammi di zucchero. Supponiamo ora che un'industria farmaceutica venda compresse di nutrimenti puri, cioè compresse di vitamina C, di sali minerali e di zucchero e che vuole immettere queste compresse su un ipotetico mercato come offerta sostitutiva al succo di frutta per l'acquisizione di vitamina C, di sali minerali e di zucchero. Naturalmente questa industria farmaceutica vuole massimizzare il profitto ricavato dalla vendita delle compresse, ma al tempo stesso deve dare un prezzo alle compresse tale da essere competitiva. Siano allora u_1 , u_2 e u_3 i prezzi di vendita rispettivamente di 1 mg di vitamina C, di 1 mg di sali minerali e di 1 grammo di zucchero; supponendo che la vendita di questi nutrimenti puri sia pari ai fabbisogni minimi (cioè a 70 mg di vitamina C, a 30 mg di sali minerali e a 75 grammi di zucchero), l'espressione del profitto dell'industria farmaceutica che dovrà essere massimizzata è

$$70u_1 + 30u_2 + 75u_3.$$

Affinché i prezzi di vendita dei componenti puri in compresse fissati dall'industria siano concorrenziali, si deve imporre che il costo unitario dei nutrimenti puri sia minore o uguale al prezzo che si dovrebbe

pagare per avere la stessa quantità di componente attraverso gli ingredienti del succo di frutta, cioè dalla polpa di frutta e dal dolcificante. Quindi si devono imporre i seguenti vincoli

$$\begin{aligned} 140u_1 + 20u_2 + 25u_3 &\leq 400 \\ 10u_2 + 50u_3 &\leq 600. \end{aligned}$$

Inoltre dovrà essere $u_1 \geq 0, u_2 \geq 0, u_3 \geq 0$.

Quindi il problema complessivo formulato dall'industria farmaceutica è

$$\begin{cases} \max(70u_1 + 30u_2 + 75u_3) \\ 140u_1 + 20u_2 + 25u_3 \leq 400 \\ 10u_2 + 50u_3 \leq 600 \\ u_1 \geq 0, u_2 \geq 0, u_3 \geq 0 \end{cases}$$

che è il problema duale del problema di miscelazione considerato.

In generale, consideriamo un generico problema di miscelazione in cui si hanno n sostanze $\mathbf{S}_j, j = 1, \dots, n$ ciascuna delle quali contiene una quantità a_{ij} di componente utile $\mathbf{C}_i, i = 1, \dots, m$. Come abbiamo già esaminato nel capitolo precedente un problema di miscelazione di questo tipo si può formulare come

$$\begin{cases} \min c^T x \\ Ax \geq b \\ x \geq 0 \end{cases} \quad (3.26)$$

dove ricordiamo che $x \in \mathbb{R}^n$ è il vettore avente per componenti le quantità di ciascuna sostanza da introdurre nella miscela, $c \in \mathbb{R}^n$ il vettore dei costi unitari delle sostanze, $b \in \mathbb{R}^m$ il vettore dei requisiti minimi di componenti utili da introdurre nella miscela, e $A \in \mathbb{R}^{m \times n}$ la matrice i cui elementi sono le a_{ij} .

Supponiamo ora che un'industria sia in grado di fornire componenti utili allo stato puro e che voglia immettere sul mercato questi componenti utili e siano $u_i, i = 1, \dots, m$ i prezzi unitari da assegnare a ciascuno di essi. Supponendo che la richiesta del mercato sia pari ai fabbisogni minimi della miscela, cioè per ciascun componente pari a b_i , il profitto totale dell'industria che vende i componenti utili allo stato puro è

$$b_1u_1 + b_2u_2 + \dots + b_mu_m.$$

Inoltre, affinché i prezzi u_i assegnati dall'industria ai componenti puri siano concorrenziali, si deve imporre che il costo dei componenti puri sia minore o uguale al prezzo che dovrebbe pagare per avere la stessa quantità di componente ottenuto attraverso le sostanze e quindi deve valere

$$\sum_{i=1}^m a_{ij}u_i \leq c_j, \quad j = 1, \dots, n.$$

Inoltre si deve imporre $u_i \geq 0, i = 1, \dots, m$.

Quindi il problema formulato si può scrivere nella forma

$$\begin{cases} \max b^T u \\ A^T u \leq c \\ u \geq 0 \end{cases}$$

che è immediato verificare essere il problema duale del problema di miscelazione assegnato (3.26).

3.2.4 Il duale del problema dei trasporti

Si consideri il problema di trasporto dell'Esempio ?? che è rappresentato dal seguente problema di Programmazione Lineare:

$$\left\{ \begin{array}{l} \min(250x_1 + 100x_2 + 85x_3 + 120y_1 + 80y_2 + 150y_3) \\ \\ \begin{array}{rcl} x_1 + x_2 + x_3 & = & 50 \\ y_1 + y_2 + y_3 & = & 55 \\ x_1 + y_1 & = & 30 \\ x_2 + y_2 & = & 40 \\ x_3 + y_3 & = & 35 \\ \end{array} \\ \\ x_1 \geq 0, \quad x_2 \geq 0, \quad x_3 \geq 0, \quad y_1 \geq 0, \quad y_2 \geq 0, \quad y_3 \geq 0. \end{array} \right.$$

dove x_1, x_2, x_3 rappresentano rispettivamente la quantità di acqua da trasportare dal primo stabilimento ai tre impianti e y_1, y_2, y_3 rappresentano rispettivamente la quantità di acqua da trasportare dal secondo stabilimento ai tre impianti. Ricordiamo inoltre che la disponibilità giornaliera di acqua presso i due stabilimenti è rispettivamente di 50 e 55 ettolitri di acqua, mentre le richieste giornaliere di acqua presso i tre impianti sono rispettivamente di 30, 40 e 35 ettolitri.

Supponiamo ora che una compagnia specializzata in trasporto di acqua (esterna all'industria) voglia proporsi all'industria di acque minerali per effettuare il trasporto dell'acqua dagli stabilimenti agli impianti. Naturalmente la compagnia di trasporti, per convincere l'industria di acque minerali ad avvalersi del servizio di trasporto esterno, dovrà proporre dei prezzi di trasporto vantaggiosi. A tale scopo la compagnia dei trasporti propone all'industria di prelevare un ettolitro di acqua da ciascuno dei due stabilimenti per un prezzo unitario (in migliaia di lire) rispettivamente pari a u_1 e u_2 e di consegnare un ettolitro di acqua a ciascuno dei tre impianti per un prezzo unitario (in migliaia di lire) rispettivamente pari a v_1 , v_2 e v_3 . Quindi la compagnia dei trasporti vorrà massimizzare la funzione che fornisce il suo profitto complessivo che è data da

$$50u_1 + 55u_2 + 30v_1 + 40v_2 + 35v_3.$$

Tuttavia affinché l'offerta della compagnia dei trasporti risulti vantaggiosa per l'industria delle acque minerali i prezzi del traporto proposti dovranno risultare non superiori a quelli che l'industria avrebbe effettuando in proprio i trasporti stessi. Quindi dovrà risultare

$$\begin{array}{lcl} u_1 + v_1 & \leq & 250 \\ u_1 + v_2 & \leq & 100 \\ u_1 + v_3 & \leq & 85 \\ u_2 + v_1 & \leq & 120 \\ u_2 + v_2 & \leq & 80 \\ u_2 + v_3 & \leq & 150. \end{array}$$

Quindi, la compagnia dei trasporti dovrà risolvere il problema

$$\left\{ \begin{array}{l} \max (50u_1 + 55u_2 + 30v_1 + 40v_2 + 35v_3) \\ \\ \begin{array}{rcl} u_1 + v_1 & \leq & 250 \\ u_1 + v_2 & \leq & 100 \\ u_1 + v_3 & \leq & 85 \\ u_2 + v_1 & \leq & 120 \\ u_2 + v_2 & \leq & 80 \\ u_2 + v_3 & \leq & 150 \end{array} \end{array} \right.$$

che si verifica immediatamente essere il problema duale del problema dei trasporti assegnato.

In generale, consideriamo ora un generico problema dei trasporti già esaminato nel capitolo precedente. Supponiamo che un'azienda voglia provvedere in proprio ad effettuare il trasporto di materiali e che quindi cerchi di risolvere il problema dei trasporti (2.10) cioè

$$\begin{cases} \min \left(\sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \right) \\ \sum_{j=1}^n x_{ij} = a_i \quad i = 1, \dots, m \\ \sum_{i=1}^m x_{ij} = b_j \quad j = 1, \dots, n \\ x_{ij} \geq 0 \quad i = 1, \dots, n; \quad j = 1, \dots, m \end{cases} \quad (3.27)$$

dove, ricordiamo, che le c_{ij} rappresentano il costo del trasporto dall'origine i alla destinazione j , le a_i le disponibilità all' i -esima origine e le b_j le richieste alla j -esima destinazione.

Supponiamo, ora che una compagnia che esegue trasporti voglia proporsi a questa azienda, come alternativa vantaggiosa all'effettuazione dei trasporti in proprio; a tale scopo questa compagnia propone all'azienda di prelevare un'unità di prodotto dall'origine i per un prezzo unitario u_i e di consegnare una unità di prodotto alla destinazione j per un prezzo unitario v_j . Per assicurare che i suoi prezzi siano competitivi rispetto a quelli che l'azienda avrebbe effettuando i trasporti in proprio, la compagnia di trasporti deve fare sì che risulti

$$u_i + v_j \leq c_{ij}$$

per ogni $i = 1, \dots, m$ e $j = 1, \dots, n$. D'altra parte la compagnia di trasporti vuole scegliere i prezzi da proporre u_1, \dots, u_m e v_1, \dots, v_n in modo da massimizzare il suo profitto complessivo. Poiché le quantità a_i e b_j di prodotto rispettivamente disponibili all'origine i e richieste alla destinazione j sono note alla compagnia di trasporti, questa cercherà di massimizzare la funzione

$$\max \left(\sum_{i=1}^m a_i u_i + \sum_{j=1}^n b_j v_j \right).$$

Quindi il problema che la compagnia di trasporti formula per determinare quali prezzi u_i e v_j proporre all'azienda è il seguente

$$\begin{cases} \max \left(\sum_{i=1}^m a_i u_i + \sum_{j=1}^n b_j v_j \right) \\ u_i + v_j \leq c_{ij} \quad i = 1, \dots, m, \quad j = 1, \dots, n \end{cases} \quad (3.28)$$

che è il problema duale del problema dei trasporti (3.27).

Capitolo 4

Problemi di Programmazione Lineare Intera

La Programmazione Lineare Intera (PLI) tratta il problema della massimizzazione (minimizzazione) di una funzione di più variabili, soggetta a vincoli di uguaglianza e disuguaglianza ed alla restrizione che una o più variabili possano assumere soltanto valori interi.

Grazie alla generalità del modello, un grandissimo numero di problemi reali possono essere rappresentati da modelli di Programmazione Lineare Intera. In generale, i modelli di Programmazione Intera sono adatti alle applicazioni caratterizzate dall'indivisibilità delle risorse e dalla necessità di scegliere tra un numero finito di alternative. Queste applicazioni includono *problemi operativi* quali la distribuzione di beni ed il sequenziamento delle attività produttive; *problemi di pianificazione* quali la gestione ottima del portafoglio titoli e la localizzazione degli impianti ed infine *problemi di progettazione* quali il progetto di circuiti VLSI ed il progetto di sistemi automatici di produzione (robotica). Recenti applicazioni dell'ottimizzazione combinatoria ad altri settori riguardano problemi in biologia molecolare, fisica delle alte energie e cristallografia a raggi X.

Le questioni teoriche poste da tali problemi e le tecniche usate per la loro soluzione hanno caratteristiche molto diverse da quelle relative ai problemi di ottimizzazione continua studiati nei precedenti capitoli. Lo scopo di questo capitolo è quindi quello di dare alcune indicazioni sulle peculiarità dei problemi di Programmazione Lineare Intera (PLI) e studiare una classe di algoritmi per la loro risoluzione.

4.1 Formulazioni Classiche di Problemi Lineari Interi

In questo paragrafo vengono presentati esempi classici di problemi che possono essere formulati come problemi di PLI. Lo scopo di questi esempi è quello di mostrare come le variabili intere e le disequazioni che le collegano possano essere usate come *linguaggio formale* per esprimere una serie di relazioni tra eventi.

4.1.1 Knapsack binario

Il primo uso delle variabili intere (binarie) che esamineremo è anche il più naturale. Si supponga di dover modellare il fatto che un dato evento possa verificarsi oppure no. La natura binaria del problema suggerisce immediatamente l'idea di modellare questa dicotomia per mezzo di una variabile x che può assumere solo valori 0, 1. In particolare, si porrà $x = 1$ se l'evento si verifica e $x = 0$ altrimenti.

Supponiamo di avere n oggetti. Il j -esimo oggetto, $j = 1, \dots, n$, ha un valore pari a c_j e un peso pari a p_j . Supponiamo di avere una bisaccia ("knapsack" in inglese) in cui vogliamo mettere alcuni degli oggetti. La bisaccia può portare al massimo un peso b . Il problema di scegliere un sottoinsieme degli

oggetti allo scopo di massimizzare la somma dei valori senza superare il limite imposto dal peso è il cosiddetto problema di *knapsack binario*:

$$\begin{aligned} \max \quad & \sum_{j=1}^n c_j x_j \\ \text{subject to} \quad & \sum_{j=1}^n p_j x_j \leq b \\ & x_j \in \{0, 1\} \quad j = 1, \dots, n. \end{aligned}$$

In questo caso l'evento è costituito dalla scelta (o meno) del singolo oggetto. In generale, questi problemi di scelta tra progetti possono avere più vincoli (si pensi al caso in cui la bisaccia abbia un limite dovuto al peso e uno dovuto al volume degli oggetti); in tal caso il problema viene detto di *knapsack multidimensionale*.

4.1.2 Assegnamento

Un altro problema classico di pianificazione riguarda l'assegnamento di lavori a persone. Supponiamo che n persone debbano svolgere n lavori. Ciascun lavoro deve essere svolto esattamente da una persona; inoltre, ciascuna persona può svolgere al più un lavoro. Il costo della persona j assegnata al lavoro i è c_{ij} . Il problema è quello di assegnare i lavori alle persone minimizzando il costo totale di realizzazione di tutti i lavori. Per formulare questo problema, che è noto come problema di *assegnamento*, introduciamo le variabili binarie x_{ij} , $i = 1, \dots, n$, $j = 1, \dots, n$ corrispondenti all'evento ij definite come segue

$$x_{ij} = \begin{cases} 1 & \text{se la persona } j \text{ è assegnata al lavoro } i \\ 0 & \text{altrimenti} \end{cases}$$

Poiché esattamente una persona deve essere assegnata al lavoro i , avremo i vincoli:

$$\sum_{j=1}^n x_{ij} = 1, \quad i = 1, \dots, n.$$

Inoltre, poiché ciascuna persona non può svolgere più di un lavoro, avremo i vincoli:

$$\sum_{i=1}^n x_{ij} = 1, \quad j = 1, \dots, n.$$

È facile verificare che un vettore $x \in \{0, 1\}^{m \times n}$ che soddisfa tutti i vincoli appena descritti individua un assegnamento ammissibile di persone ai lavori. La funzione obiettivo, ovviamente da minimizzare, può essere scritta come $\sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$.

4.1.3 Problema del costo fisso.

Nei modelli di Programmazione Lineare le variabili di decisione rappresentano usualmente i livelli ai quali le varie attività vengono svolte e la funzione obiettivo da minimizzare è una funzione lineare di tali variabili. In molti problemi pratici, tuttavia, tale ipotesi non è giustificata in quanto il costo di una attività, in funzione del livello cui essa viene svolta, può avere un andamento come quello riportato in Figura 4.1.

In particolare, il costo dell'attività j è zero se $x_j = 0$ (cioè se l'attività non è avviata) ed è invece uguale a $f_j + c_j x_j$ se $x_j > 0$ con f_j positivo. La funzione relativa all'attività j è quindi

$$z_j(x) = \begin{cases} 0 & \text{se } x_j = 0 \\ f_j + c_j x_j & \text{se } x_j > 0 \end{cases}$$

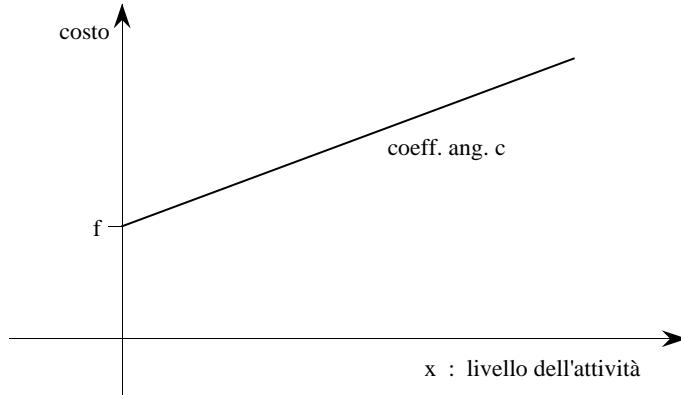


Figura 4.1: Costo dell'attività j .

Il valore f_j è detto *costo fisso* dell'attività j e deve essere pagato solamente se l'attività j viene svolta ad un livello non nullo. Per esempio, supponiamo che la variabile x_j rappresenti la quantità di petrolio greggio che deve essere trasportata da un pozzo A ad una raffineria B . In questo caso, se il pozzo A non rifornisce la raffineria B il costo complessivo di trasporto è ovviamente nullo. Se, al contrario, si decide di inviare una quantità non nulla x_j di greggio da A a B , al costo di trasporto $c_j x_j$ (proporzionale alla quantità trasferita) dovrà essere sommato il costo fisso di costruzione dell'oleodotto f_j .

Se indichiamo che J_f le attività che prevedono un costo fisso, il problema che dobbiamo risolvere può essere scritto

$$\begin{aligned} \min \quad z(x) &= \sum_{j \notin J_f} c_j x_j + \sum_{j \in J_f} z_j(x) \\ Ax &= b \\ x &\geq 0_n \end{aligned}$$

Chiaramente la funzione $z(x)$ è discontinua nell'origine e quindi il problema non è di Programmazione Lineare. Una possibile formulazione alternativa del Problema di Costo Fisso come problema di PLI si ottiene introducendo, per ciascuna attività, una variabile y_j che valga 1 quando $x_j > 0$ e 0 quando $x_j = 0$, cioè

$$y_j = \begin{cases} 0 & \text{se } x_j = 0 \\ 1 & \text{altrimenti} \end{cases}$$

In questo modo la funzione obiettivo può essere scritta

$$z(x) = \sum_{j=1}^n c_j x_j + \sum_{j \in J_f} y_j f_j = c^T x + f^T y,$$

dove $y \in \{0, 1\}^{|J_f|}$ è il vettore le cui componenti sono y_j e $z(x)$ è una funzione lineare.

Si tratta ora di imporre un vincolo che consenta di modellare le condizioni logiche

$$\begin{aligned} x_j > 0 &\Rightarrow y_j = 1 \\ x_j = 0 &\Rightarrow y_j = 0 \end{aligned}$$

Se supponiamo di conoscere un limite superiore per la variabile x_j , cioè un valore α positivo maggiore del più grande valore che può assumere la x_j , il vincolo

$$x_j - \alpha y_j \leq 0$$

forza la variabile y_j ad assumere valore 1 se $x_j > 0^1$.

Per quanto riguarda la seconda condizione $x_j = 0 \Rightarrow y_j = 0$, osserviamo che se $x_j = 0$ la variabile y_j il processo di minimizzazione farà in modo che y_j sia nulla all'ottimo poiché $f_j \geq 0$.

Il problema di costo fisso può essere quindi formulato, assumendo che tutti i costi fissi f_j siano positivi, nel modo seguente:

$$\begin{aligned}
 (FC2) \quad \min z(x) &= c^T x + f^T y \\
 Ax &= b \\
 x_j &\leq \alpha y_j, \quad j \in J_f \\
 x &\geq 0_n \\
 y_j &\in \{0, 1\}, \quad j \in J_f
 \end{aligned}$$

dove α è un numero positivo maggiore del più grande valore che può essere assunto da ciascuna delle variabili x_j in una soluzione ottima. Se $x_j > 0$ la variabile y_j sarà forzata ad assumere il valore 1 ed il suo costo fisso si aggiungerà al valore della funzione obiettivo. È quindi evidente che una soluzione ottima del problema FC1 è anche ottima per FC2 e viceversa.

4.1.4 Capital Budgeting

Questo esempio illustra l'applicazione della PLI al problema della pianificazione degli investimenti ("Capital Budgeting" in inglese). Si tratta di uno degli esempi più significativi di applicazione della PLI alle problematiche della pianificazione finanziaria.

Un'azienda genera continuamente, nello svolgimento delle sue attività, proposte di investimento e spesa (*progetti* nel seguito). Alcuni di questi progetti richiedono l'impiego di risorse finanziarie per consentire lo sviluppo dei prodotti e della produzione; altre proposte possono invece riguardare il miglioramento delle strutture produttive dell'azienda. Usualmente la decisione di attivare o meno un progetto condiziona la possibilità di attivare altri progetti, sia per la presenza di limitazioni sulla disponibilità dei capitali (*Razionamento dei Capitali*), sia per la presenza di vincoli sulla disponibilità di personale, macchine etc.

Un'azienda che si appresti a operare delle scelte individua solitamente un orizzonte temporale T entro il quale intende limitare l'analisi. Per esempio si decide di considerare gli investimenti e le loro conseguenze limitando l'analisi ai prossimi tre anni, $T = 3$ anni. L'orizzonte temporale viene poi suddiviso in *periodi* $1, 2, \dots, t$. Per esempio, se $T = 3$ anni, e i periodi sono i trimestri abbiamo dodici periodi, cioè $t = 12$. Il progetto i può essere caratterizzato dal vettore $a_i = (a_{i1}, a_{i2}, \dots, a_{it})$ del *Flusso di Cassa*. Il valore a_{ij} rappresenta il flusso di cassa (positivo o negativo) generato dal progetto i nel periodo j . Assumiamo che un flusso di cassa positivo corrisponda a una spesa, mentre uno negativo a un guadagno. Quindi se il vettore di flusso di cassa relativo a un certo progetto su un orizzonte temporale diviso in 5 periodi è $(5, 2, -1, -3, -8)$, vorrà dire che l'attivazione del progetto richiede una spesa di 5 nel primo periodo, di 2 nel secondo, e dà quindi un guadagno di 1, 3, 8 nel terzo, quarto e quinto periodo rispettivamente. Notiamo per inciso che questa è una struttura tipica (anche se esistono certamente delle eccezioni) dei flussi di cassa. Nei primi periodi l'attivazione di un progetto richiede degli investimenti, e quindi delle spese, una volta superata questa fase, si iniziano ad avere guadagni. L'esborso totale generato dal progetto è dato dalla somma dei flussi di cassa nei vari periodi. In questo caso, l'esborso totale è $5+2-1-3-8 = -5$. Ricordiamo che con la nostra convenzione sui segni un numero negativo rappresenta un guadagno² In

¹Osserviamo che la condizione $x_j > 0 \Rightarrow y_j = 1$ è equivalente a $y_j = 0 \Rightarrow x_j = 0$. Inoltre, volendo modellare la condizione $y_j = 1 \Rightarrow x_j = 0$, si può imporre il vincolo $x_j - \alpha(y_j - 1) \leq 0$.

²E' evidente che un guadagno (o un costo) all'istante j ha un "valore" diverso da un guadagno (o un costo) di pari importo ottenuto all'istante $k > j$. Per esempio, chiunque, dovendo scegliere tra l'avere un milione oggi o tra 2 anni, preferirebbe avere un milione oggi. Infatti, un milione oggi può essere investito in modo sicuro, (per esempio in buoni del tesoro) in modo di avere, tra due anni, più di un milione. Nel valutare una proposta

una situazione reale, per ogni periodo $j = 1, 2, \dots, t$ abbiamo un "budget" cioè un limite agli esborsi (flussi di cassa positivi) che possiamo fare. Tali limiti derivano, ad esempio, da scelte aziendali o dalla limitata capacità dell'azienda di accedere al credito. Indichiamo con b_j il limite di budget nel periodo j -esimo. Cerchiamo di formulare adesso il problema di scegliere un sottoinsieme di progetti da attivare con il vincolo che in ogni periodo il vincolo di budget sia rispettato e in modo tale da rendere massimo il guadagno totale. A questo fine possiamo introdurre una variabile binaria x_i per ognuno degli n progetti possibili. Le variabili sono definite nel seguente modo:

$$x_i = \begin{cases} 1 & \text{se il progetto } i \text{ è attivato} \\ 0 & \text{altrimenti} \end{cases}$$

Possiamo a questo punto formulare facilmente il problema come problema di PLI:

$$\begin{aligned} \max \quad & \sum_{i=1}^n x_i \left(-\sum_{j=1}^t a_{ij} \right) \\ & \sum_{i=1}^n a_{ij} x_i \leq b_j, \quad j = 1, \dots, t \\ & x_i \in \{0, 1\} \quad i = 1, \dots, n. \end{aligned}$$

Per capire la funzione obiettivo, bisogna ricordare che con la nostra convenzione di segni i guadagni sono numeri negativi. Quindi il termine $\left(-\sum_{j=1}^t a_{ij} \right)$ rappresenta, per ogni i , il guadagno derivante dall'attivazione del progetto i . Il cambio di segno serve a far diventare i guadagni numeri positivi e le perdite numeri negativi. I vincoli sono t , uno per ciascun periodo in cui è diviso l'orizzonte temporale e impongono che in ogni periodo non si superi il budget disponibile.

La formulazione del problema come problema di PLI permette di tenere facilmente conto di *vincoli logici* tra le attività. Facciamo alcuni esempi.

1. Se vogliamo imporre che il progetto 1 venga sicuramente attivato, basta aggiungere alla formulazione precedente il vincolo $x_1 = 1$.
2. Se vogliamo dire che almeno uno tra i progetti 2, 5, 89 deve essere attivato, basta aggiungere alla formulazione precedente il vincolo

$$x_2 + x_5 + x_{89} \geq 1.$$

3. Se vogliamo dire che uno e uno solo tra i progetti 2, 5, 89 deve essere attivato, basta aggiungere alla formulazione precedente il vincolo

$$x_2 + x_5 + x_{89} = 1.$$

4. Se vogliamo dire che al più uno tra i progetti 2, 5, 89 deve essere attivato, basta aggiungere alla formulazione precedente il vincolo

$$x_2 + x_5 + x_{89} \leq 1.$$

In questo, e nel precedente caso, i progetti di dicono *alternativi*.

5. Se vogliamo modellare il fatto che il progetto 7 può essere attivato solo se è stato attivato il progetto 9, basta aggiungere il vincolo

$$x_7 \leq x_9.$$

Se il progetto 7 può essere attivato solo se sono stati attivati i progetti 9 e 10, si può aggiungere il vincolo

$$x_7 \leq \frac{1}{2}(x_9 + x_{10}).$$

di investimento è fondamentale riportare i flussi di cassa nei vari periodi a una base comune. Questo processo è detto *attualizzazione*. Senza approfondire l'argomento, noi supponiamo che i vari a_{ij} siano stati attualizzati e siano quindi tra loro pienamente confrontabili. La procedura di attualizzazione è una procedura standard, molto semplice che non riportiamo qui solo perché non aggiunge nulla agli argomenti cui siamo interessati.

Questi esempi possono essere facilmente generalizzati, e rappresentano il prototipo di come le variabili intere (e in particolare le variabili binarie) possano essere usate per modellare relazioni logiche tra eventi (in questo caso l'evento consiste nell'attivazione o meno di un progetto).

Facciamo un esempio di quanto visto finora. Supponiamo che il nostro orizzonte temporale di un anno sia diviso in 6 bimestri ($T = \text{un anno}$, $t = 6$), e che all'inizio dell'anno si debba decidere quali tra 5 progetti attivare. Nella tabella seguente si riportano i dati di interesse. La riga relativa ad ogni progetto riporta, per ogni periodo, il flusso di cassa generato dal progetto, mentre nell'ultima riga è riportato, per ogni periodo, il budget disponibile (le cifre sono tutte in milioni di euro e sono state attualizzate).

	Periodo 1	Periodo 2	Periodo 3	Periodo 4	Periodo 5	Periodo 6
Progetto 1	2	1	0	-2	-3	-1
Progetto 2	3	3	1	-4	-3	-4
Progetto 3	1	3	-1	2	-3	-4
Progetto 4	0	4	0	-5	-1	0
Progetto 5	4	-1	-3	-2	-1	-1
budget	8	8	5	2	2	1

Indichiamo con a_i la somma dei flussi di cassa generati dal progetto nei sei periodi i : $a_i = \sum_{j=1}^6 a_{ij}$. Abbiamo

$$a_1 = -3, \quad a_2 = -4, \quad a_3 = -2, \quad a_4 = -2, \quad a_5 = -4.$$

Notiamo che gli a_i sono tutti negativi, e questo corrisponde al fatto che i progetti considerati, al termine dell'orizzonte temporale, danno dei guadagni. A questo punto possiamo scrivere facilmente il problema di PLI che ci permette di determinare la scelta ottima. Il problema ha cinque variabili, una per ogni progetto, e sei vincoli di budget, uno per ogni periodo.

$$\begin{aligned}
 \max \quad & 3x_1 + 4x_2 + 2x_3 + 2x_4 + 4x_5 \\
 & 2x_1 + 3x_2 + x_3 + 4x_5 \leq 8 \\
 & x_1 + 3x_2 + 3x_3 + 4x_4 - x_5 \leq 8 \\
 & x_2 - x_3 - 3x_5 \leq 5 \\
 & -2x_1 - 4x_2 + 2x_3 - 5x_4 - 2x_5 \leq 2 \\
 & -3x_1 - 3x_2 - 3x_3 - x_4 - x_5 \leq 2 \\
 & -x_1 - 4x_2 - 4x_3 - x_5 \leq 1 \\
 & x_i \in \{0, 1\} \quad i = 1, \dots, 5.
 \end{aligned}$$

Supponiamo ora che il progetto 4 copra solo 4 periodi dei 6 considerati. Questo è congruente col fatto che nel primo e nel sesto periodo quel progetto genera un flusso di cassa nullo.³ Più precisamente, possiamo supporre che il progetto 4 copra il secondo, terzo, quarto e quinto periodo. Ci possiamo allora porre il problema di "posizionare" temporalmente il progetto. In particolare ci possiamo domandare se è conveniente far partire il progetto 4 e, nel caso, se sia più conveniente farlo partire nel primo nel secondo

³Attenzione, però, non è detto che un flusso di cassa nullo debba necessariamente corrispondere al fatto che un progetto non copra tutti i periodi. Un flusso di cassa zero può benissimo indicare semplicemente il fatto che in quel periodo le spese e i guadagni si equilibrano (vedi per esempio il progetto 1 nel periodo 3, o lo stesso progetto 4 sempre nel terzo periodo). In altre parole, l'informazione sulla durata effettiva dei progetti non è interamente ricavabile da una tabella come quella riportata. Per esempio, sarebbe stato congruente con la tabella data sapere che il progetto 4 copre 5 periodi (dal secondo al seto), ma che nell'ultimo periodo non genera né guadagni né perdite. Al contrario, dalla tabella si può dire con certezza che il progetto 5 copre tutti e sei i periodi considerati, perché esso genera un flusso di cassa non nullo in ogni periodo.

o nel terzo periodo. Notiamo che nella versione attuale il progetto 4 viene fatto iniziare nel secondo periodo. Per analizzare questo caso possiamo introdurre, al posto del progetto 4, tre progetti, che non sono altro che copie del progetto 4 posizionate temporalmente in modo diverso. La tabella con i dati del problema diventa allora la seguente.

	Periodo 1	Periodo 2	Periodo 3	Periodo 4	Periodo 5	Periodo 6
Progetto 1	2	1	0	-2	-3	-1
Progetto 2	3	3	1	-4	-3	-4
Progetto 3	1	3	-1	2	-3	-4
Progetto 4a	4	0	-5	-1	0	0
Progetto 4b	0	4	0	-5	-1	0
Progetto 4c	0	0	4	0	-5	-1
Progetto 5	4	-1	-3	-2	-1	-1
budget	8	8	5	2	2	1

In corrispondenza il problema di PLI diventa:

$$\begin{aligned}
 \max \quad & 3x_1 + 4x_2 + 2x_3 + 2(x_{4a} + x_{4b} + x_{4c}) + 4x_5 \\
 2x_1 + 3x_2 + x_3 + 4x_{4a} + 4x_5 & \leq 8 \\
 x_1 + 3x_2 + 3x_3 + 4x_{4b} - x_5 & \leq 8 \\
 x_2 - x_3 - 5x_{4a} + 4x_{4c} - 3x_5 & \leq 5 \\
 -2x_1 - 4x_2 + 2x_3 - x_{4a} - 5x_{4b} - 2x_5 & \leq 2 \\
 -3x_1 - 3x_2 - 3x_3 - x_{4b} - 5x_{4c} - x_5 & \leq 2 \\
 -x_1 - 4x_2 - 4x_3 - x_{4c} - x_5 & \leq 1 \\
 x_{4a} + x_{4b} + x_{4c} & \leq 1 \\
 x_i \in \{0, 1\}^7.
 \end{aligned}$$

Notiamo che, oltre all'ovvio incremento di variabili, abbiamo aggiunto il vincolo

$$x_{4a} + x_{4b} + x_{4c} \leq 1,$$

che ci dice che al più una delle “copie” del progetto 4 può essere attivata, non avendo ovviamente senso dire che bisogna far partire lo stesso progetto in due periodi differenti.

4.1.5 Localizzazione

I modelli di localizzazione sono uno dei principali strumenti per la pianificazione territoriale di *reti di servizio*. L’obiettivo generico è quello di decidere dove localizzare dei *centri di servizio*, quali impianti di produzione, depositi per la distribuzione, sportelli bancari, ospedali, allo scopo di soddisfare una domanda distribuita sul territorio, minimizzando un’opportuna funzione di costo.

Si consideri l’esempio di un’amministrazione cittadina che debba decidere dove costruire un numero prefissato di centri di pronto soccorso per servire i quartieri della città. Per ogni possibile sito di localizzazione sono noti i tempi medi di percorrenza da ciascun quartiere. Per fissare le idee, supponiamo che il numero di centri da costruire sia p , il numero di quartieri sia m e il numero di possibili siti per la localizzazione degli impianti sia n . Inoltre, indicheremo con c_{ij} il tempo medio di percorrenza dal quartiere $i \in \{1, \dots, m\}$ al sito $j = \{1, \dots, n\}$. L’obiettivo degli amministratori è quello di non sfavorire (troppo) nessuno dei potenziali utenti. In altri termini, a) tutti i quartieri devono essere serviti da un pronto soccorso e b) si deve minimizzare il tempo di percorrenza necessario all’utente più sfavorito (e cioè quello che impiega più tempo di tutti) a raggiungere il pronto soccorso.

Per poter modellare il problema di localizzazione come problema di programmazione lineare intera dobbiamo innanzitutto decidere le associazioni fra variabili ed eventi da rappresentare. In primo luogo si deve stabilire quali siti vengono scelti per localizzare i centri: associamo quindi una variabile booleana con ogni possibile sito, e cioè introduciamo una variabile booleana $x_j \in \{0, 1\}$ per $j = 1, \dots, n$, con la convenzione che:

$$x_j = \begin{cases} 1 & \text{se il sito } j \text{ è scelto} \\ 0 & \text{altrimenti.} \end{cases}$$

La seconda informazione necessaria è a quale centro viene assegnato ciascun quartiere. Introduciamo quindi una nuova variabile booleana y_{ij} per ogni quartiere $i \in \{1, \dots, m\}$ e per ogni possibile sito $j \in \{1, \dots, n\}$, ove

$$y_{ij} = \begin{cases} 1 & \text{se il quartiere } i \text{ è servito da un centro localizzato nel sito } j \\ 0 & \text{altrimenti.} \end{cases}$$

Poichè verranno costruiti esattamente p centri di pronto soccorso, sarà:

$$x_1 + x_2 + \dots + x_n = p \quad (4.1)$$

Inoltre, poichè ogni quartiere deve essere assegnato ad (esattamente) un sito, si deve avere

$$y_{i1} + y_{i2} + \dots + y_{in} = 1 \quad \text{per } i = 1, \dots, m. \quad (4.2)$$

I vincoli (4.2) assicurano che ogni quartiere sia servito da un sito. Ovviamente, affinchè un quartiere possa essere servito da un determinato sito, occorre che in tale sito sia effettivamente localizzato un centro di pronto soccorso. Questo fatto può essere espresso dai seguenti vincoli lineari:

$$y_{ij} \leq x_j \quad \text{per ogni } i = 1, \dots, m, j = 1, \dots, n. \quad (4.3)$$

Si osservi infatti che, qualora si abbia $x_j = 0$ per qualche $j \in \{1, \dots, n\}$ - e cioè nel sito j -esimo non è localizzato un centro di pronto soccorso, il vincolo (4.3) comporta $y_{ij} = 0$ per $i = 1, \dots, n$ e quindi nessun quartiere può essere assegnato a un sito non selezionato per la localizzazione di un centro.

I vincoli finora descritti assicurano che ogni quartiere sia servito da qualche sito in cui sia localizzato un centro di pronto soccorso. Per descrivere la funzione obiettivo, abbiamo bisogno di qualche considerazione (e di una variabile) aggiuntiva. In particolare, se \bar{x}, \bar{y} è una soluzione che soddisfa i vincoli (4.1), (4.2) e (4.3), qual è il tempo di percorrenza $t(i, \bar{y})$ dall' i -esimo quartiere al sito assegnatogli? Si osservi che se $\bar{y}_{ij} = 1$, il quartiere i -esimo è servito dal centro localizzato nel sito j -esimo, e quindi il tempo di percorrenza sarà c_{ij} . Quindi, grazie al vincolo (4.2), il generico tempo di percorrenza $t(i, y)$ può essere espresso come

$$t(i, y) = c_{i1}y_{i1} + c_{i2}y_{i2} + \dots + c_{in}y_{in} \quad \text{per } i = 1, \dots, m.$$

L'obiettivo è quello di trovare la soluzione (x, y) che minimizza il più grande $t(i, y)$ per $i = 1, \dots, m$. Introduciamo quindi una nuova variabile $z \in \mathbb{R}_+$ che rappresenta (un limite superiore per) il massimo tempo di percorrenza associato alla soluzione (x, y) . Poichè z deve essere (maggiore o) uguale al massimo tempo di percorrenza, sarà maggiore o uguale di ogni tempo di percorrenza $t(i, y)$ per $i = 1, \dots, m$. Quindi introduciamo i seguenti m vincoli lineari:

$$z \geq c_{i1}y_{i1} + c_{i2}y_{i2} + \dots + c_{in}y_{in} \quad \text{per } i = 1, \dots, m. \quad (4.4)$$

Per minimizzare il massimo tempo di percorrenza è sufficiente minimizzare il suo limite superiore z e la funzione obiettivo si scriverà semplicemente

$$\min z$$

Riepilogando, il modello di localizzazione può essere scritto come segue:

$$\begin{aligned}
& \min z \\
& \sum_{j=1}^n x_j = p \\
& \sum_{j=1}^n y_{ij} = 1 \quad i = 1, \dots, m \\
& z \geq \sum_{j=1}^n c_{ij} y_{ij} \quad i = 1, \dots, m \\
& x \in \{0, 1\}^n, \quad y \in \{0, 1\}^{m \times n} .
\end{aligned}$$

Esempio. Siano dati 3 siti candidati e 4 quartieri e supponiamo si vogliano localizzare 2 centri di pronto soccorso. I tempi di trasporto da quartiere a sito sono espressi nella seguente tabella

	quart. 1	quart. 2	quart. 3	quart. 4
Sito 1	7	6	7	8
Sito 2	10	10	1	1
Sito 3	9	5	4	1

Scriviamo innanzitutto il vincolo (4.1):

$$x_1 + x_2 + x_3 = 2$$

Scriviamo ora i vincoli (4.2)

$$\begin{cases} y_{11} + y_{12} + y_{13} = 1 \\ y_{21} + y_{22} + y_{23} = 1 \\ y_{31} + y_{32} + y_{33} = 1 \\ y_{41} + y_{42} + y_{43} = 1 \end{cases}$$

Scriviamo ora i vincoli (4.3)

$$\begin{cases} y_{11} \leq x_1 \\ y_{12} \leq x_2 \\ y_{13} \leq x_3 \\ y_{21} \leq x_1 \\ y_{22} \leq x_2 \\ y_{23} \leq x_3 \\ y_{31} \leq x_1 \\ y_{32} \leq x_2 \\ y_{33} \leq x_3 \\ y_{41} \leq x_1 \\ y_{42} \leq x_2 \\ y_{43} \leq x_3 \end{cases}$$

Infine, i vincoli (4.4)

$$\begin{cases} z \geq 7y_{11} + 10y_{12} + 9y_{13} \\ z \geq 6y_{21} + 10y_{22} + 5y_{23} \\ z \geq 7y_{31} + 1y_{32} + 4y_{33} \\ z \geq 8y_{41} + 1y_{42} + 1y_{43} \end{cases}$$

Si osservi che per un problema così piccolo è facile calcolare la soluzione ottima enumerando tutte le soluzioni. A tal scopo, notiamo innanzitutto che, una volta scelti i due siti ove costruire i centri, la soluzione ottima è ottenuta assegnando ciascun cliente al sito più vicino. Ad esempio, se sceglieremo il sito 1 e il sito 2 ($x_1 = 1, x_2 = 1, x_3 = 0$), ci verrà assegnare il quartiere 1 e 2 al sito 1, mentre i quartieri 3 e 4 al sito 2 ($y_{11} = y_{21} = y_{32} = y_{42} = 1$). Il quartiere più svantaggiato è il quartiere 1, con un tempo di percorrenza pari a 7. Se sceglieremo il sito 1 e il sito 3, ci verrà assegnare il quartiere 1 al sito 3,

mentre i quartieri 2, 3 e 4 al sito 2. Anche in questo caso il quartiere più svantaggiato è il quartiere 1 con tempo medio di percorrenza pari a 9. Infine, se sceglio il sito 2 e il sito 3, ci converrà assegnare i 1 e 2 al sito 3, mentre i quartieri 3 e 4 al sito 2. Anche in questo caso il quartiere più svantaggiato è il quartiere 1 con tempo medio di percorrenza pari a 9. Quindi, la soluzione che minimizza il tempo di percorrenza del quartiere più svantaggiato è la prima, in corrispondenza alla scelta dei siti 1 e 2.

Per comprendere il significato dell'introduzione della variabile z , si consideri ancora l'esempio descritto. Si è visto che la soluzione ottima corrisponde alle seguenti assegnazioni ottime per il vettore (x, y) : $x_1 = 1, x_2 = 1, x_3 = 0, y_{11} = y_{21} = y_{32} = y_{42} = 1, y_{ij} = 0$ altrimenti. Sostituendo nei vincoli (4.4), otteniamo:

$$\begin{cases} z \geq 7 \\ z \geq 6 \\ z \geq 1 \\ z \geq 1 \end{cases}$$

Quindi si deve avere $z \geq 7$ che corrisponde al tempo di percorrenza del quartiere più svantaggiato. Naturalmente, siccome si sta cercando il minimo valore di z , e non ci sono ulteriori vincoli sulla variabile z , all'ottimo avremo $z = 7$.

Questa particolare classe di problemi di programmazione lineare in cui si vuole minimizzare il massimo di una famiglia di funzioni lineari, viene detta *problema di min-max*: come visto, problemi di questo tipo possono essere risolti mediante l'introduzione di una variabile artificiale. Generalizzando leggermente, quello che abbiamo appena mostrato è che, se abbiamo un problema di min-max del tipo

$$\begin{aligned} \min \quad & (\max_i c_i^T x) \\ Ax \leq & b \\ (x & \text{ intero}), \end{aligned}$$

lo possiamo riscrivere come problema lineare introducendo una nuova variabile z :

$$\begin{aligned} \min \quad & z \\ c_i^T x \leq & z, \quad \forall i \\ Ax \leq & b \\ (x & \text{ intero}). \end{aligned}$$

Se (z^*, x^*) è una soluzione di questo problema, allora x^* è una soluzione del problema di min-max.

Un'ultima osservazione riguarda una naturale e più realistica estensione del problema di localizzazione. Infatti, molto spesso si devono aggiungere nuovi centri a un insieme di centri già attivi e localizzati sul territorio. Supponiamo ad esempio che k centri di pronto soccorso siano già localizzati nella città e che se ne vogliano attivare altri q . Allora, è possibile ancora una volta risolvere un problema di localizzazione in cui si vogliano attivare $p = k+q$ centri; nel modello, tuttavia, si porrà $x_j = 1$ per ogni sito j corrispondente a un centro già attivato.

4.1.6 Scheduling (Sequenziamento)

Concludiamo questa sezione sulle formulazioni con un esempio di utilizzazione delle variabili binarie per modellare *vincoli disgiuntivi*. Nell'usuale definizione di un problema di ottimizzazione si assume che tutti i vincoli debbano essere soddisfatti da una soluzione ammissibile. Tuttavia, in molte applicazioni capita che solo un sottoinsieme dei vincoli debba essere soddisfatto e che tale sottoinsieme sia specificato dal valore assunto da una opportuna variabile di decisione. In questo caso si dice che i vincoli sono *disgiuntivi*. Un esempio di applicazione di tali vincoli disgiuntivi è fornito dal problema di *Scheduling*.

Vincoli di tipo disgiuntivo sorgono abbastanza naturalmente in problemi di sequenziamento. In tali problemi si ha l'obiettivo di decidere l'ordine di processamento di un certo numero di lavori su una macchina a capacità unitaria. I vincoli disgiuntivi appaiono in quanto due lavori i e j non possono essere processati contemporaneamente sulla macchina e quindi *uno solo* dei vincoli (i precede j) o (j precede i) deve essere soddisfatto.

Supponiamo che debbano essere sequenziati n lavori su una macchina. Sia p_i il tempo di processamento del lavoro i sulla macchina. Poichè la macchina ha capacità unitaria, essa dovrà completare un lavoro prima di iniziare il lavoro successivo. Sia t_i l'istante in cui la macchina effettivamente inizia la lavorazione del lavoro i . Formulare il problema di scheduling consiste nel determinare vincoli sulle variabili t_i in modo tale che esse rappresentino sequenze effettivamente realizzabili sulla macchina.

Se il lavoro i è iniziato sulla macchina prima del lavoro j , dobbiamo avere $t_j \geq t_i + p_i$. D'altra parte, se il lavoro j inizia prima del lavoro i , dobbiamo avere $t_i \geq t_j + p_j$. Sia α un numero positivo molto grande e sia $y_{ij} = 1$ se i precede j e $y_{ij} = 0$ se j precede i . Considera il seguente sistema di vincoli:

$$\begin{aligned} \alpha y_{ij} + t_i - t_j &\geq p_j, & 1 \leq i < j \leq n \\ \alpha(1 - y_{ij}) + t_j - t_i &\geq p_i, & 1 \leq i < j \leq n \end{aligned}$$

Osserviamo che se $y_{ij} = 1$ (cioè se i precede j) allora il primo vincolo è sempre soddisfatto poichè $\alpha > p_j + t_j - t_i$, mentre il secondo vincolo esprime la condizione che la lavorazione di j può iniziare solo dopo il completamento di i . Una situazione analoga si avrà quando $y_{ij} = 0$.

Se un vettore (t, y) con $t \in \mathbb{R}^n$ ed $y \in \{0, 1\}^{n \times n}$ soddisfa questo sistema allora, per quanto detto, ciascuna componente del vettore t rappresenta un istante ammissibile di inizio processamento per il corrispondente lavoro. Viceversa, per ogni vettore ammissibile t esiste sicuramente un vettore y (che rappresenta l'ordine di processamento dei lavori sulla macchina) tale che il vettore (t, y) è ammissibile per il precedente sistema di vincoli. Vincoli di precedenza o altre restrizioni temporali possono essere facilmente inseriti nel modello aggiungendo vincoli lineari sulle variabili t ed y .

4.2 Tecniche di soluzione per problemi di PLI

In forma del tutto generale un problema di PLI può essere scritto come

$$\begin{aligned} \max \quad & c^T x \\ \text{a} \quad & Ax \leq b \\ & 0 \leq x \leq U \\ & x \text{ intera} \end{aligned} \tag{PLI}$$

dove U è un vettore a n componenti che limita superiormente il valori che possono essere assunti dalle variabili. La presenza di questo vincolo implica che *la regione ammissibile del problema PLI è limitata*. Questa assunzione semplificherà le considerazioni svolte in questo paragrafo. Essa è, peraltro, non limitativa, in quanto in ogni problema reale il valore assunto dalle componenti delle soluzioni ammissibili è sempre limitato superiormente da qualche costante.

In questo paragrafo consideriamo alcune tecniche risolutive per problemi di PLI. In particolare, dopo alcune brevi considerazione sulla soluzione per enumerazione totale e sulla soluzione approssimata per arrotondamento di problemi di PLI, descriveremo una delle più note famiglie di algoritmi per la soluzione esatta di problemi di PLI: gli algoritmi di *Branch and Bound*.

Prima di procedere vogliamo introdurre un concetto molto importante, quello di *problema rilassato*, che verrà ampiamente ripreso e utilizzato nel seguito. Dato il problema PLI si definisce suo *rilassamento lineare* (o *problema rilassato associato*) il seguente problema.

$$\begin{aligned} \max \quad & c^T x \\ \text{a} \quad & Ax \leq b \\ & 0 \leq x \leq U. \end{aligned} \tag{PR}$$

Si tratta cioè di un problema di PL ottenuto eliminando dal problema PLI il vincolo di interezza. La regione di un Problema di PLI è ovviamente contenuta in quella del suo rilassamento PR (che è ottenuto, lo ripetiamo, *eliminando* un vincolo da PLI) e quindi ne segue che il valore ottimo di PR è sicuramente maggiore o uguale di quello di PLI. In particolare, se la soluzione ottima di PR è intera allora questa è anche la soluzione ottima di PLI.

Per chiarire ulteriormente quanto detto facciamo un semplice esempio informale. Supponiamo di voler trovare lo studente più alto della facoltà di economia e commercio. Si tratta di trovare lo studente (“punto ammissibile”) che massimizza la funzione obiettivo che associa ad ogni studente la sua altezza. Consideriamo poi il problema di trovare lo studente più alto di tutta l’università (“problema rilassato”). Ovviamente l’insieme ammissibile di questo nuovo problema è più ampio di quello del problema precedente, comprendendo tutti gli studenti di economia e commercio più gli studenti di tutte le altre facoltà. Se noi risolviamo il “problema rilassato” il valore ottimo che troviamo (altezza dello studente più alto di tutta l’università) è ovviamente maggiore o tutt’al più uguale al valore ottimo del problema di trovare lo studente più alto della facoltà di economia e commercio. Se poi risulta che lo studente più alto di tutta l’università è proprio uno studente della facoltà di economia e commercio, allora è ovvio che questo studente è anche “la soluzione ottima” del primo problema, è cioè lo studente più alto della facoltà di economia e commercio.

4.2.1 Soluzione per enumerazione totale

Nelle ipotesi fatte la regione ammissibile di PLI è costituita da un insieme finito di punti: l’insieme di punti a coordinate intere contenuti nell’insieme costituito dai punti che soddisfano i vincoli espressi dalle disequazioni lineari di PLI (che coincide con l’insieme ammissibile di PR) è limitato per ipotesi. In linea di principio è quindi sempre possibile risolvere PLI calcolando il valore della funzione obiettivo in ogni punto ammissibile e scegliendo quello che la massimizza. Definiremo questo metodo **Enumerazione Totale**. Se la cardinalità dell’insieme ammissibile di PLI è piccola allora l’enumerazione totale non solo è possibile, ma è certamente il modo migliore di risolvere un problema di PLI. Se, viceversa, la cardinalità dell’insieme delle soluzioni ammissibili è molto grande, l’enumerazione totale diviene non proponibile.

Osserviamo che se abbiamo un problema di PLI con dieci variabili e supponiamo che ognuna di queste variabili possa assumere dieci valori diversi, il numero di possibili punti ammissibili è di 10^{10} , cioè 10 miliardi. Questo semplice esempio mostra che l’enumerazione totale è raramente utile nei casi pratici, dove spesso il numero di variabili intere è dell’ordine delle centinaia se non delle migliaia.

4.2.2 Soluzione approssimata per arrotondamento

Abbiamo già avuto modo di osservare, nell’introduzione di questo paragrafo, che se la soluzione ottima del rilassamento lineare di PLI, \bar{x} , è intera allora \bar{x} è la soluzione ottima di PLI. Se, invece, anche una sola delle componenti di \bar{x} non è intera \bar{x} non è nemmeno ammisibile per PLI.

Si può allora essere tentati di concludere che la soluzione ottima di PLI può essere ottenuta *arrotondando* \bar{x} a un punto a coordinate intere “vicino” a \bar{x} .

Questo metodo di arrotondamento può essere un metodo pratico di grande utilità per la soluzione di PLI, se tutte variabili del problema sono variabili intere che rappresentano il numero di oggetti indivisibili usati o prodotti, ed è ragionevole aspettarsi che le variabili stesse abbiano valori abbastanza grandi all’ottimo. Inoltre i vincoli lineari dovrebbero essere tali da poter facilmente decidere se l’ammissibilità è preservata arrotondando una variabile non intera al più vicino valore intero (o, per esempio, al più vicino valore intero inferiore). Per esempio, se x_1 rappresenta il numero di automobili di un certo modello che devono essere assemblate durante un mese e se esiste una limitazione inferiore su x_1 , è ragionevole aspettarsi che un valore $\bar{x}_1 = 1000.1$ possa essere arrotondato a 1001 senza che venga violata l’ammissibilità. Se tutte le variabili del problema sono di questo tipo e se esiste un metodo semplice di passare dalla soluzione non intera \bar{x} di PR a una soluzione “vicina”, intera e ammisibile per PLI, allora

un ragionevole approccio pratico alla soluzione di PLI è quello di risolvere il suo rilassamento lineare ed arrotondare quindi la soluzione.

Tuttavia, se il valore delle variabili della soluzione ottima del rilassamento lineare è presumibilmente piccolo, questa tecnica non è molto sensata. Se, per esempio, x_1 rappresenta il numero di portaaeri che devono essere costruite dall'esercito italiano, è intuitivo capire che è molto difficile decidere se si debba arrotondare $\bar{x}_1 = 2.5$ a 2 o a 3, e che le due possibilità portano a risultati completamente diversi. La situazione diventa ancora più drammatica quando le variabili intere sono, in realtà, variabili binarie. Abbiamo visto nel primo paragrafo di questo capitolo che questa è una situazione abbastanza frequente. In questi problemi le variabili 0-1 sono variabili che indicano quale di due possibili scelte alternative deve essere attuata. In questo caso l'arrotondamento è totalmente privo di senso logico. Se $\bar{x} \in \mathbb{R}^n$, possono esistere 2^n vettori interi ottenuti arrotondando le componenti non intere di \bar{x} al valore intero superiore o inferiore più prossimo, e non si vede con quale criterio si possa sceglierne uno quale soluzione di PLI. Inoltre molti (se non tutti) di questi 2^n vettori possono non essere ammissibili per PLI, e il problema di determinare un vettore a componenti intere, ammissibile per PLI, e "vicino" a \bar{x} può essere, in generale, un problema di difficoltà paragonabile alla soluzione di PLI.

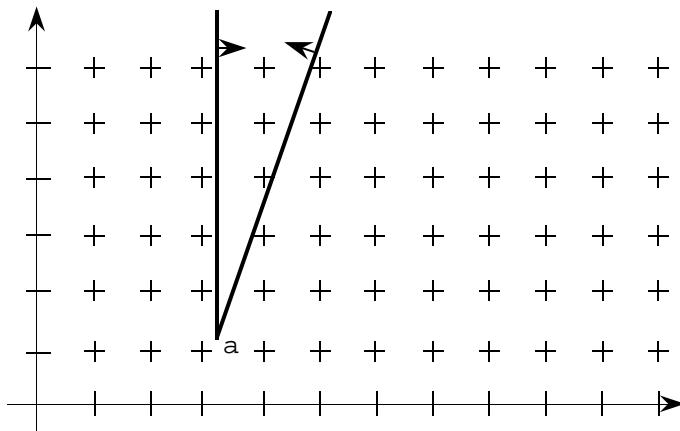


Figura 4.2: Soluzione per arrotondamento: primo esempio.

Consideriamo l'esempio di figura 4.2. Supponiamo che la soluzione del rilassamento lineare sia a ($\bar{x} = a$). Il punto a ha tutte le componenti non intere e tutti i punti a componenti intere ottenuti arrotondando le sue componenti all'intero immediatamente superiore o inferiore non sono ammissibili per PLI. Da questo esempio dovrebbe risultare chiaro, inoltre, che è possibile costruire problemi PLI in cui la soluzione ottima del relativo rilassamento lineare è lontana "quanto si vuole" dal più vicino punto ammissibile di PLI.

Anche nel caso in cui possa essere trovato un punto ammissibile "vicino" a \bar{x} , questi può essere molto lontano dalla soluzione ottima di PLI.

Un esempio al riguardo è dato nella figura 4.3, dove la soluzione ottima del rilassamento lineare è a , b è la soluzione ottenuta per arrotondamento, mentre c è la vera soluzione di PLI.

Tutte le considerazioni svolte finora mostrano chiaramente la necessità di algoritmi esatti ed efficienti per la soluzione di PLI. Un possibile algoritmo di questo genere viene illustrato nella sezione seguente.

4.2.3 La Tecnica del Branch and Bound

Abbiamo già visto che una possibile, semplice tecnica di soluzione per PLI, purtroppo quasi mai utilizzabile in pratica, è l'enumerazione totale.

Il "Branch and Bound" (BB) fornisce, allora, una metodologia di ricerca della soluzione ottima che effettua una esplorazione *parziale* dell'insieme delle soluzioni ammissibile. In particolare la funzione obiet-

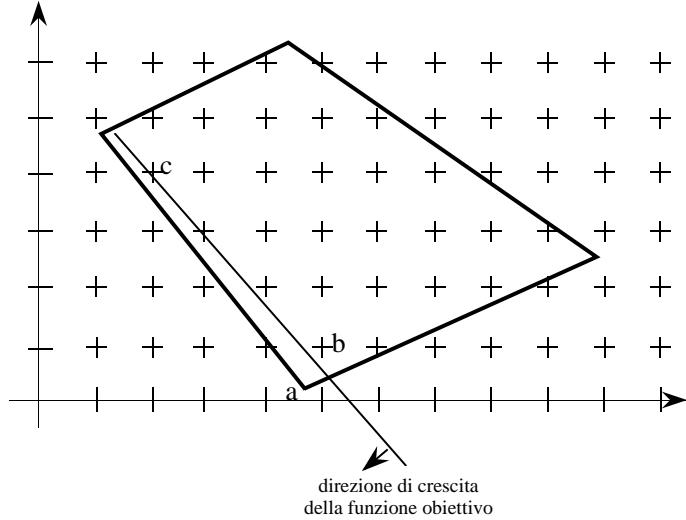


Figura 4.3: Soluzione per arrotondamento: secondo esempio.

tivo viene calcolata per una sottoinsieme (piccolo) delle soluzioni ammissibili con la proprietà di contenere almeno una soluzione ottima. Descriviamo ora in maggior dettaglio i vari aspetti della procedura. Sia S_0 l'insieme delle soluzioni ammissibili PLI. Una *partizione* di S_0 è una famiglia di sottoinsiemi (S_1, \dots, S_r) , $r \geq 2$ tale che:

$$S_i \cap S_j = \emptyset \quad \text{per ogni coppia} \quad 1 \leq i < j \leq r$$

e

$$\bigcup_{i=1}^r S_i = S_0$$

Evidentemente, la soluzione ottima del problema originario è data dal massimo tra i valori $z_1^* = c^T x_1^*, \dots, z_r^* = c^T x_r^*$ dove x_i^* è la soluzione ottima del sottoproblema PLI_i definito dal problema di massimizzare la funzione obiettivo del problema PLI sulla regione ammissibile S_i (nel seguito, con un piccolo abuso di notazione, indicheremo spesso con S_i sia la regione ammissibile sia il sottoproblema PLI_i). Se un certo sottoproblema PLI_i risulta, a sua volta, di difficile soluzione si partiziona ulteriormente l'insieme S_i producendo nuovi sottoproblemi ed iterando la procedura fino a che il problema originario non risulti decomposto in problemi elementari di facile soluzione.

Cerchiamo di chiarire la procedura con un esempio informale. Supponiamo di dover trovare la persona più alta dell'università. Risolvere questo problema confrontando le altezze di tutti gli studenti fra loro (enumerazione totale) può risultare in un processo troppo lungo. Si può allora chiedere ai presidi di ogni facoltà qual è lo studente più alto delle loro facoltà e scegliere il più alto tra questi studenti. Abbiamo decomposto il problema originario in più sottoproblemi: la determinazione dello studente più alto di ogni facoltà. In alcuni casi la risoluzione di questi sottoproblemi può essere agevole. Per esempio il preside della facoltà di Noia Applicata non ha difficoltà alcuna ad individuare lo studente più alto tra i suoi *tre* iscritti. Anche il preside della facoltà di Bel Turpiloquo non ha difficoltà ad individuare il suo studente più alto, avendo appena dovuto rompere le porte di tutte le aule del primo anno per permettere (come da regolamento) il passaggio senza chinare la testa della matricola Amilcare Basso. In altri casi il preside può non essere in grado di fornire facilmente il nome dello studente più alto. In questo caso si può scomporre ulteriormente il problema. Per esempio, il preside della facoltà di Dilapidazione e Latrocincio chiede a tutti i direttori dei suoi istituti di fornirgli il nome dello studente più alto che frequenta l'istituto stesso. Da questi egli potrà così risalire al nome dello studente più alto della facoltà. E così via.

È evidente che la procedura appena descritta non compie una totale enumerazione dell'insieme S ed è computazionalmente efficiente solo se il numero dei sottoproblemi generati si mantiene estremamente limitato e quindi solo se la strategia di soluzione dei sottoproblemi è sufficientemente efficace.

In generale però, risolvere un sottoproblema può essere altrettanto gravoso che risolvere il problema originario. È per questo che, in luogo della soluzione esatta del problema S_i si preferisce calcolare un “upper bound” di z_i^* e cioè un valore $U_i \geq z_i^*$. Tale valore viene poi confrontato con il miglior valore \tilde{z} (ad un certo passo della procedura) della funzione obiettivo calcolata in un punto ammissibile (**ottimo corrente**). Se il valore approssimato risulta non superiore a quello dell'ottimo corrente, ovvero se

$$\tilde{z} \geq U_i \geq z_i^*$$

si deduce che non esiste nell'insieme S_i un punto in cui la funzione obiettivo abbia un valore migliore di \tilde{z} . Un tale risultato ci permette di sospendere l'esame dell'insieme S_i e di eliminarlo dalla lista dei sottoproblemi da risolvere.

Per tornare all'esempio informale precedente, supponiamo che il preside della facoltà di Bel Torpilo-quio abbia comunicato che Amilcare Basso è alto 2 metri e 10 centimetri (ottimo corrente). Supponiamo anche che il preside della facoltà di Occultismo, pur senza sapere esattamente qual è l'altezza esatta del suo studente più alto, sia riuscito ad avere la certezza, con mezzi solo a lui noti, che ogni studente della sua facoltà non è alto più di 2 metri (upper bound). È evidente che è inutile fare lo sforzo di cercare di trovare l'altezza esatta del più alto studente della facoltà di Occultismo, tanto Amilcare Basso è comunque più alto.

Evidentemente la soluzione del problema originario sarà tanto più efficiente quanto migliori saranno i valori degli “upper bound” ed a loro volta tali valori approssimeranno tanto meglio il valore ottimo del sottoproblema quanto più efficace sarà stata la decomposizione del problema originario. Di conseguenza l'efficienza della tecnica del BB dipende essenzialmente dalla qualità delle due strategie che ne caratterizzano la struttura:

- (a) **Strategia di Soluzione**, ovvero la strategia per il calcolo di un valore che approssimi per eccesso (in un problema di massimo) il valore ottimo di un sottoproblema.
- (b) **Strategia di Separazione**, ovvero la strategia per la scelta della partizione dell'insieme delle soluzioni ammissibili di un sottoproblema.

È anche evidente che a seconda della strategia di soluzione e di quella di separazione adottate, lo schema generale appena descritto si concretizzerà in un algoritmo differente.

Nel seguito di questo paragrafo descriveremo un possibile schema (semplificato) di BB che corrisponde a uno degli schemi di BB più utilizzati nella pratica dando al contempo maggiore sistematicità al metodo presentato.

Ricordiamo che vogliamo risolvere un problema di PLI della seguente forma

$$\begin{aligned} \max \quad & c^T x \\ \text{a.s.} \quad & Ax \leq b \\ & 0 \leq x \leq U \\ & x \text{ intera} \end{aligned} \tag{PLI}$$

Questo problema, nel seguito sarà indicato anche come problema S_0 . Nel corso del metodo considereremo dei sottoproblemi che hanno una struttura simile. Più precisamente considereremo sottoproblemi S_i con la seguente struttura.

$$\begin{aligned} \max \quad & c^T x \\ \text{a.s.} \quad & Ax \leq b \\ & l_i \leq x \leq u_i \\ & x \text{ intera,} \end{aligned} \tag{S_i}$$

dove i vettori n dimensionali l_i e u_i sono calcolati dall'algoritmo. Nel seguito indicheremo, inoltre, con x_i^* e z_i^* rispettivamente una soluzione ottima e il valore ottimo di S_i .

La strategia che proponiamo per calcolare un upper bound U_i di z_i^* è la seguente:

dato il problema S_i , risolvere il suo rilassamento lineare.

Indichiamo con y^i la soluzione ottima del rilassamento lineare di S_i e con U_i il corrispondente valore ottimo. Per quanto visto precedentemente il valore ottimo U_i così ottenuto costituisce un upper bound di z_i^* , inoltre il rilassamento lineare è facilmente risolubile con il metodo del simplex.

Dobbiamo ora descrivere un metodo per la separazione di un generico S_i . Supponiamo di aver risolto il rilassamento lineare di S_i e sia y^i la sua soluzione ottima e, come già detto, con U_i il corrispondente valore ottimo. Se y^i ha tutte componenti intere allora $y^i = x_i^*$ e il problema non va separato. Se U_i è minore o uguale all'*ottimo corrente*, cioè al valore massimo della funzione obiettivo finora trovato, il problema non può dar origine ad una soluzione intera migliore di quella corrente e non è necessario separarlo per trovare la sua soluzione ottima intera. Supponiamo quindi che nessuno di questi due casi si sia verificato, e vediamo come separare questo sottoproblema. Sia y_j^i una componente non intera del vettore y^i (almeno una ne deve esistere), α_j la sua parte intera inferiore (ossia il più grande intero minore o uguale a $(x_i^*)_j$) e β_j la sua parte intera superiore (ossia il più piccolo intero maggiore o uguale a y_j^i). Ovviamente valgono le seguenti relazioni:

$$\begin{aligned}\alpha_j &< \beta_j \\ \beta_j - \alpha_j &= 1.\end{aligned}$$

Separiamo il problema S_i nei seguenti due problemi

$$\begin{aligned}\max \quad & c^T x \\ \text{A}x &\leq b \\ l_i^1 &\leq x \leq u_i^1 \\ x &\text{ intera,}\end{aligned}$$

dove

$$\begin{aligned}(l_i^1)_j &= (l_i)_j \quad \text{se } i \neq j \quad \text{e} \quad (l_i^1)_j = \beta_j \quad \text{se } i = j \\ u_i^1 &= u_i\end{aligned}$$

e

$$\begin{aligned}\max \quad & c^T x \\ \text{A}x &\leq b \\ l_i^2 &\leq x \leq u_i^2 \\ x &\text{ intera,}\end{aligned} \tag{S_i}$$

dove

$$\begin{aligned}(u_i^2)_j &= (u_i)_j \quad \text{se } i \neq j \quad \text{e} \quad (u_i^2)_j = \alpha_j \quad \text{se } i = j \\ l_i^2 &= l_i\end{aligned}$$

Notiamo che è facile convincersi che l'unione delle regioni ammissibili di questi due problemi dà la regione ammissibile del problema S_i , di modo tale che questa è in effetti una separazione del problema S_i del tipo prima considerato.

Per concludere vogliamo osservare che dobbiamo considerare un altro caso che finora non abbiamo menzionato. Nel risolvere il rilassamento del problema S_i può accadere che la regione ammissibile del rilassamento risulti vuota. Siccome la regione ammissibile del problema S_i è contenuta in quella del suo rilassamento, questo vuol dire che il problema S_i ha regione ammissibile vuota, e quindi non c'è bisogno di separarlo.

Abbiamo ora tutti gli elementi per descrivere il metodo di BB proposto.

1. Inizializzazione

Inizialmente il problema $S_0 = PLI$ è l'unico problema candidato. La procedura inizia con l'applicazione della strategia di soluzione ad S_0 ottenendo un primo “upper bound” U_0 .

Inoltre, viene calcolato un “**lower bound**” \tilde{z} per il problema S_0 , cioè un valore sicuramente non superiore a quello z^* della soluzione ottima. Tale valore viene usualmente ottenuto in corrispondenza ad una soluzione ammissibile \tilde{x} facile da individuare (ad es. mediante una semplice euristica). I valori \tilde{x} e \tilde{z} vengono successivamente aggiornati nel corso della procedura non appena vengono individuate soluzioni ammissibili per il problema originario con un valore più alto della funzione obiettivo. Se non è possibile individuare facilmente una soluzione ammissibile si può porre $\tilde{z} = -\infty$, mentre \tilde{x} viene, per il momento lasciato indefinito.

Se l'ottimo del problema rilassato y^0 appartiene all'insieme S_0 il problema è risolto. Altrimenti è necessario applicare una strategia di separazione al problema S_0 ottenendo due nuovi problemi candidati S_1 ed S_2 .

2. Passo generico dell'algoritmo.

Ad un generico passo dell'algoritmo supponiamo di avere la lista $\mathcal{L} = \{S_1, \dots, S_q\}$ dei sottoproblemi che non sono ancora stati esaminati (**problemi terminali**). Illustriamo ora brevemente le possibili azioni dell'algoritmo in un passo generico.

2.1. Scelta del problema aperto da esaminare

Si estrae un problema dalla lista \mathcal{L} , diciamo S_i , che diventa il *problema corrente*. Se la lista \mathcal{L} è vuota, l'algoritmo termina e \tilde{x} , la soluzione ottima corrente, è la soluzione ottima del problema. Se la lista \mathcal{L} non è vuota possono essere adottate diverse strategie per la scelta del problema S_i da esaminare. Diamo qui due esempi:

- (1) *Scelta con criterio di priorità FIFO (First In First Out)*. In questo caso i sottoproblemi terminali sono gestiti dalla procedura secondo lo schema a *coda*. In particolare, il problema scelto è il primo problema generato nella fase di separazione (tra quelli presenti nella lista).
- (2) *Scelta con criterio di priorità LIFO (Last In First Out)*. In questo caso i sottoproblemi terminali sono gestiti dalla procedura secondo lo schema a *stack*. In particolare, il problema scelto è l'ultimo problema generato nella fase di separazione.

Si risolve il rilassamento lineare di S_i . Se S_i è vuoto si elimina e il passo dell'algoritmo è terminato. Si può quindi passare ad esaminare un nuovo problema della lista \mathcal{L} .

Se S_i non è vuoto si calcolano y^i e U_i , rispettivamente la soluzione ottima e il valore ottimo del problema rilassato (si noti che siccome stiamo supponendo che la regione ammissibile del problema rilassato è limitata, e avendo appena verificato che non è vuota, per il teorema fondamentale della Programmazione Lineare, il rilassamento del problema in esame deve avere una soluzione ottima y^i).

2.1. Confronto con l'ottimo corrente

Se $U_i \leq \tilde{z}$, detto z_i^* il valore ottimo della funzione obiettivo per il problema S_i , abbiamo che:

$$z_i^* \leq U_i \leq \tilde{z}$$

e quindi che nessuna soluzione *migliore di \tilde{z}* può essere contenuta in S_i . In tal caso il problema S_i può essere eliminato della lista dei problemi candidati e viene definito **chiuso** e il passo dell'algoritmo è terminato. Si può quindi passare ad esaminare un nuovo problema della lista \mathcal{L} .

2.3. Aggiornamenti

Arrivati a questo punto sono possibili due casi, che esaminiamo separatamente. Ricordiamo che y^i è l'ottimo del problema rilassato, cioè il punto che soddisfa $U_i = c^T y^i$.

2.3.1. y^i è intero

Se l'ottimo del problema rilassato y^i ha componenti intere abbiamo che esso è ottimo anche per il sottoproblema S_i . Inoltre poiché $U_i > \tilde{z}$, y^i è una soluzione ammissibile del problema originario con valore della funzione obiettivo ($c^T y^i = U_i$) migliore di \tilde{z} . Di conseguenza, possiamo porre $\tilde{x} = y^i$ e $\tilde{z} = U_i$. **(aggiornamento dell'ottimo corrente)**. Il problema S_i viene chiuso.

2.3.2. y^i non è intero

In tal caso si deve applicare la strategia di separazione precedentemente illustrata, produrre i due nuovi sottoproblemi ed aggiungerli alla lista \mathcal{L} dei problemi terminali.

In tutt'e due i casi il passo dell'algoritmo è terminato e si può passare ad esaminare un nuovo problema della lista \mathcal{L} . Si riporta di seguito lo schema a blocchi dell'algoritmo appena descritto. \square

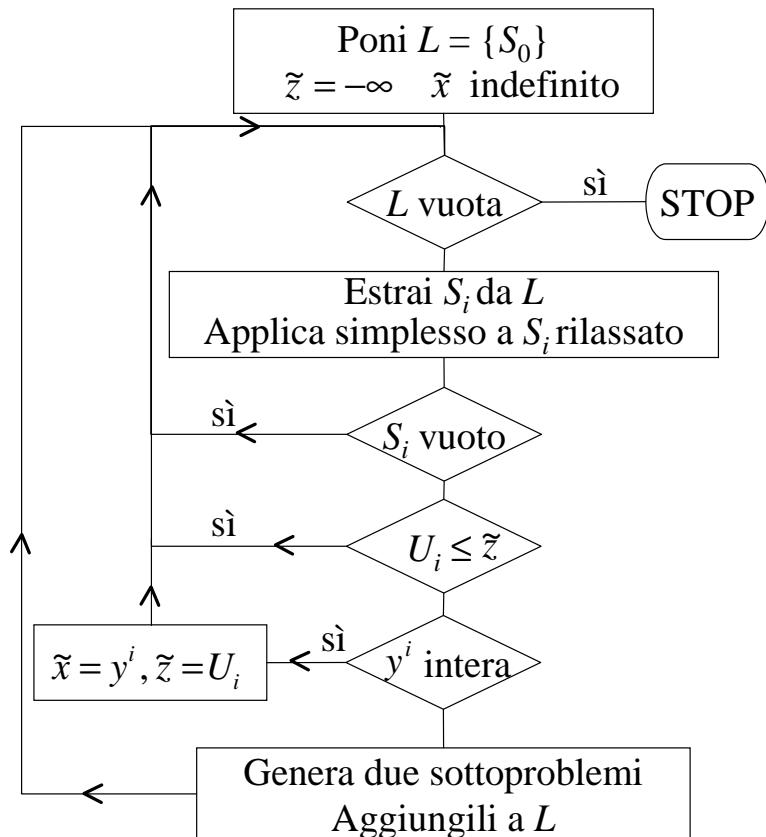


Figura 4.4: Schema di branch and bound

4.3 Esempi

Completiamo il capitolo con degli esempi di applicazione del Branch and Bound.

Esempio1. Sia dato il seguente problema di programmazione lineare a numeri interi⁴:

$$(S_0) \quad \begin{aligned} & \max \quad -x_1 + 2x_2 \\ & -4x_1 + 6x_2 \leq 9 \\ & x_1 + x_2 \leq 4 \\ & x \geq 0 \\ & x_1, x_2 \in \mathbb{Z} \end{aligned}$$

Il problema è un problema di massimizzazione, quindi le soluzioni dei problemi rilassati forniranno dei limiti superiori ("upper bound") al valore della soluzione ottima intera. Inoltre, il valore della funzione obiettivo calcolato in corrispondenza ad una soluzione ammissibile fornirà un limite inferiore ("lower bound"). Si noti che, benché non siano esplicitamente presenti degli upper bound sulle variabili (del tipo $x \leq U$), il problema ha ovviamente una regione ammissibili limitata (infatti le variabili sono non negative e la loro somma deve essere minore o uguale a quattro, si veda il secondo vincolo). È facile convincersi quindi, che nulla cambia nell'applicazione dell'algoritmo e nella sua validità.

Poniamo inizialmente $\tilde{z} = -\infty$, l'ottimo corrente è per il momento non definito. Poniamo il problema S_0 nella lista \mathcal{L} .

Viene selezionato un problema in $\mathcal{L} = \{S_0\}$. In particolare, viene estratto dalla lista il problema S_0 , unico problema nella lista.

La soluzione ottima y^0 del rilassamento lineare di S_0 è $y^0 = (3/2, 5/2)^T$ (si veda la figura Figura 4.5) e fornisce un "upper bound" pari a $U_0 = c^T y^0 = 7/2$.

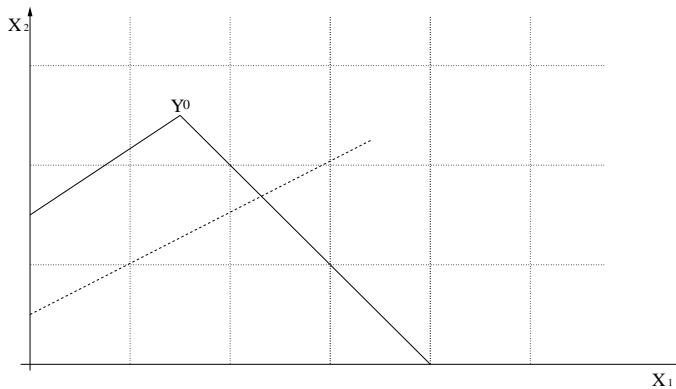


Figura 4.5: Problema S_0 .

Non è possibile chiudere il problema S_0 in quanto la soluzione y^0 non è intera e l' "upper bound" è strettamente maggiore del "lower bound" che vale al momento $-\infty$. Siccome la soluzione ottima del rilassamento non è intera, si procede alla separazione di S_0 . Possiamo separare rispetto ad una qualsiasi delle variabili non intere, per semplicità decidiamo di separare rispetto alla variabile con indice più piccolo (x_1). A tale scopo, aggiungiamo ai vincoli della formulazione di S_0 , i due vincoli $x_1 \leq 1$ ($x_1 \leq \lfloor y_1^0 \rfloor$ ⁵) e

⁴Ricordiamo che con \mathbb{Z} si indica l'insieme dei numeri interi

⁵Ricordiamo che $\lfloor y_1^0 \rfloor$ indica il più grande numero intero minore o uguale a y_1^0

$x_1 \geq 2$ ($x_1 \geq \lceil y_1^0 \rceil^6$), ottenendo i due problemi seguenti :

$$(S_1) \quad \begin{aligned} \max \quad & -x_1 + 2x_2 \\ \text{subject to} \quad & -4x_1 + 6x_2 \leq 9 \\ & x_1 + x_2 \leq 4 \\ & x_1 \leq 1 \\ & x \geq 0 \\ & x_1, x_2 \in \mathbb{Z} \end{aligned}$$

$$(S_2) \quad \begin{aligned} \max \quad & -x_1 + 2x_2 \\ \text{subject to} \quad & -4x_1 + 6x_2 \leq 9 \\ & x_1 + x_2 \leq 4 \\ & x_1 \geq 2 \\ & x \geq 0 \\ & x_1, x_2 \in \mathbb{Z} \end{aligned}$$

che vengono messi in \mathcal{L} : $\mathcal{L} = \{S_1, S_2\}$. Estraiamo un problema da \mathcal{L} , per esempio, S_2 . Risolviamo il rilassamento di S_2 (si veda la figura Figura 4.6), e otteniamo $y^2 = (2, 2)^T$ e un corrispondente $U_2 =$

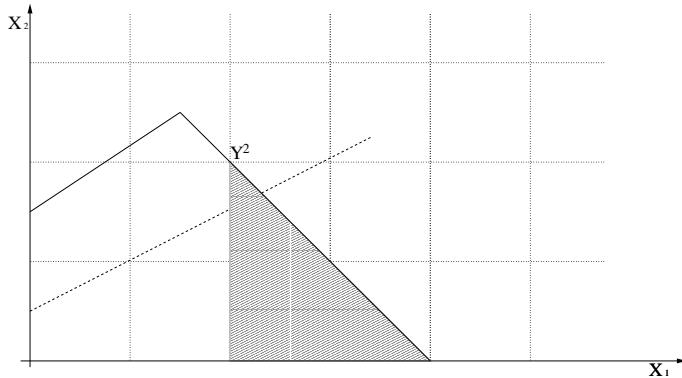


Figura 4.6: Problema S_2 .

$c^T y^2 = 2$. Siccome la soluzione del problema rilassato è intera e il valore ottimo corrispondente è migliore dell'ottimo corrente \tilde{z} , aggiorniamo e poniamo: $\tilde{z} = 2$ e $\tilde{x} = (2, 2)^T$. Il problema S_2 è chiuso e ora $\mathcal{L} = \{S_1\}$.

Estraiamo un problema da \mathcal{L} , siccome c'è solo S_1 esaminiamo S_1 . Risolviamo il rilassamento di S_1 (si veda la figura Figura 4.7) e otteniamo $y^1 = (1, 13/6)^T$ e un corrispondente $U_1 = c^T y^1 = 10/3$. Siccome $U_1 > \tilde{z} = 2$ e y^1 non è intera, generiamo due sottoproblemi e li mettiamo in \mathcal{L} . Poiché solo x_2 è frazionaria, possiamo fare il branching solo su x_2 . Otteniamo così i due problemi:

$$(S_3) \quad \begin{aligned} \max \quad & -x_1 + 2x_2 \\ \text{subject to} \quad & -4x_1 + 6x_2 \leq 9 \\ & x_1 + x_2 \leq 4 \end{aligned}$$

⁶Ricordiamo che $\lceil y_1^0 \rceil$ indica il più piccolo numero intero maggiore o uguale a y_1^0

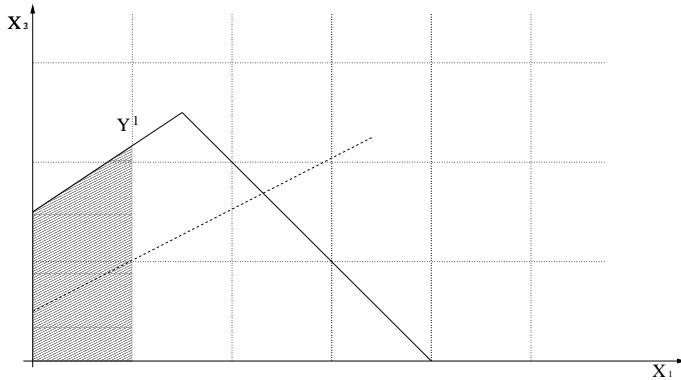


Figura 4.7: Problema S_1 .

$$\begin{aligned} x_1 &\leq 1 \\ x_2 &\leq 2 \\ x &\geq 0 \\ x_1, x_2 &\in \mathcal{Z} \end{aligned}$$

$$\begin{aligned} (S_4) \quad \max \quad & -x_1 + 2x_2 \\ & -4x_1 + 6x_2 \leq 9 \\ & x_1 + x_2 \leq 4 \\ & x_1 \leq 1 \\ & x_2 \geq 3 \\ & x \geq 0 \\ & x_1, x_2 \in \mathcal{Z} \end{aligned}$$

A questo punto $\mathcal{L} = \{S_3, S_4\}$. Esaminiamo il problema S_4 . Trovo che il suo rilassamento lineare è vuoto. Chido il problema. Estraggo dalla lista \mathcal{L} l'unico problema presente, S_3 . Il suo rilassamento lineare ha per soluzione $y^3 = (3/4, 2)^T$ (si veda la figura Figura 4.8), con un corrispondente valore $U_3 = c^T y^3 = 13/4$.

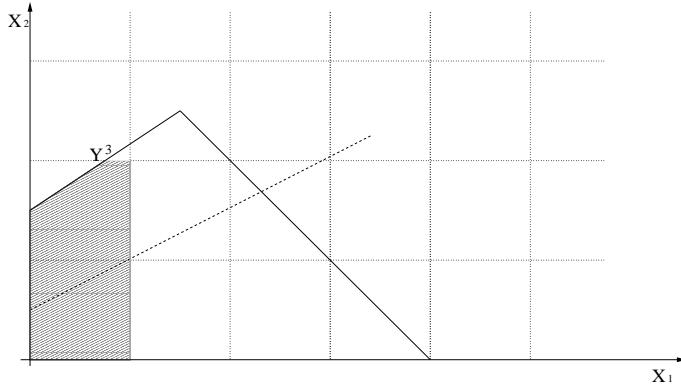


Figura 4.8: Problema S_3 .

Siccome $U_3 > \tilde{z}$ e y^3 non è intero, genero due sottoproblemi a partire da S_3 facendo branching sull'unica variabile frazionaria, x_1 :

$$(S_5) \quad \begin{aligned} \max \quad & -x_1 + 2x_2 \\ \text{subject to} \quad & -4x_1 + 6x_2 \leq 9 \\ & x_1 + x_2 \leq 4 \\ & x_1 \leq 1 \\ & x_1 \leq 0 \\ & x_2 \leq 2 \\ & x \geq 0 \\ & x_1, x_2 \in \mathbb{Z} \end{aligned}$$

$$(S_6) \quad \begin{aligned} \max \quad & -x_1 + 2x_2 \\ \text{subject to} \quad & -4x_1 + 6x_2 \leq 9 \\ & x_1 + x_2 \leq 4 \\ & x_1 \leq 1 \\ & x_1 \geq 1 \\ & x_2 \leq 2 \\ & x \geq 0 \\ & x_1, x_2 \in \mathbb{Z} \end{aligned}$$

Notiamo che in S_5 i due vincoli $x_1 \leq 1$ è reso superfluo dal nuovo vincolo $x_1 \leq 0$. Abbiamo lasciato il vincolo superfluo per rendere chiaro che ogni volta che si compie un'operazione di branching per generare due nuovi sottoproblemi che eridatano tutti i vincoli del problema che li genera e a cui vengono aggiunti i vincoli specifici del branching. Analogamente, per S_6 , i due vincoli $x_1 \leq 1$ e $x_1 \geq 1$ dicono che $x_1 = 1$, ma abbiamo preferito lasciare i due vincoli distinti.

A questo punto $\mathcal{L} = \{S_5, S_6\}$. Esaminiamo il problema S_6 . Il suo rilassamento lineare ha per soluzione $y^6 = (1, 2)^T$ (si veda la figura Figura 4.9), con un corrispondente valore $U_6 = c^T y^6 = 3$. Siccome

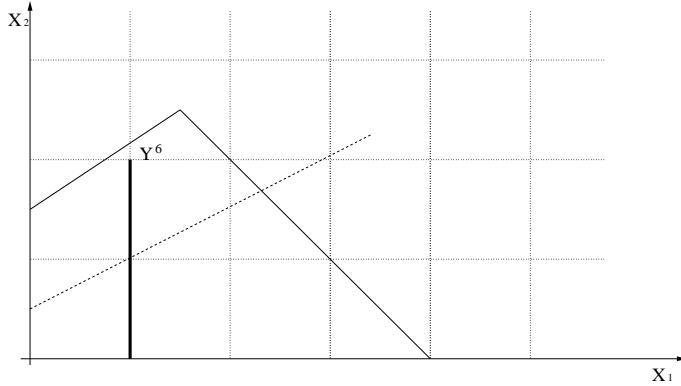


Figura 4.9: Problema S_6 .

$U_6 > \tilde{z} = 2$ e y^6 è intero, aggiorniamo e poniamo: $\tilde{z} = 3$ e $\tilde{x} = (1, 2)^T$. Il problema S_6 è chiuso e ora $\mathcal{L} = \{S_5\}$.

Estraggo il problema S_5 dalla lista dei problemi aperti e ne risolvo il rilassamento. Ottengo $y^5 = (0, 3/2)$ e $U_5 = c^T y^5 = 3$ (si veda la figura Figura 4.10). Siccome $U_5 \leq \tilde{z} = 3$, chiudo il problema.

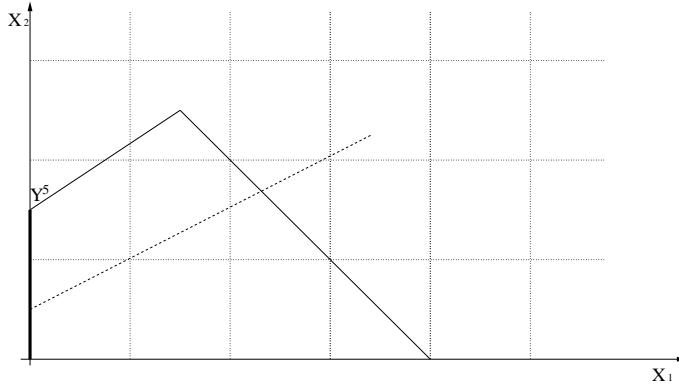


Figura 4.10: Problema S_5 .

Siccome la lista è ora vuota, il punto $(1, 2)^T$, ottimo corrente, è la soluzione del problema.

I vari passi della procedura di soluzione possono essere schematizzati utilizzando il cosiddetto *Albero di Enumerazione* riportato in figura 4.11. nodi indicano i problemi mentre i rami stabiliscono le relazioni padre-figlio (generatore-generato) nel processo di generazione.

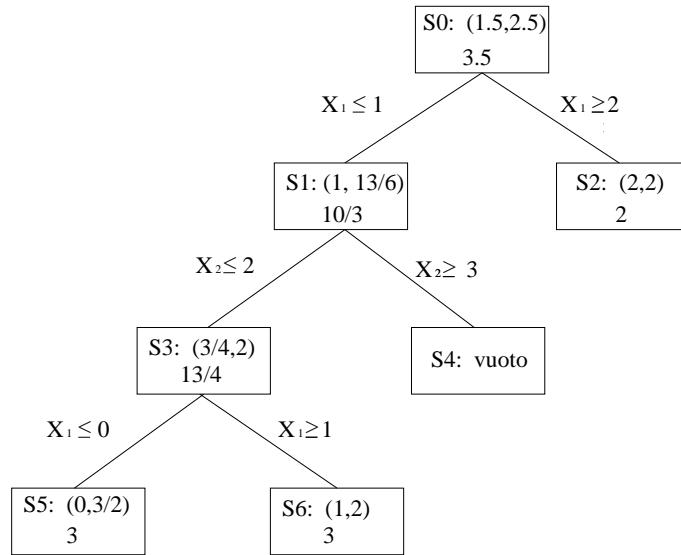


Figura 4.11: Albero di enumerazione.

Notiamo che la strategia adottata in questo caso, nello scegliere i problemi nella lista è stata quella LIFO.

Esempio 2. Passiamo ora ad esaminare, più sinteticamente, il secondo esempio. Sia dato il seguente problema di programmazione lineare a numeri interi.

$$\begin{aligned}
 \max \quad & x_1 + x_2 \\
 \text{s.t.} \quad & 2x_1 + 5x_2 \leq 30 \\
 & 4x_1 - 3x_2 \leq 6 \\
 & 0 \leq x_1 \\
 & 0 \leq x_2 \\
 & x_1, x_2 \in \mathbb{Z}
 \end{aligned} \tag{S_0}$$

Notiamo che in anche in questo problema non sono esplicitamente presenti gli “upper bound” sulle variabili. Essendo comunque la regione ammissibile del problema rilassato limitato, non è necessario inserirli esplicitamente.

Poniamo inizialmente $\tilde{z} = -\infty$ e risolviamo il rilassamento lineare di S_0 . Si ottiene

$$U_0 = \frac{114}{13} \quad y_1^0 = \frac{60}{13} \quad y_2^0 = \frac{54}{13}.$$

Si elimina S_0 dalla lista \mathcal{L} , e vi si aggiungono i due nuovi problemi S_1 e S_2 individuati dai seguenti vincoli aggiuntivi.

$$\begin{aligned}
 S_1 : \quad & x_1 \leq 4 \\
 S_2 : \quad & x_1 \geq 5.
 \end{aligned}$$

Si avrà pertanto $\mathcal{L} = \{S_1, S_2\}$.

Si sceglie S_1 da L e lo si risolve il suo rilassamento. Si ottiene

$$U_1 = \frac{42}{5} \quad y_1^1 = 4 \quad y_2^1 = \frac{22}{5}$$

Essendo $U_1 > \tilde{z}$ si elimina S_1 da \mathcal{L} e vi si aggiungono, invece altri due problemi, S_3 e S_4 , individuati dai seguenti vincoli aggiuntivi.

$$\begin{aligned}
 S_3 : \quad & x_1 \leq 4 \quad x_2 \leq 4 \\
 S_4 : \quad & x_1 \leq 4 \quad x_2 \geq 5.
 \end{aligned}$$

Risulta pertanto $\mathcal{L} = \{S_2, S_3, S_4\}$.

Si sceglie S_3 da L e lo si risolve il suo rilassamento. Si ottiene

$$U_3 = 8 \quad y_1^3 = 4 \quad y_2^3 = 4$$

Essendo $U_3 > \tilde{z}$ ed essendo intera la soluzione trovata, si pone

$$\tilde{z} = U_3 = 8 \quad \tilde{x}_1 = y_1^3 = 4 \quad \tilde{x}_2 = y_2^3 = 4,$$

e si elimina il problema S_3 . Risulta a questo punto $\mathcal{L} = \{S_2, S_4\}$.

Si sceglie da \mathcal{L} S_4 e lo si risolve il suo rilassamento. Si ottiene

$$U_4 = \frac{15}{2} \quad y_1^4 = \frac{5}{2} \quad y_2^4 = 5$$

Essendo $U_4 < \tilde{z}$ si chiude S_4 e la lista diventa $\mathcal{L} = \{S_2\}$.

Si sceglie S_2 dalla lista e lo si risolve il suo rilassamento. Si vede che non esistono soluzioni ammissibili. Si chiude S_2 e la lista L è vuota. La soluzione corrente è la soluzione ottima.

$$z^* = 8 \quad x_1^* = 4 \quad x_2^* = 4.$$

Capitolo 5

Grafi: nozioni fondamentali

In questo capitolo introduciamo la nozione di grafo ed alcune definizioni ad essa collegate. I grafi sono una struttura matematica molto usata nelle applicazioni e si prestano a rappresentare problemi apparentemente molto diversi tra di loro con un linguaggio semplice ed unificato. Questo spiega la loro importanza nella matematica applicata e, in particolare, nella Ricerca Operativa.

5.1 Definizioni fondamentali

Un *grafo non orientato* $G = (V, E)$ è definito da un insieme finito $V(G) = \{v_1, \dots, v_n\}$ di elementi detti *nodi* o *vertici* e da un insieme $E(G) = \{e_1, \dots, e_m\} \subseteq V \times V$ di coppie non ordinate di nodi dette *archi* o *spigoli*.

Dato l'arco $e = (v, w) = (w, v)$, i nodi v e w sono detti *estremi* di e , e si dice che l'arco e *incide* su u e v . Una comoda rappresentazione del grafo viene mostrata in figura 5.1a. I nodi sono rappresentati da cerchi,

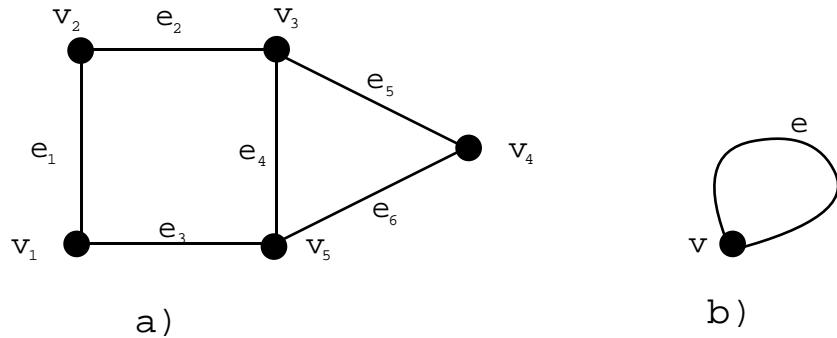


Figura 5.1: a) Grafo non-orientato b) Loop

mentre gli archi sono tratti di curva congiungenti i due estremi. Per il grafo $G = (V, E)$ rappresentato in figura si ha

$$V = \{v_1, v_2, v_3, v_4, v_5\}$$

$$E = \{e_1, e_2, e_3, e_4, e_5, e_6\} = \{(v_1, v_2), (v_2, v_3), (v_1, v_5), (v_3, v_5), (v_3, v_4), (v_4, v_5)\}.$$

Due nodi u, v sono detti *adiacenti* (reciprocamente) se l'arco (u, v) appartiene ad E . Nella figura sono adiacenti, per esempio, i nodi v_1 e v_2 . Due archi sono detti *adiacenti* se hanno un estremo in comune, come gli archi e_2 ed e_4 in figura.

Si definisce *intorno* di un nodo v in G , indicato con $N(v)$, l'insieme dei nodi adiacenti a v . Nella figura $N(v_5) = \{v_1, v_3, v_4\}$. Un nodo v si dice *isolato* se $N(v) = \emptyset$. Si definisce *stella* di v in G , indicata con $\delta(v)$, l'insieme degli archi incidenti su v . Nella figura $\delta(v_5) = \{e_3, e_4, e_6\}$.

Si definisce *sottografo* di $G = (V, E)$ un grafo $H = (W, F)$ tale che $W \subseteq V$ e $F \subseteq E$. Si definisce *sottografo indotto* da $W \subseteq V$ in $G = (V, E)$ il grafo $H = (W, F)$, ove l'insieme degli archi F è tale che l'arco (u, v) appartiene a F se e solo se: (a) u e v appartengono a W e (b) $(u, v) \in E$. Informalmente possiamo dire che il sottografo H eredita tutti gli archi di G i cui estremi sono entrambi contenuti nel sottoinsieme W . In figura 5.2 vengono mostrati due sottografi del grafo di figura 5.1. Il primo non è un sottografo indotto ("manca" l'arco e_4), mentre il secondo è indotto in G dall'insieme di nodi $\{v_2, v_3, v_4, v_5\}$.

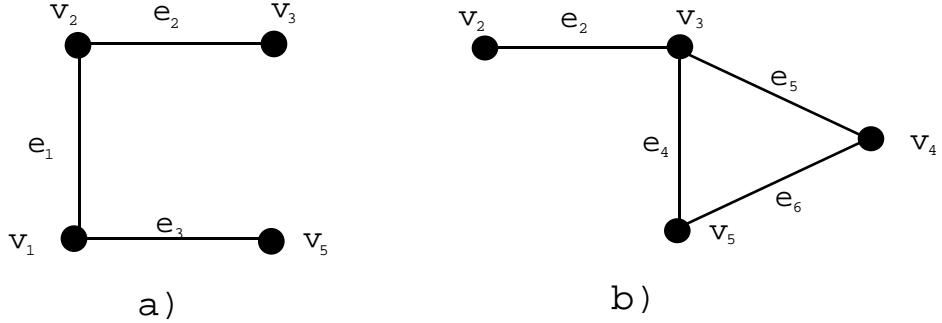


Figura 5.2: a) Sottografo b) Sottografo indotto

Un *grafo orientato* $G = (V, E)$ è definito da un insieme finito $V(G) = \{v_1, \dots, v_n\}$ di elementi detti *nodi* e da un insieme $E(G) = \{e_1, \dots, e_m\} \subseteq V \times V$ di coppie **ordinate** di nodi dette *archi*. Dato l'arco $e = (v, u)$, il primo nodo nella coppia (v) è detto *coda*, e si dice che l'arco e è *uscente* da v . Il secondo nodo (u) è detto *testa*, e si dice che l'arco e è *entrante* in u . L'arco si dice *orientato* dal nodo v al nodo u . I nodi v e u sono detti *estremi* di e , e si dice che l'arco e *incide* su v e u . Notiamo che a differenza del caso non orientato, l'arco (v, u) è distinto dall'arco (u, v) . Una rappresentazione del grafo orientato viene mostrata in figura 5.3. L'unica differenza con la rappresentazione del grafo non orientato sta nella presenza sull'arco di una freccia che indica l'orientamento dell'arco stesso.

Analogamente al caso non-orientato, si definisce *stella* di v in G , indicata con $\omega(v)$, l'insieme degli archi incidenti su v . In figura 5.3, $\omega(v_1) = \{e_1, e_2, e_5\}$. La stella $\omega(v)$ può essere partizionata in *stella entrante* $\omega^-(v)$, cioè l'insieme degli archi entranti in v , e in *stella uscente* $\omega^+(v)$, l'insieme degli archi uscenti da v . Nell'esempio, $\omega^-(v_1) = \{e_2\}$, mentre $\omega^+(v_1) = \{e_1, e_5\}$. Sottografi e sottografi indotti sono definiti come per il caso non orientato.

Cammini, Cammini Orientati, Cicli. Dato un grafo $G = (V, E)$ (indifferentemente orientato o non orientato), ove $V = \{v_1, \dots, v_n\}$ e $E = \{e_1, \dots, e_m\}$ diremo *cammino* in (di) G , un insieme ordinato $P = \{v_{j_0}, e_{h_1}, v_{j_1}, e_{h_2}, \dots, v_{j_{p-1}}, e_{h_p}, v_{j_p}\}$ con: $j_k \in \{1, \dots, n\}$, $h_i \in \{1, \dots, m\}$, per $k = 0, \dots, p$ e $i = 1, \dots, p$, e con l'arco e_{h_i} incidente sui nodi $v_{j_{i-1}}$ e v_{j_i} ¹. Notiamo che nel caso del grafo orientato, l'orientamento degli archi del cammino è indifferente. Si richiede semplicemente che l'arco e_{h_i} incida sui

¹Spesso, quando non c'è possibilità di ambiguità, un cammino viene indicato indicando la sequenza dei soli nodi o quella dei soli archi

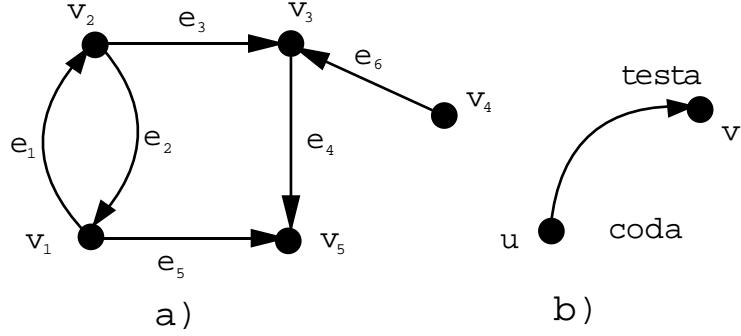


Figura 5.3: a) Grafo orientato b) Arco orientato

nodi $v_{j_{i-1}}$ e v_{j_i} , senza specificare quale dei due nodi sia la testa e quale la coda di e_{h_i} . I nodi v_{j_0} e v_{j_p} sono detti *estremi* del cammino P . Un cammino è detto *semplice* se gli archi e i nodi che lo definiscono sono tutti distinti.

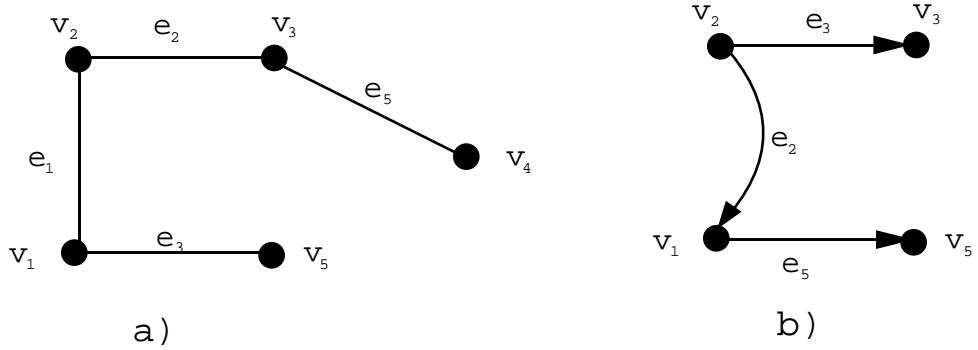


Figura 5.4: Cammini semplici

In figura 5.4a è mostrato un cammino semplice del grafo non orientato di figura 5.1, $P = \{v_5, e_3, v_1, e_1, v_2, e_2, v_3, e_5, v_4\}$, mentre in figura 5.4b è mostrato il cammino semplice del grafo orientato di figura 5.3, $P = \{v_5, e_5, v_1, e_2, v_2, e_3, v_3, e_4, v_5\}$. In figura 5.5a è mostrato il cammino (non semplice) del grafo non orientato di figura 5.1, $P = \{v_1, e_1, v_2, e_2, v_3, e_5, v_4, e_6, v_5, e_4, v_3, e_5, v_4\}$. Notiamo che se in un grafo esiste un cammino fra i nodi u e v , esisterà un cammino **semplice** fra u e v .

In un grafo orientato, chiameremo *cammino orientato* un cammino $P = \{v_{j_0}, e_{j_1}, v_{j_1}, \dots, e_{j_p}, v_{j_p}\}$ tale che $e_{j_i} = (v_{j_{i-1}}, v_{j_i})$. Nel cammino orientato è dunque importante anche l'orientamento degli archi: il nodo che precede l'arco nel cammino deve esserne la coda, mentre il nodo che succede all'arco deve esserne la testa. In figura 5.5b è mostrato il cammino orientato del grafo orientato di figura 5.3, $P = \{v_1, e_1, v_2, e_3, v_3, e_4, v_5\}$. Se u e v sono, rispettivamente, il primo e l'ultimo nodo di un cammino orientato P , si dirà che P *va da* u a v ².

²Spesso, quando si tratta di grafi orientati e non ci sia possibilità di ambiguità, si indica col nome di cammino quello che più propriamente dovrebbe essere chiamato cammino orientato

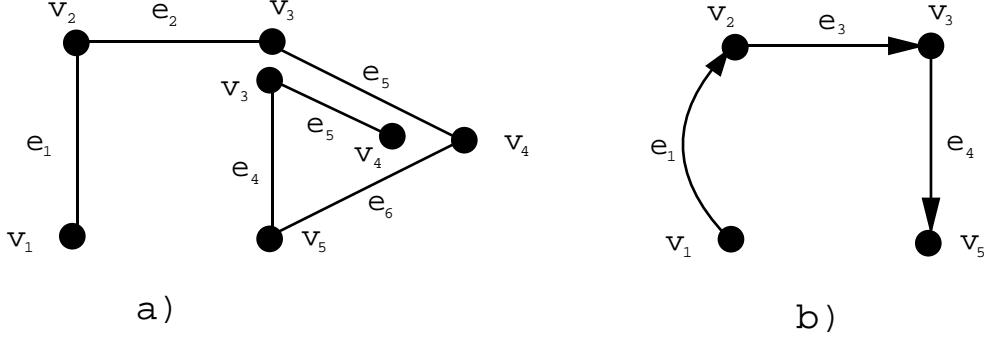


Figura 5.5: a) Cammino (non semplice) b) Cammino orientato

Un cammino è detto *chiuso* se i suoi estremi coincidono. Definiremo *ciclo* un cammino chiuso $C = \{v_{j_0}, e_{h_1}, v_{j_1}, \dots, v_{h_{p-1}}, v_{h_p}, v_{j_0}\}$, tale che $\{v_{j_0}, e_{h_1}, v_{j_1}, \dots, e_{h_{p-1}}, v_{h_p}\}$ è un cammino semplice. Il cammino chiuso $P = \{v_1, e_1, v_2, e_2, v_3, e_3, v_4, e_4, v_2, e_5, v_5, e_6, v_1\}$ di figura 5.6a non è un ciclo perché il cammino $P = \{v_1, e_1, v_2, e_2, v_3, e_3, v_4, e_4, v_2, e_5, v_5\}$ non è semplice (il nodo v_2 è contenuto due volte). In figura 5.6b è mostrato un ciclo. Notiamo che ancora una volta l'orientamento degli archi nei grafi orientati è indifferente. Per i grafi orientati si definisce *ciclo orientato* un cammino orientato chiuso $C = \{v_{j_0}, e_{h_1}, v_{j_1}, \dots, e_{h_{p-1}}, v_{h_p}, e_{h_p}, v_{j_0}\}$, tale che $\{v_{j_0}, e_{h_1}, v_{j_1}, \dots, e_{h_{p-1}}, v_{h_p}\}$ è un cammino orientato semplice (Fig. 5.6c)

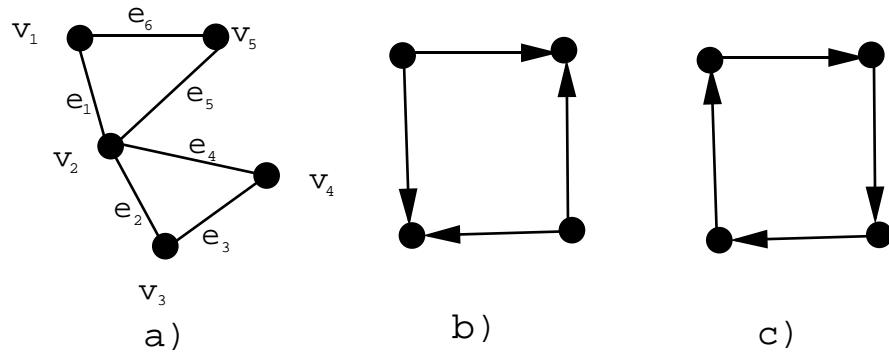


Figura 5.6: a) Cammino chiuso b) Ciclo c) Ciclo orientato

Componenti Connesse. Dato un grafo $G = (V, E)$ (orientato o non orientato), un nodo $v \in V$ si dice connesso a un nodo $u \in V$ se esiste un cammino (non necessariamente orientato) tra v e u . La relazione di connessione è una relazione binaria, ed è facile vedere che gode delle seguenti proprietà:

1. v è connesso a v (RIFLESSIVITÀ).
2. Se v è connesso a u , allora u è connesso a v (SIMMETRIA).
3. Se v è connesso a u , u è connesso a z , allora v è connesso a z (TRANSITIVITÀ).

Quindi la relazione *essere connesso a* è una relazione di equivalenza, che definisce una partizione dell'insieme dei nodi V in classi di equivalenza C_1, \dots, C_q . Si ha dunque $V = C_1 \cup C_2 \cup \dots \cup C_q$, e $C_i \cap C_j = \emptyset$, per $i \neq j$. Il grafo indotto in G dai nodi appartenenti a una classe d'equivalenza C_i è detto *componente connessa* di G . Un grafo è detto *connesso* se possiede una sola componente connessa. In figura 5.7 è mostrato un grafo non orientato formato da due componenti connesse (notare che $|V| = 6$)³.

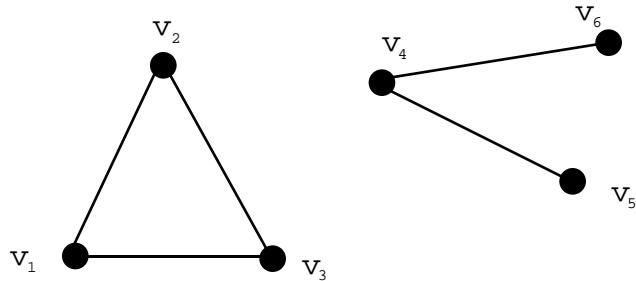


Figura 5.7: Componenti connesse

Dato un grafo orientato $G = (V, E)$, un nodo $v \in V$ è *fortemente connesso* a un nodo $u \in V$ se esistono due cammini orientati in G , uno da v ad u e l'altro da u ad v . Anche in questo caso è facile vedere che la relazione di forte connessione è una relazione d'equivalenza che definisce le classi C_1, \dots, C_q nell'insieme dei nodi V . Il grafo indotto in G dai nodi appartenenti a una classe C_i è detto *componente fortemente connessa* di G . Un grafo si dice *fortemente connesso* se possiede una sola componente fortemente connessa. In figura 5.8a è mostrato un grafo orientato e in figura 5.8b le sue tre componenti fortemente connesse.

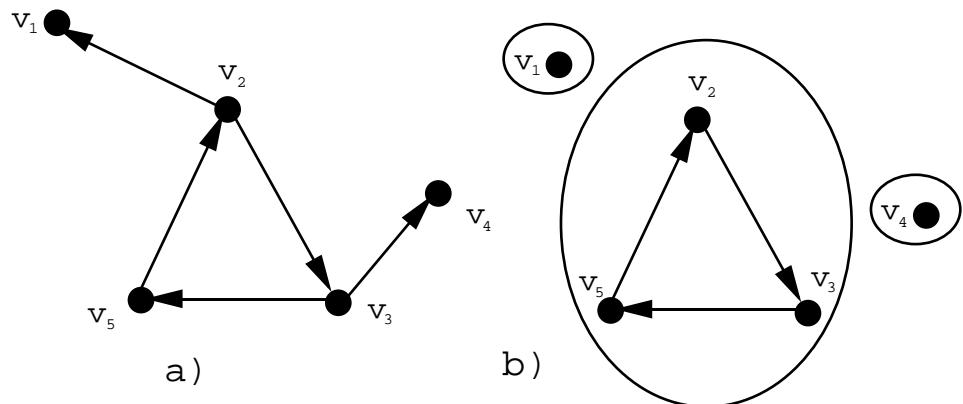


Figura 5.8: a) Grafo orientato b) Componenti fortemente connesse

Alberi. Un grafo (orientato o non orientato) è detto *aciclico* se non contiene cicli (non necessariamente orientati) come sottografi. Si dice *albero* un grafo aciclico e connesso. Ogni grafo aciclico è l'unione di

³Con il simbolo $|I|$, dove I è un insieme finito di elementi, si indica la cardinalità dell'insieme I , cioè il numero di elementi che compongono l'insieme stesso.

uno o più alberi e viene detto *foresta*. In figura 5.9a è mostrato un albero, in figura 5.9b una foresta.

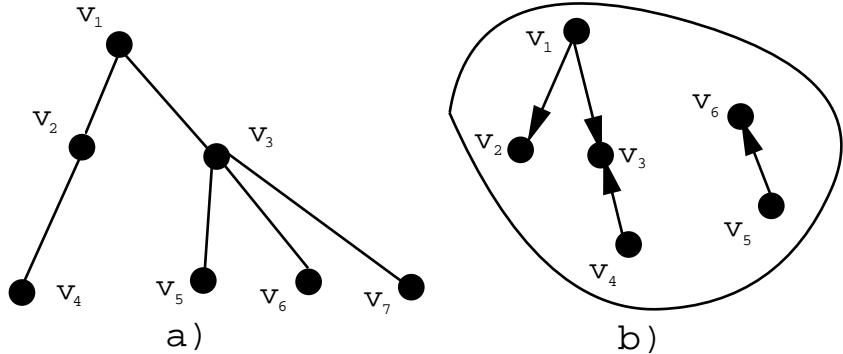


Figura 5.9: a) Albero b) Foresta

Dato un albero $T = (V, E)$, con $|V| \geq 2$, diremo *foglia* un nodo $v \in V$ tale che $|\delta(v)| = 1$ ($|\omega(v)| = 1$). Ad esempio, in figura 5.9a sono foglie v_4, v_5, v_6, v_7 .

Valgono i due seguenti risultati, che diamo senza dimostrazione. Notiamo, in particolare, che il secondo lemma fornisce una serie di definizioni alternative di albero.

Proposizione 5.1.1 *Le seguenti asserzioni sono equivalenti:*

1. $G = (V, E)$ è un albero.
2. Ogni coppia di nodi è connessa da un unico cammino.

Dato un grafo connesso $G = (V, E)$ diremo *albero ricoprente* di G un albero $T = (W, F)$ con $W = V$ e $F \subseteq E$.

Grafi bipartiti Un grafo $G = (V, E)$ non orientato è detto *bipartito* se è possibile trovare una partizione V_1, V_2 dei suoi nodi⁴ tale che gli estremi di ogni arco appartengono uno a V_1 e uno a V_2 . Nella figura 5.10 è riportato un esempio di grafo bipartito.

Si puo dimostrare che vale la seguente caratterizzazione dei grafi bipartiti

Proposizione 5.1.2 *Un grafo è bipartito se e solo se non contiene cicli di lunghezza dispari.*

Spesso un grafo bipartito viene indicato, mettendo in evidenza la partizione dell'insieme dei nodi, come $G = (V_1, V_2, E)$.

5.2 Rappresentazioni di un grafo

Un grafo $G(V, E)$ pu essere rappresentato in molti modi diversi. Nelle pagine precedenti abbiamo già incontrato due modi diversi: la rappresentazione estensiva e quella grafica.

Nella rappresentazione estensiva vengono elencati tutti gli elementi dell'insieme V e tutti quelli dell'insieme E , nella rappresentazione grafica, invece, il grafo viene rappresentato mediante un disegno in cui ai nodi si fanno corrispondere piccoli cerchi e agli archi tratti di curva congiungenti i cerchi. Già da

⁴Deve risultare, cioè, $V_1 \cup V_2 = V$ e $V_1 \cap V_2 = \emptyset$

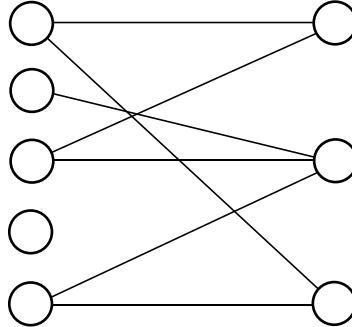


Figura 5.10: Grafo bipartito

questi due esempi si capisce che, benché concettualmente equivalenti, questi modi diversi di rappresentare un grafo possano avere caratteristiche molto differenti. Per esempio, la rappresentazione grafica è molto immediata e ci dà subito una visione intuitiva delle caratteristiche del grafo, e molta della terminologia introdotta finora è ispirata proprio dalla rappresentazione grafica (si pensi, ad esempio, alla definizione di albero e foresta). Tuttavia, se, come spesso accade nella pratica, il grafo ha moltissimi nodi (per esempio migliaia) si capisce subito che la rappresentazione grafica non è più di pratica utilità.

Si capisce allora come sia utile avere a disposizione diversi modi di rappresentare un grafo. La decisione di scegliere una rappresentazione piuttosto che un'altra andrà poi fatta di volta in volta in base al problema che si vuole risolvere, alle dimensioni del grafo, alle sue caratteristiche etc.

Passiamo allora a descrivere tre possibili modi di rappresentare un grafo.

1. matrice di adiacenze;
2. matrice di incidenza;
3. lista di adiacenze.

La prima e la terza modalità sono le più diffuse per la memorizzazione di grafi.

La prima è preferibile nel caso di grafi con un elevato numero di archi rispetto al numero di nodi (per esempio quando $m \approx n^2$) in quanto, pur essendo in questo caso particolare la prima e la terza modalità equivalenti dal punto di vista dell'efficienza computazionale, la rappresentazione in termini di matrice di adiacenze è più immediata e fornisce una rappresentazione del grafo più efficace.

La terza è preferibile nel caso di grafi “sparsi”, in cui cioè il numero di archi è piccolo rispetto al numero di nodi (per esempio quando $m \approx n$, come negli alberi).

La seconda, pur se meno efficiente delle altre dal punto di vista computazionale, è preferibile in alcune applicazioni della Ricerca Operativa per la sua corrispondenza ad alcune formulazioni standard di modelli di ottimizzazione e la conseguente maggior facilità di analisi del modello.

Matrice di adiacenze. Questa modalità di rappresentazione è basata su una matrice quadrata $n \times n$. Il generico elemento (i, j) della matrice sarà pari a 1 se l'arco (i, j) del grafo esiste, sarà pari a 0 se l'arco (i, j) non esiste.

In questo modo si possono memorizzare grafi sia orientati che non orientati. Nel caso di grafi non orientati la matrice di adiacenze è simmetrica.

Matrice di incidenza. Questa modalità di rappresentazione è basata su una matrice $n \times m$. La matrice ha quindi un numero di righe pari al numero di nodi e un numero di colonne pari al numero di archi.

Nel caso di grafi non orientati, il generico elemento (i, j) della matrice sarà pari a 1 se il j -esimo arco del grafo incide sul nodo i , sarà pari a 0 se l'arco j -esimo non incide sul nodo i .

Nel caso di grafi orientati, il generico elemento (i, j) della matrice sarà pari a -1 se l'arco j -esimo esce dal nodo i , sarà pari a 1 se l'arco j -esimo entra nel nodo i , sarà pari a 0 se l'arco j -esimo non incide sul nodo i .

Nella matrice di incidenza ogni colonna ha esattamente due elementi diversi da zero; essi sono in corrispondenza delle due righe della matrice relative ai due nodi estremi dell'arco.

Lista di adiacenze. Questa modalità di rappresentazione è basata su una lista di adiacenze del grafo, in cui per ogni nodo del grafo vengono elencati i nodi adiacenti.

Questa rappresentazione è in genere la più efficiente dal punto di vista computazionale, specialmente per grafi con pochi archi.

Completiamo questa sezione con un esempio. Consideriamo il grafo $G(V, E)$ definito nel seguente modo:

$$V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$$

$$E = \{e_1 = (v_1, v_2), e_2 = (v_1, v_3), e_3 = (v_3, v_2), e_4 = (v_2, v_4), e_5 = (v_3, v_5), e_6 = (v_4, v_5), e_7 = (v_4, v_6)\}$$

Spesso, nel riportare la rappresentazione estensiva di un grafo, si identificano, per comodità, i vertici con i primi n numeri naturali. In questo caso la rappresentazione estensiva del grafo diventa

$$V = \{1, 2, 3, 4, 5, 6\}$$

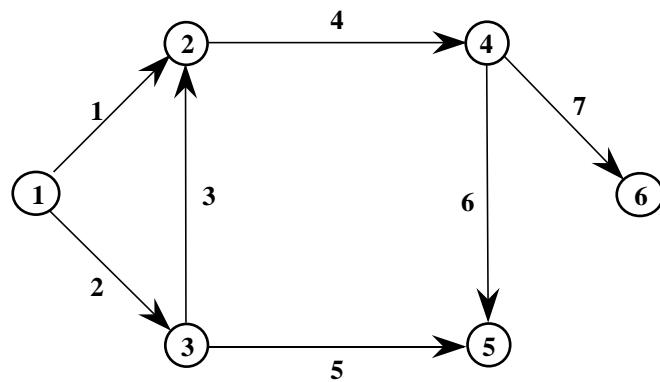
$$E = \{(1, 2), (1, 3), (3, 2), (2, 4), (3, 5), (4, 5), (4, 6)\},$$

e spesso ci si limita a fornire il valore di n , cioè il numero di nodi e la lista degli archi. In figura 5.11 sono riportate le rappresentazioni di questo grafo con le varie tecniche appena descritte.

5.3 Alcuni esempi

In questa ultima sezione verranno formulati alcuni problemi utilizzando il formalismo dei grafi. Scopo di questi esempi non è quello di dare una panoramica completa di quei problemi la cui formulazione (e risoluzione) è basata sui grafi (non sarebbe sufficiente un intero corso dedicato a questo tema!). Piuttosto vogliamo mostrare come problemi molto diversi tra loro possano essere formulati (e risolti) usando una rappresentazione basata sui grafi, che spesso contribuisce a rendere chiari problemi apparentemente molto intricati, dando così, inoltre, una prova dell'importanza della scelta di un buon modello nella risoluzione di un problema e della necessità di disporre di un ampio spettro di modelli per poter adeguatamente modellare problemi pratici.

Scambio dei cavalli. Sia data una scacchiera 3×3 in cui le caselle siano numerate con i numeri 1,2,3,4,5,6,7,8,9 come in figura 5.12, due cavalli bianchi nelle caselle 1 e 3, due cavalli neri nelle caselle 7 e 9. Il problema è quello di spostare i due cavalli bianchi al posto di quelli neri e viceversa, spostando un cavallo per volta secondo le modalità delle mosse di scacchi (per esempio il cavallo nella casella 1 in una mossa potrà andare solo nelle caselle 6 e 8, il cavallo nella casella 7 potrà andare nelle caselle 2 e 6, ecc.) e facendo in modo che due cavalli non occupino mai contemporaneamente la stessa casella. Questo problema, apparentemente intricato, diventa semplice non appena viene formulato come problema su grafi. Associamo ad ogni casella della scacchiera un nodo di un grafo. Due nodi sono collegati da un arco se (e solo se) è possibile passare con un cavallo dall'una all'altra delle corrispondenti caselle. Il grafo risultante è quello disegnato in figura 5.12 (da cui ovviamente si osserva come non sia mai possibile passare attraverso la casella 5). Osservando il grafo si individua immediatamente la strategia da seguire per ottenere la soluzione desiderata: basta far circolare i cavalli sul grafo (in questo caso formato da un unico ciclo) in modo da non sovrapporli mai, fino a quando ogni cavallo non ha raggiunto il posto desiderato. Si osservi che il minimo numero di mosse necessario per raggiungere l'obiettivo è pari a 24.



Rappresentazione grafica

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Matrice di connessione

$$\begin{bmatrix} -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Matrice di incidenza

1 : 2 , 3
 2 : 4 .
 3 : 2 , 5
 4 : 5 , 6
 5 :
 6 :

archi da 1 a 2 e da 1 a 3
 arco da 2 a 4
 archi da 3 a 2 e da 3 a 5
 archi da 4 a 5 e da 4 a 6
 nessun arco
 nessun arco

Lista di adiacenze

Figura 5.11: Rappresentazioni di un grafo

1	2	3
4	5	6
7	8	9

CB		CB
CN		CN

CB = Cavallo bianco
 CN = Cavallo nero

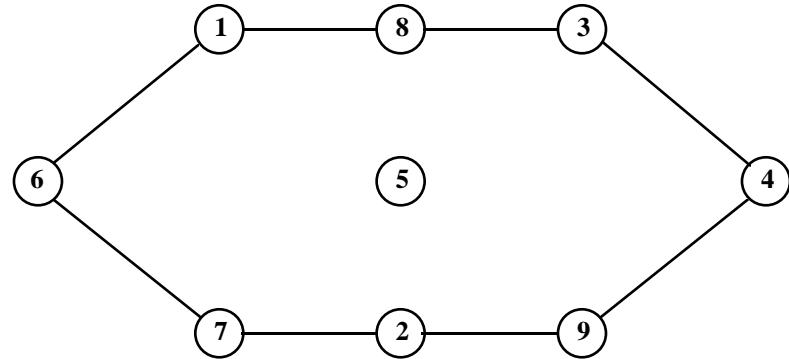


Figura 5.12: Scambio di cavalli

Assegnazione delle aule. In una facoltà universitaria vi sono nello stesso orario n corsi e m aule disponibili. Per ragioni di capienza delle aule e di attrezzature disponibili, ogni corso pu essere tenuto solo in alcune delle aule. Il problema è quello di stabilire quale è il numero massimo di corsi che è possibile tenere nell'orario considerato.

Il problema può essere scritto come problema su grafi nel seguente modo: sia $G(V_1, V_2, E)$ un grafo bipartito in cui l'insieme di nodi V_1 corrisponde ai corsi, l'insieme di nodi V_2 corrisponde alle aule ed esiste un arco fra due nodi i e j (con i appartenente ad V_1 e j appartenente a V_2) se (e solo se) il corso i può essere tenuto nell'aula j . Il problema di stabilire qual è il numero massimo di corsi che è possibile tenere nell'orario considerato può ora essere formulato come il problema di scegliere sul grafo il massimo numero di archi (corrispondenti ad assegnazioni di corsi ad aule) tali che due archi scelti non siano mai adiacenti. Infatti se due archi fossero adiacenti in un nodo dell'insieme V_1 , allora vorrebbe dire che uno stesso corso si tiene contemporaneamente in due aule diverse. Se due archi fossero adiacenti in un nodo dell'insieme V_2 , allora vorrebbe dire che in una stessa aula si tengono contemporaneamente due corsi. In entrambi i casi le assegnazioni di aule non sono ovviamente ammissibili. Questo problema è detto di “accoppiamento bipartito” e verrà risolto nel capitolo 8.

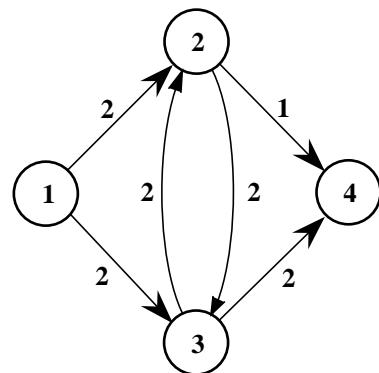
Rete di trasporto con ritardo Sia data una rete di trasporto con 4 nodi e 5 archi, pesata sugli archi⁵ con pesi corrispondenti alla quantità trasportabile sull'arco in un singolo viaggio (per esempio 5 tonnellate). Su ogni arco un viaggio richiede una unità di tempo (per esempio un'ora) e in ogni unità di tempo può essere effettuato al più un viaggio su ogni arco. In ogni nodo può essere inoltre immagazzinata merce senza limiti di tempo e di quantità.

La rete è riportata come grafo $G = (V, E)$ nella figura 5.13. Il problema è quello di stabilire se è possibile trasferire una assegnata quantità di merce dal nodo 1 della rete al nodo 4 entro 3 unità di tempo. Per formulare il problema in modo da individuare più facilmente la soluzione è opportuno introdurre un altro grafo (orientato) $H = (V', E')$ (vedi figura 5.13). Nel grafo H , ogni nodo corrisponde ad un nodo del grafo G ad un dato istante di tempo t . Il tempo è in questo caso assunto discreto, per cui t può essere uguale a 0, 1, 2 oppure 3. Ogni nodo di H può quindi essere indicato con due indici: uno relativo al nodo di G considerato, l'altro relativo al tempo. Nel grafo H , esiste un arco tra due nodi (per esempio (i, j) e (k, h)) se è possibile inviare in una unità di tempo del materiale da un nodo all'altro. Quindi, nel nostro esempio, perché esista un arco, dovrà essere $h = j + 1$ e (i, k) dovrà appartenere all'insieme di archi A . La capacità di ogni arco di H sarà pari alla capacità del corrispondente arco di G . Per esempio la capacità dell'arco da (i, j) a (k, h) di H sarà pari alla capacità dell'arco (i, k) di G . Inoltre, poiche si suppone che la merce possa restare quanto si vuole in ogni nodo senza limiti di capienza, è necessario introdurre alcuni archi aggiuntivi, di capacità infinita, tra tutte le coppie di nodi (i, j) e (k, h) tali che $i = k$ e $h = j + 1$. Il problema, formulato sul nuovo grafo H , è ora quello di decidere se è possibile inviare un flusso pari alla quantità di merce da trasportare dal nodo $(1, 0)$ al nodo $(4, 3)$. Questo problema può essere facilmente risolto con l'algoritmo del massimo flusso che verrà trattato nel capitolo 8.

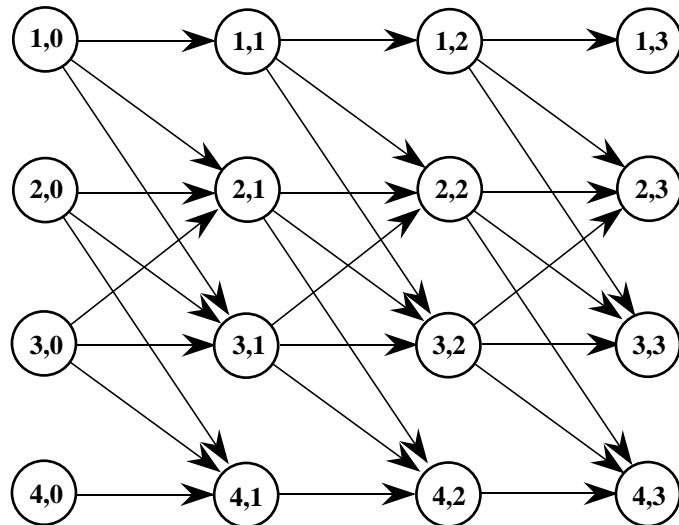
Colorazione di una carta geografica. Il problema è quello di stabilire il numero minimo di colori necessario per colorare una carta geografica, assegnando colori alle varie nazioni rappresentate sulla carta, in modo che due nazioni adiacenti (che quindi abbiano un tratto di frontiera in comune) siano sempre colorate in modo diverso. Si suppone che ogni nazione sia formata da una unica regione connessa e che nazioni adiacenti abbiano una linea di confine in comune di lunghezza maggiore di zero (e non quindi solo un singolo punto in comune). Assegnando ad ogni nazione un nodo di un grafo e collegando due nodi se e solo se le due nazioni corrispondenti sono adiacenti, si ottiene un grafo (che viene detto planare).

La famosa “congettura dei quattro colori”, ormai da chiamare “teorema dei quattro colori”, afferma che ogni carta geografica è colorabile utilizzando al più quattro colori. Questa congettura era già nota nel 1840. Nel corso di oltre un secolo sono state fornite dimostrazioni per molti casi particolari del problema

⁵questo vuol dire che ad ogni arco è associato un “peso”, cioè un numero



Grafo G



Grafo H

Figura 5.13: Rete di trasporto con ritardo

(oltre a molte dimostrazioni della congettura, poi rivelatesi sbagliate o valide per particolari classi di grafi). Solo nel 1976 la congettura è stata dimostrata in tutta la sua generalità usando gli strumenti della teoria dei grafi.

Capitolo 6

Cammini minimi

Come abbiamo accennato nel capitolo precedente, il linguaggio dei grafi permette di rappresentare in modo semplice la struttura di molti problemi applicativi, consentendo, in molti casi di grande importanza, di costruire metodi razionali di soluzione dei problemi stessi.

Fra i problemi più importanti, più semplici e più antichi, per la cui soluzione sono utilizzate rappresentazioni basate su grafi, vi sono i problemi di ricerca di cammini minimi, di cui ci occupiamo in questo capitolo.

I grafi considerati in questo capitolo sono, salvo diversa specificazione, grafi orientati. Nel caso della ricerca di cammini infatti è sempre possibile ricondursi con poca fatica a questo caso. Per fare questo è sufficiente sostituire ogni arco non orientato (e quindi percorribile in entrambi i versi) con due archi diretti in direzione opposta. Se esiste un cammino con le caratteristiche richieste nel grafo originario, allora esiste anche nel grafo trasformato e viceversa.

Per brevità, infine, trattando solo di grafi orientati, *indichiamo con grafo aciclico un grafo che non contenga cicli orientati*.

6.1 Il problema del cammino minimo e alcuni esempi di applicazioni

Dato un grafo orientato $G = (V, E)$, associamo a ciascun arco $e = (u, v) \in E$ un *peso* $p(u, v) \in \mathbb{R}$. Per ogni cammino **orientato** $P = \{v_1, e_1, \dots, e_{p-1}, v_p\}$, definiamo peso $p(P)$ del cammino P la somma dei pesi degli archi che appartengono a P , e cioè: $p(P) = \sum_{(u,v) \in P} p(u, v)$. Il problema del cammino minimo può essere enunciato nel modo seguente:

dati due nodi $s \in V$ e $t \in V$, trovare un cammino orientato
 P^* in G da s a t che abbia peso minimo.

Notiamo che:

- Se non esiste un cammino orientato che vada da s a t in G , il problema non ha soluzioni ammissibili.
- Se esiste un ciclo orientato C in G , tale che $p(C) < 0$ (peso negativo), il problema è illimitato inferiormente.

In genere, dato un nodo s gli algoritmi per il calcolo di cammini minimi determinano non il (o uno dei) cammini minimi da s a un fissato nodo t , ma il cosiddetto *albero dei cammini minimi*, cioè un sottografo di G che è un albero i cui nodi includono s e tutti i nodi da esso raggiungibili con un cammino orientato e tale che l'unico (vedi teorema 5.1.1) cammino da s a ogni altro nodo t dell'albero sia un cammino minimo da s a t . A prima vista questa può sembrare una complicazione non necessaria, ma in effetti, per come

sono organizzati gli algoritmi, vedremo che non è così. Quindi, in pratica, gli algoritmi che studieremo risolvono il seguente problema:

dato un nodo $s \in V$, trovare un albero dei cammini minimi
da s a ogni nodo in V raggiungibile da s .

Nella letteratura esistono moltissimi algoritmi per il calcolo dei cammini minimi. Ognuno di essi ha le sue peculiarità e le sue limitazioni. Per esempio alcuni algoritmi funzionano solo su determinate classi di grafi (grafi aciclici, grafi con pesi non negativi, etc.), inoltre ci possono essere notevoli differenze nella efficienza degli algoritmi. In generale un algoritmo che funziona su una classe ampia di grafi sarà più complesso di un algoritmo studiato per una classe ristretta di grafi, tenendo conto delle peculiarità di quella classe.

In questo capitolo considereremo due algoritmi: il primo calcola i cammini minimi su grafi aciclici, il secondo calcola i cammini minimi su grafi con pesi non negativi.

Nella parte rimanente di questa sezione illustriamo alcuni semplici esempi di calcolo dei cammini minimi. Alla fine del capitolo considereremo due casi più complessi e interessanti.

6.1.1 Percorso di tempo minimo su una rete stradale

Dato un grafo pesato che rappresenta la rete stradale italiana in cui i pesi degli archi indicano il (valore atteso del) tempo di percorrenza dell'arco, il problema è quello di trovare il cammino che congiunge due particolari nodi del grafo (nodo di partenza e nodo di arrivo) con tempo di percorrenza minimo. Si noti che possono esistere più cammini con la caratteristica di essere minimi.

Questo esempio, apparentemente semplice, solleva complessi problemi di modellistica. Un primo problema è relativo al livello di dettaglio necessario nella rappresentazione. Se, per esempio, il percorso da effettuare parte da Milano e arriva a Brindisi, il grafo dovrà contenere solo le autostrade e le principali strade di collegamento fra città diverse e un'intera città potrà essere rappresentata con un nodo del grafo. Un eccesso di dettaglio appesantirebbe inutilmente la rappresentazione, rendendo ogni algoritmo di soluzione lento e inefficiente. Se, d'altra parte, il percorso da effettuare parte da piazza San Pietro a Roma e arriva a un indirizzo di Frascati (a circa 20 km. da Roma), la rappresentazione dovrà essere completamente diversa, non solo perché il grafo sarà diverso, ma anche perché il peso degli archi, per essere significativo, dovrà prendere in considerazione i problemi del traffico urbano e quindi essere funzione dell'ora ed eventualmente del giorno. Infatti, effettuare tale percorso alle 10 di mattina di un giorno feriale non sarà ovviamente la stessa cosa che effettuarlo alle 4 del mattino (magari di un giorno festivo).

Un secondo problema è relativo al tipo di obiettivi che ci si propone. Infatti, una rappresentazione può essere o può non essere adeguata, a seconda del motivo per cui si vuole conoscere il cammino di tempo minimo. Per esempio, per alcune applicazioni critiche (ambulanze, vigili del fuoco, polizia), non basta l'informazione sul valore atteso del tempo di transito, ma serve anche valutare la varianza di tale tempo, ossia le possibili variazioni rispetto alla media. È meglio infatti utilizzare un percorso un po' più lungo ma con un tempo di percorrenza prevedibile con relativa certezza, piuttosto che un percorso mediamente più breve ma per cui vi sia il rischio di restare imbottigliati nel traffico.

Un terzo problema è relativo alla quantità di informazioni che è necessario inserire nell'elaboratore per affrontare il problema. Per calcolare il cammino da Milano a Brindisi è necessario inserire l'intera carta stradale italiana, o basta una porzione? Da un lato, più informazioni vengono inserite maggiore è il tempo necessario per inserirle e il costo dell'operazione; d'altro canto se vengono calcolati spesso percorsi di tempo minimo sulla rete stradale italiana forse conviene memorizzare tutto in modo organico una volta per tutte.

6.1.2 Costruzione di una autostrada

Il problema considerato è quello di costruire al costo minimo una autostrada fra le città A e B. Nel grafo riportato in figura 6.1 i nodi rappresentano i punti per cui l'autostrada può passare e gli archi i

possibili collegamenti fra punti. Il peso degli archi rappresenta il costo di costruzione della relativa tratta di autostrada.

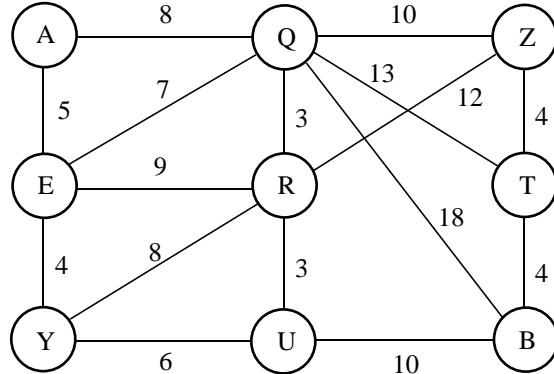


Figura 6.1: Cammino minimo tra A e B

Si noti che in questo caso il grafo è non orientato in quanto ogni arco può essere percorso in entrambi i sensi; per ricondursi al caso orientato basta sostituire ogni arco con due archi orientati in senso opposto, a ognuno dei quali viene attribuito un peso pari al peso dell'arco eliminato.

La scelta di una autostrada tra A e B con costo complessivo di costruzione minimo, corrisponde alla scelta del cammino di minimo costo dal nodo A al nodo B sul grafo. Nella figura 6.1 tale cammino è indicato in grassetto e il costo complessivo è pari a 24.

6.2 Cammini minimi e massimi su grafi aciclici

Una classe di grafi orientati di particolare interesse in campo applicativo (vedi gli esempi alla fine del capitolo) è la classe di grafi aciclici. La ricerca di cammini minimi o massimi su tali grafi è uno strumento di progetto di notevole importanza. Come si vedrà nel seguito, i due problemi di minimo e di massimo sono, in questo caso particolare, risolti da due algoritmi identici eccetto che per la sostituzione di un massimo a un minimo nella formula ricorsiva alla base del procedimento. Nel seguito verrà prima considerato il caso di un problema di cammino minimo. Per poter procedere con la descrizione dell'algoritmo, è necessario studiare prima una particolare tecnica di numerazione dei nodi di un grafo aciclico.

6.2.1 Numerazione topologica dei nodi di un grafo

Una caratteristica peculiare dei grafi aciclici (sia $G = (V, E)$ il grafo, con $|V| = n$ e $|E| = m$) consiste nella possibilità di numerare i nodi del grafo con i numeri $1, 2, 3, \dots, n-1, n$ in modo tale che:

se esiste un arco dal nodo i al nodo j allora $j > i$

Tale numerazione viene detta numerazione topologica dei nodi del grafo e non è in generale unica.

Non tutti i grafi possono essere numerati topologicamente. In effetti l'esistenza di una numerazione topologica dei nodi di un grafo caratterizza esattamente la classe dei grafi aciclici. Vale infatti il seguente teorema.

Teorema 6.2.1 *Un grafo è aciclico se e solo se esiste una numerazione topologica dei suoi nodi.*

Dimostrazione.

Sufficienza. Supponiamo che esista una numerazione topologica dei nodi e facciamo vedere che l'esistenza di un ciclo porterebbe ad una contraddizione. Possiamo assumere che i nodi siano numerati topologicamente, indicheremo l' i -esimo nodo di questa particolare numerazione, come v_i . Se esiste un ciclo (orientato) vuol dire che esiste una successione di nodi $(v_i, v_j, v_k, \dots, v_r, v_s)$ tali che

- esistono gli archi $(v_i, v_j), (v_j, v_k), \dots, (v_r, v_s)$;
- $v_i = v_s$.

Ma allora, da una parte, siccome la numerazione è topologica abbiamo $i < j < k < \dots < r < s$, cioè $i < s$, mentre dall'altra, poiché $v_i = v_s$ abbiamo $i = s$. Questa è una contraddizione e così il grafo deve essere aciclico.

Necessità. Supponiamo che il grafo sia aciclico e mostriamo che deve esistere almeno una numerazione topologica. La dimostrazione è costruttiva, faremo cioè vedere che esiste una numerazione topologica costruendone una.

Come primo passo osserviamo che se il grafo è aciclico, deve esistere almeno un nodo che non abbia archi entranti. Infatti, se ciò non fosse vero, potremmo ragionare nel modo seguente. Prendiamo un nodo qualunque, chiamiamolo v_1 . Siccome tutti i nodi hanno archi entranti esiste un nodo predecessore di v_1 , indichiamolo con v_2 ; notiamo che per come abbiamo scelto v_2 esiste l'arco (v_2, v_1) (attenzione, questa è una numerazione non topologica). Possiamo ripetere il ragionamento con v_2 e trovare un nodo v_3 tale che esista l'arco (v_3, v_2) . Siccome stiamo supponendo, per assurdo, che tutti gli archi abbiano degli archi entranti, possiamo ripetere il ragionamento quante volte vogliamo. Notiamo che ogni nodo generato deve essere diverso dai precedenti, altrimenti avremmo trovato un ciclo, contraddicendo l'aciclicità del grafo. D'altra parte, arrivati a v_n i nodi del grafo sono "finiti" e quindi il predecessore di v_n che stiamo supponendo esistente per assurdo, deve per forza essere uno dei nodi già esaminati. Così si viene a formare un ciclo.

Quindi dato un grafo aciclico deve per forza esistere almeno un nodo che non ha archi entranti. Prendiamo uno di questi nodi e numeriamolo con il numero 1. Eliminiamo dal grafo il nodo 1 e tutti gli archi uscenti da esso. Il nuovo grafo che otteniamo è ovviamente ancora un grafo aciclico. Quindi per lo stesso ragionamento fatto prima deve esistere almeno un nodo che non ha archi entranti. Prendiamo uno di questi nodi e numeriamolo con il numero 2. Notiamo che ovviamente se consideriamo il grafo originario il nodo 2 può avere archi entranti, ma solo provenienti dal nodo 1 e quindi la condizione $i < j$ è rispettata. Possiamo ora ripetere il procedimento n volte (quanti sono i nodi) ed ottenere così una numerazione topologica del grafo. \square

Il precedente teorema è importante, anche perché nella dimostrazione della necessità è sostanzialmente dato un algoritmo per numerare topologicamente i nodi di un grafo.

Riesponiamo qui l'algoritmo per chiarezza.

- Siccome il grafo è aciclico, deve esistere almeno un nodo con solo archi uscenti;
- individuiamo uno di questi nodi e attribuiamogli il numero 1;
- cancelliamo il nodo numerato e tutti gli archi adiacenti, nel nuovo grafo ridotto individuiamo un nodo con soli archi uscenti e attribuiamogli il numero 2, e così via fino ad aver numerato tutti i nodi

La correttezza e validità di questa procedura è stata provata nella dimostrazione della necessità del Teorema 6.2.1.

Notiamo che se ad un certo punto dell'applicazione della procedura non possiamo procedere, se succede cioè che ad un determinato passo non riusciamo a trovare un nodo senza archi entranti, questo vuol dire che il grafo considerato contiene un ciclo.

Quindi la procedura per la numerazione topologica di un grafo può anche essere utilizzata per determinare se un grafo è aciclico o meno.

Come esempio consideriamo il grafo di figura 6.2.

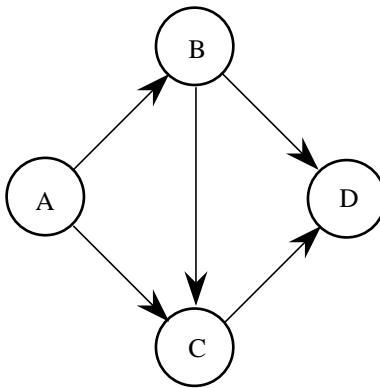


Figura 6.2: Numerazione dei nodi di un grafo

Si tratta di decidere se il grafo è aciclico e, in caso, numerare i nodi topologicamente. I vari passi dell'algoritmo sono riportati qui di seguito

Passo	Nodo senza archi entranti	Nodi non ancora numerati	Numerazione
1	A	B,C,D	A=1
2	B	C,D	B=2
3	C	D	C=3
4	D	\emptyset	D=4

Il procedimento è terminato con la numerazione di tutti i nodi. Il grafo è dunque aciclico e la numerazione trovata è topologica.

Supponiamo ora che nel precedente grafo l'arco (B, D) sia orientato da D a B . Ovviamente si verrebbe a creare un ciclo. Se proviamo ad applicare la procedura per la determinazione di una numerazione topologica possiamo iniziare numerando il nodo A come nodo 1. Ma dopo non possiamo più procedere perché eliminato il nodo A e gli archi da esso uscenti $((A,B) \text{ e } (A,C))$ non è più possibile individuare un nodo che non abbia archi entranti.

6.2.2 Un algoritmo per il cammino minimo su grafi aciclici

La numerazione dei nodi di un grafo aciclico descritta nella sezione precedente consente di costruire un algoritmo di soluzione per il problema di cammino minimo particolarmente semplice. Infatti, nella ricerca di un cammino tra una qualsiasi coppia di nodi i e j del grafo, a causa della numerazione attribuita ai nodi, si può affermare che:

se $j < i$ allora non esistono cammini da i a j ;

se $j > i$ allora gli unici nodi che è necessario considerare nella ricerca del cammino da i a j sono i nodi con indice k tale che $i < k < j$.

Infatti, se il cammino passasse per un nodo $h > j$, allora non potrebbe tornare su j , a causa della mancanza di archi che collegano nodi con indice maggiore a nodi con indice minore; se passasse per un nodo $h < i$, allora dovrebbe esistere un cammino da h a i , il che comporterebbe l'esistenza di archi che collegano nodi con indice maggiore a nodi con indice minore.

Sulla base di queste considerazioni, è possibile impostare un algoritmo per il calcolo dell'albero dei cammini minimi tra un nodo del grafo (per esempio il nodo 1) e tutti i nodi con indice superiore (per quelli con indice inferiore non esiste sicuramente un cammino; ovviamente, se il nodo di partenza è quello contrassegnato con l'indice 1, allora si tratta di calcolare l'albero dei cammini minimi tra il nodo 1 e tutti gli altri). L'algoritmo per il calcolo dell'albero dei cammini minimi dal nodo 1 a tutti gli altri nodi si basa sul calcolo in sequenza dei cammini minimi dal nodo 1 al nodo 2, dal nodo 1 al nodo 3, dal nodo 1 al nodo 4, e così via. Indichiamo con:

- $p(i, j)$ il peso dell'arco (i, j) che parte dal nodo i e arriva al nodo j ;
- $f(i)$ il valore del cammino minimo dal nodo 1 al nodo i ;
- $J(i)$ il nodo che precede i su tale cammino (nel caso il cammino minimo non sia unico, allora se ne sceglie uno qualsiasi fra quelli minimi).

Si noti che dai valori $J(i)$ è possibile ricostruire in modo immediato l'albero (o uno dei possibili alberi) dei cammini minimi.

Possiamo allora illustrare l'algoritmo per il calcolo dei percorsi minimi.

- $f(1) := 0; J(1) := 1;$
- per $j = 2, 3, 4, \dots, n - 1, n$ ripeti la seguente serie di operazioni

$$f(j) := \min_{(i,j) \in \omega^-(j)} \{f(i) + p(i, j)\};$$

$$J(j) := \text{valore di } i \text{ per cui si è verificato il minimo};$$

Si noti che una volta assegnato un peso $f(i)$ a un nodo (cioè un valore del cammino minimo dal nodo 1 al nodo considerato), tale peso non viene più modificato nel corso dell'algoritmo, ma indica in modo definitivo il valore del cammino. Questo è dovuto al fatto, già citato, che tutti i nodi successivi non devono essere considerati per il calcolo del percorso dal nodo 1 al nodo i .

Come esempio consideriamo il grafo di figura 6.3.

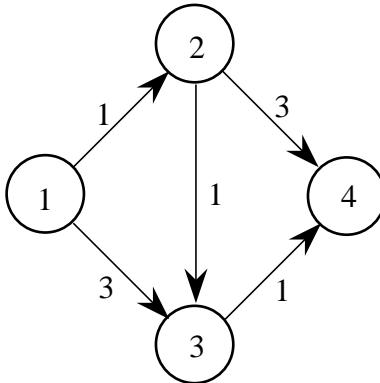


Figura 6.3: Percorso minimo su grafi aciclici

Si tratta di determinare l'albero dei cammini minimi tra il nodo 1 e tutti gli altri nodi. I vari passi dell'algoritmo (corrispondenti alla successione di nodi visitati, si osservi che in questo caso particolare l'indice del passo coincide con l'indice del nodo visitato) portano alla seguente successione di valori $f(i)$ e $J(i)$

Passo	Valore di $f(i)$	Valore di $J(i)$
1	$f(1) = 0$	$J(1) = 1$
2	$f(2) = 1$	$J(2) = 1$
3	$f(3) = \min\{3, 1 + 1\} = 2$	$J(3) = 2$
4	$f(4) = \min\{1 + 3, 2 + 1\} = 3$	$J(4) = 3$

Il corrispondente albero dei cammini minimi è riportato in figura 6.4

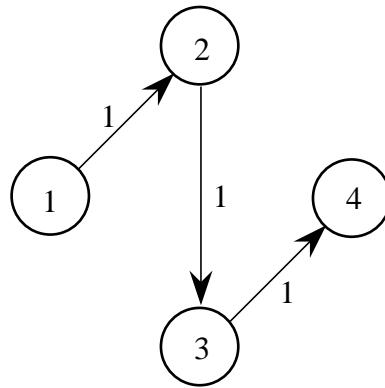


Figura 6.4: Albero dei cammini minimi

Esempio 6.2.2 Sia dato il grafo di Figura 6.5. Determinare l'albero dei cammini minimi, utilizzando

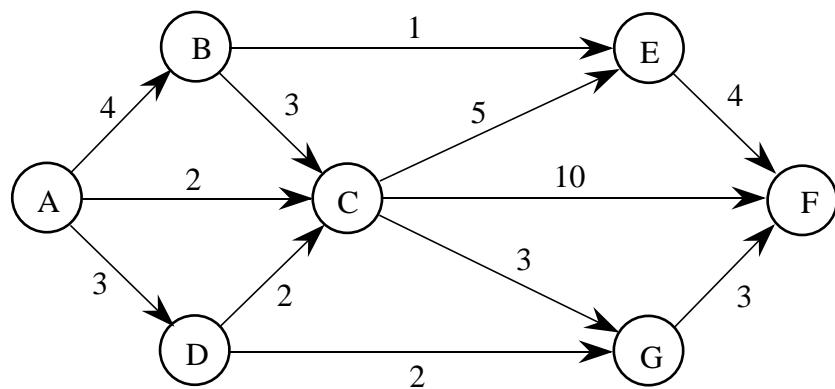


Figura 6.5: Grafo Esercizio 6.2.2

l'algoritmo per grafi aciclici.

Soluzione. Si deve prima numerare topologicamente il grafo. I passi sono riportati nella seguente tabella ed il grafo risultante in Figura 6.6

Passo	Nodo senza archi entranti	Nodi non ancora numerati	Numerazione
1	A	B,C,D,E,F,G	A=1
2	B o D	B,C,E,F,G	D=2
3	B	C,E,F,G	B=3
4	C	E,F,G	C=4
5	E o G	F,G	E=5
6	G	F	G=6
7	F	\emptyset	F=7

Osserviamo che la numerazione topologica in questo esempio non è unica; infatti ai passi 2 e 5 potevamo scegliere tra due nodi. Applichiamo ora l'algoritmo. I passi sono riportati nella seguente tabella

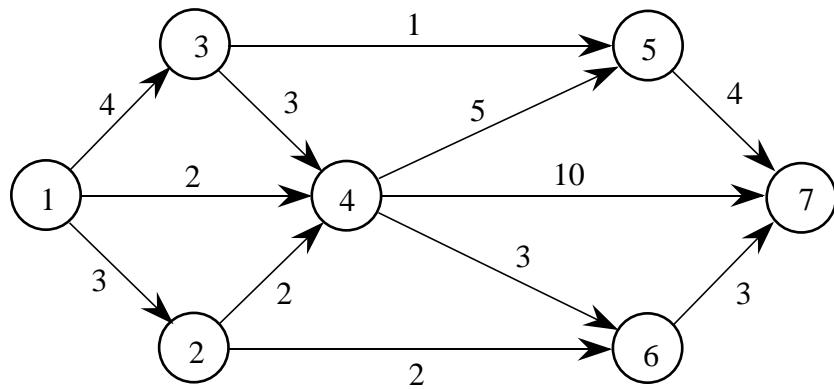


Figura 6.6: Numerazione topologica del grafo di Figura 6.5

Passo	Valore di $f(i)$	Valore di $J(i)$
1	$f(1) = 0$	$J(1) = 1$
2	$f(2) = 3$	$J(2) = 1$
3	$f(3) = 4$	$J(3) = 1$
4	$f(4) = \min\{0 + 2, 3 + 2, 4 + 3\} = 2$	$J(4) = 1$
5	$f(5) = \min\{4 + 1, 2 + 5\} = 5$	$J(5) = 3$
6	$f(6) = \min\{3 + 2, 2 + 3\} = 5$	$J(6) = 4$
7	$f(7) = \min\{2 + 10, 5 + 4, 5 + 3\} = 8$	$J(7) = 6$

Il corrispondente albero dei cammini minimi è riportato in figura 6.7.

6.2.3 Un algoritmo per il cammino massimo su grafi aciclici

Se il problema di ottimo è quello della determinazione del cammino di peso massimo sul grafo, allora è facile convincersi che basta sostituire nella formula ricorsiva al min un max e tutte le considerazioni fatta continuano, in questo caso particolare di grafi aciclici, a essere valide. Oltre che da considerazioni dirette questo risultato può essere dedotto considerando che il problema di trovare un cammino di peso massimo su un grafo aciclico è equivalente a quello di trovare un cammino di peso minimo sullo stesso grafo dove i pesi, però, sono stati cambiati di segno.

L'algoritmo per il calcolo dei cammini massimi su grafici aciclici è allora il seguente

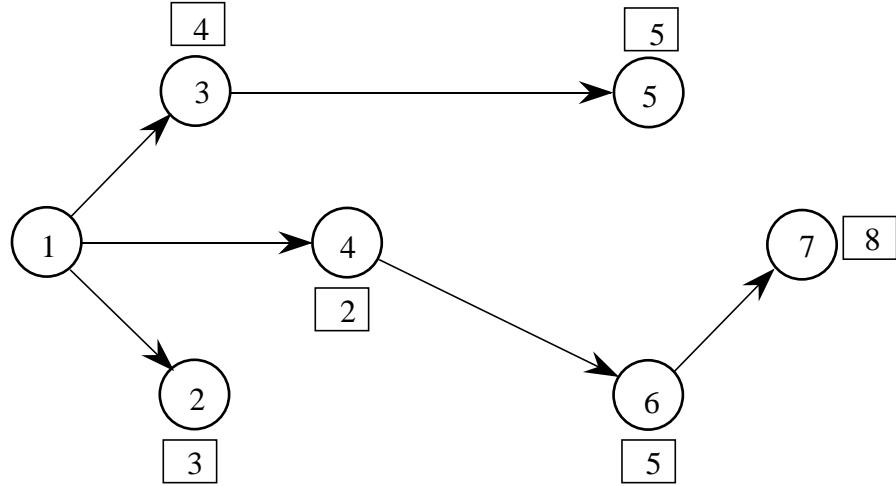


Figura 6.7: Albero dei cammini minimi

- $f(1) := 0; J(1) := 1;$
- per $j = 2, 3, 4, \dots, n - 1, n$ ripeti la seguente serie di operazioni
 - $f(j) := \max_{(i,j) \in \omega^-(j)} \{f(i) + p(i,j)\};$
 - $J(i) :=$ valore di i per cui si è verificato il massimo;

Come esempio consideriamo sempre il grafo di figura 6.3. Si tratta di determinare l'albero dei cammini minimi tra il nodo 1 e tutti gli altri nodi. I vari passi dell'algoritmo portano alla seguente successione di valori $f(i)$ e $J(i)$

Passo	Valore di $f(i)$	Valore di $J(i)$
1	$f(1) = 0$	$J(1) = 1$
2	$f(2) = 1$	$J(2) = 1$
3	$f(3) = \max\{3, 1 + 1\} = 3$	$J(3) = 1$
4	$f(4) = \max\{1 + 3, 3 + 1\} = 4$	$J(4) = 2$ (oppure $J(4) = 3$)

6.3 Cammini minimi su grafi con pesi positivi: algoritmo di Dijkstra

L'algoritmo di Dijkstra permette di risolvere il problema del cammino minimo fra due nodi qualora tutti i pesi degli archi siano **non negativi**. Più precisamente, l'algoritmo calcola il peso del cammino minimo da un nodo s a tutti gli altri nodi del grafo, costruendo contemporaneamente l'albero dei cammini minimi. Siccome in questo caso la numerazione non gioca nessun ruolo, in questo paragrafo supponiamo, senza perdita di generalità, che s sia sempre uguale a 1.

Notiamo la differenza con il caso esaminato nel paragrafo precedente: nel caso precedente non c'era nessuna restrizione sui pesi, ma c'era una restrizione sulla *topologia* del grafo, che non doveva contenere cicli orientati. Nel caso esaminato in questo paragrafo, non c'è nessuna restrizione sulla topologia del grafo (che può essere qualunque e contenere, dunque, cicli orientati), ma c'è una restrizione sui pesi, che devono essere non negativi.

È evidente che il caso analizzato in questo paragrafo è di interesse in quanto, per esempio, in tutti i casi in cui la ricerca di cammini minimi corrisponde alla ricerca di un reale percorso in, per esempio, una città, esisteranno cicli, ma i pesi, che corrispondono a distanze fisiche, sono ovviamente positivi.

Un'altra differenza che vogliamo segnalare subito è che in questo caso, non è possibile dare una semplice variante dell'algoritmo che calcoli i cammini di peso massimo in quanto, se facciamo diventare il problema di minimo un problema di massimo cambiando i segni dei pesi, il grafo che otteniamo ha i pesi tutti *non positivi*, e quindi l'algoritmo non è più utilizzabile.

L'algoritmo per il calcolo dei cammini minimi su grafi aciclici si basa fortemente sul fatto che i nodi del grafo siano numerati topologicamente. Tenendo conto del fatto che il cammino minimo tra il nodo i e il nodo j (con $j > i$), se esiste, può passare solo per i nodi k , con k compreso tra i e j , abbiamo sviluppato una semplice procedura iterativa. Nel caso di grafi con pesi non negativi vogliamo, in qualche modo, ancora cercare di sviluppare un algoritmo che abbia le stesse caratteristiche. Ovviamente, non disponendo più di una numerazione topologica dobbiamo ragionare in maniera diversa. Vediamo su un esempio come possiamo ragionare.

Consideriamo il grafo in figura 6.8 e proponiamoci di trovare i cammini minimi dal nodo 1 a tutti i nodi da esso raggiungibili. Analogamente al caso dei grafi aciclici, poniamo $f(1) = 0$ e $J(1) = 1$. Vogliamo anche in questo caso arrivare ad associare ad ogni nodo i del grafo due etichette, $f(i)$ e $J(i)$ che diano rispettivamente, la distanza dal nodo 1 e il predecessore di i su un cammino minimo che va da 1 a i .

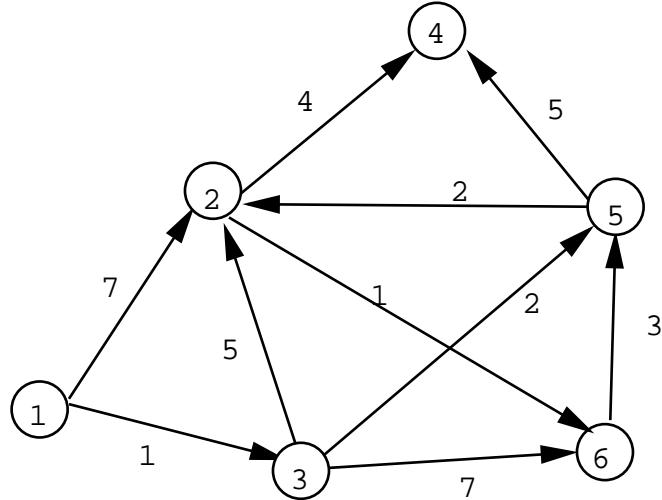


Figura 6.8: Grafo con pesi non negativi

Consideriamo ora i nodi raggiungibili da 1 con un solo arco: sono 2 e 3. Notiamo che il peso dell'arco $(1, 3)$, che è uguale a 1, è minore del peso dell'arco $(1, 2)$, che è uguale a 7. Possiamo allora porre $f(3) = 1$ e $J(3) = 1$. Infatti, supponiamo per assurdo che esista un cammino minimo per andare da 1 a 3 diverso da quello fornito dall'arco $(1, 3)$ e con un peso più piccolo. Questo cammino dovrebbe prima "uscire" da 1, passando quindi per un arco di peso 1 o uno di peso 7, quindi dovrebbe "tornare" a 3. Ma siccome gli archi hanno tutti pesi non negativi, il peso di questa seconda parte di cammino si va ad aggiungere a quello dell'arco usato per "uscire" da 1. E' quindi ovvio che, essendo il peso dell'arco $(1, 3)$ il più piccolo tra i pesi degli archi uscenti da 1, il peso di questo altro ipotetico cammino, deve essere almeno di 1. Quindi le etichette assegnate al nodo 3 sono corrette.

Possiamo ora ripetere questo ragionamento. Consideriamo i nodi raggiungibili da 1 e 3 (i nodi già

etichettati). Sono 2, 5, 6. Le distanze minime da 1, con la restrizione di passare solo per 1 e 3, sono:

- per il nodo 2: 6 (distanza data dal cammino che passa per i nodi 1, 3, 2);
- per il nodo 5: 3 (distanza data dal cammino che passa per i nodi 1, 3, 5);
- per il nodo 6: 8 (distanza data dal cammino che passa per i nodi 1,3,6).

Notiamo che per andare al 2 due passando solo per i nodi 1 e 3 (già etichettati) esiste anche dato dall'arco (1,2), ma questi ha un peso 7 superiore a quello che si ottiene passando per 3. Osserviamo anche che il nodo scelto è, *tra i nodi ancora non etichettati, quello che ha la distanza minima da 1 se ci limitiamo a considerare solo i cammini che passano per i nodi già etichettati*. A questo punto possiamo ragionare in modo simile a quello adottato prima, anche se la situazione è leggermente più complessa. Consideriamo il nodo 5 (quello che ha la distanza più piccola, tra quelli raggiungibili dai nodi 1 e 3), e poniamo $f(5) = 3$ e $J(5) = 3$. Queste sono etichette corrette. Supponiamo infatti che esista un altro cammino, C, da 1 a 5, diverso da quello trovato (e dato dai nodi 1,3 e 5) e con un peso minore di 3. Questo cammino C deve passare per almeno un nodo diverso da 1 e 3 e 5 per l'osservazione in corsivo fatta poche righe sopra. Ora, sia j il primo nodo diverso da 1, 3 e 5 nel cammino C. Il peso del cammino da 1 a j deve essere di almeno 3, sempre per l'osservazione in corsivo fatta prima. Siccome la parte del cammino che va da j a 5 ha un valore non negativo (per l'ipotesi che i pesi siano tutti non negativi) abbiamo di nuovo un assurdo.

A questo punto la tecnica da adottare dovrebbe essere chiara. A un generico passo dell'algoritmo possiamo supporre di avere un insieme di nodi, diciamo S , che hanno già le loro etichette f e J correttamente assegnate. Si sceglie il nuovo nodo da mettere in S come il nodo (o uno dei nodi) che ha la distanza minima da 1 con il vincolo di passare solo per nodi di S . Ad ogni passo aggiungiamo un nodo in S e quindi in un numero di passi uguale al numero dei nodi, l'algoritmo termina con le etichette correttamente assegnate.

Qui di seguito diamo una descrizione più dettagliata (e più vicina a una possibile implementazione sul calcolatore) dell'algoritmo delineato. In questa descrizione indichiamo con S i nodi a cui sono state assegnate le etichette corrette f e J , e con T tutti gli altri nodi. A differenza di quanto visto finora, però, noi diamo delle *etichette provvisorie*, che denominiamo sempre, per brevità, f e J , anche ai nodi in T . Se i appartiene a T , $f(i)$ rappresenta le distanze minima del nodo i dal nodo 1, con il vincolo di passare solo per nodi di S , e $J(i)$ è il corrispondente predecessore lungo il cammino minimo. Se non esiste nessun cammino da 1 a i che passa solo per nodi in S , poniamo $f(i) = +\infty$ e $J(i)$ è lasciato indefinito. Queste etichette hanno l'unico scopo di facilitare, ad ogni passo, la scelta del nodo in T da mettere in S . Infatti è chiaro che basterà scegliere, di volta in volta, il nodo in T con il valore di f più piccolo. Ovviamente, ad ogni iterazione, cambiando i nodi in S , le etichette provvisori dovranno essere aggiornate. Questo viene fatto nel punto (c) dell'algoritmo, la cui logica dovrebbe essere chiara, e che verrà ulteriormente chiarito dall'esempio che faremo subito dopo la descrizione dell'algoritmo. L'algoritmo termina quando o tutti i nodi del grafo sono in S o quando tutti i nodi in T hanno il valore di f uguale a $+\infty$, fatto che ovviamente indica che i nodi in T non sono raggiungibili da 1.

Agoritmo di Dijkstra

(a) Inizializzazione.

Poni $S \leftarrow \{1\}$, $T \leftarrow \{2, \dots, n\}$. $f(1) = 0$, $J(1) = 1$.

Poni $f(i) = p(1, i)$, $J(i) = 1$, per $(1, i) \in \omega^+(1)$.

Poni $f(i) = +\infty$, per $(1, i) \notin \omega^+(1)$.

(b) Assegnazione etichetta permanente

Trova $j \in T$ tale che $f(j) = \min_{i \in T} f(i)$.

Poni $T = T - \{j\}$, $S = S \cup \{j\}$.

Se $T = \emptyset$ o $f(i) = +\infty, \forall i \in T$ **STOP** (terminazione dell'algoritmo).

(c) *Assegnazione etichetta provvisoria*

Per ogni $(j, i) \in T \cap \omega^+(j)$ tale che $f(i) > f(j) + p(j, i)$ *Poni*:

$$\text{c.1 } f(i) = f(j) + p(j, i)$$

$$\text{c.2 } J(i) = j$$

Vai al passo (b).

Vediamo, nel caso del grafo di figura 6.8, come l'algoritmo proceda (per semplicità, l'evoluzione di T è omessa essendo $T = V - S$).

Iterazione 0 Inizializzazione.

- (a) $S = \{1\}$. $d(1) = 0$. $d(2) = 7$. $d(3) = 1$. $d(4) = +\infty$. $d(5) = +\infty$. $d(6) = +\infty$.
 $J(2) = 1$. $J(3) = 1$. $J(4) = 1$. $J(5) = 1$. $J(6) = 1$.

Iterazione 1.

- (b) $j = 3$. $S = \{1, 3\}$.

(c.1) $(\omega^+(j) \cap T = \{(3, 2), (3, 5), (3, 6)\}$. E' facile vedere che per ognuno dei nodi $\{2, 5, 6\}$ è verificata la condizione $d(i) > d(j) + p(j, i)$, e quindi le etichette vanno tutte aggiornate. $d(2) = d(3) + 1 = 6$. $d(5) = 2 + d(3) = 3$. $d(6) = 7 + d(3) = 8$.

- (c.2) $J(2) = 3$. $J(5) = 3$. $J(6) = 3$.

Iterazione 2.

- (b) $j = 5$. $S = \{1, 3, 5\}$.

(c.1) $\omega^+(j) \cap T = \{(5, 2), (5, 4)\}$. $d(2) = d(5) + 2 = 5$. $d(4) = d(5) + 5 = 8$.

- (c.2) $J(2) = 5$. $J(4) = 5$.

Iterazione 3

- (b) $j = 2$. $S = \{1, 2, 3, 5\}$.

(c.1) $\omega^+(j) \cap T = \{(2, 4), (2, 6)\}$. L'etichetta del nodo 4 non soddisfa la condizione al passo (c), per cui va' aggiornata solo l'etichetta (e il predecessore) del nodo 6. $d(6) = d(2) + 1 = 6$.

- (c.2) $J(6) = 2$.

Iterazione 4.

- (b) $j = 6$. $S = \{1, 2, 3, 5, 6\}$.

(c) $\omega^+(j) \cap T = \emptyset$.

Iterazione 5.

- (b) $j = 4$. $S = \{1, 2, 3, 4, 5, 6\}$. **STOP**.

I pesi dei cammini minimi saranno quindi: $d(1) = 0$. $d(2) = 5$. $d(3) = 1$. $d(4) = 8$. $d(5) = 3$. $d(6) = 6$.

Una comoda rappresentazione dell'evolvere dell'algoritmo è la seguente forma tabellare ove le righe rappresentano iterazioni mentre le colonne rappresentano i nodi selezionati ad ogni iterazione. Per ciascun nodo j ci sono due colonne che riportano il valore della variabile $d(j)$ e della $J(j)$ all'iterazione i -esima. L'elemento selezionato all'iterazione i -esima è rappresentato in grassetto e, nelle iterazioni successive, il valore della variabile corrispondente non viene più riportato. La colonna corrispondente al nodo 1 è omessa.

	nodo									
	2		3		4		5		6	
	d	J	d	J	d	J	d	J	d	J
It. 0	7	1	1	1	$+\infty$	1(fitt.)	$+\infty$	1(fitt.)	$+\infty$	1(fitt.)
It. 1	6	3	1	1	$+\infty$	1(fitt.)	3	3	8	3
It. 2	5	5			8	5	3	3	8	3
It. 3	5	5			8	5			6	2
It. 4					8	5			6	2
It. 5					8	5				

6.4 Due esempi

In questo paragrafo consideriamo più in dettaglio due esempi di applicazioni non banali di quanto visto in questo capitolo.

6.4.1 Tecniche reticolari di programmazione delle attività

I progetti di grandi dimensioni sono costituiti da più attività, che devono essere tutte completate affinché il progetto di cui fanno parte sia completato, ma che possono essere iniziate e svolte indipendentemente l'una dall'altra, purché sia rispettata una data sequenza. Queste condizioni sono caratteristiche di molti progetti di sviluppo e produzione, ad esempio nel settore aeronautico ed aeroespaziale, o di costruzione, ad esempio nell'ingegneria civile, o di manutenzione di grossi sistemi; tutti progetti in cui il numero di attività costituenti può essere dell'ordine delle migliaia. La gestione di un progetto consiste nel coordinamento dell'esecuzione delle varie attività, unitamente al controllo dei tempi e dei costi di esecuzione. Poiché questo è evidentemente un problema di rilevante importanza economica, e a volte strategica, per esso sono state sviluppate, a partire dal 1958, alcune tecniche particolarmente efficaci, tra cui hanno assunto un ruolo importante il PERT (Program Evaluation and Review Technique) e il CPM (Critical Path Method). Il PERT è stato sviluppato inizialmente per pianificare le operazioni di ricerca e sviluppo connesse al progetto del missile Polaris, e l'applicazione di questa tecnica ha consentito di concludere il progetto con due anni di anticipo sui cinque anni inizialmente preventivati. Questo successo iniziale ha portato ad una sua rapida diffusione. Lo scopo principale del PERT è quello di pianificare e controllare i tempi di completamento delle attività di un progetto, e del progetto nel suo insieme, tenendo conto del fatto che i tempi di esecuzione delle varie attività non sono a priori noti con certezza, e possono variare in dipendenza di molteplici fattori aleatori (PERT-Time); successivamente sono state fatte estensioni alla pianificazione e controllo dei costi di esecuzione, soggetti anch'essi ad aleatorietà (PERT-Cost). PERT-Time e PERT-Cost costituiscono applicazioni software molto diffuse. Il CPM si applica invece quando al tempo di esecuzione di ogni attività può essere attribuito un valore certo, più o meno lungo a seconda di quanto si decide di spendere per l'esecuzione dell'attività stessa; è esperienza comune che se si riduce il tempo di esecuzione di un'attività il suo costo aumenta e viceversa. Il CPM ha come scopo principale quello di pianificare e controllare i tempi di esecuzione di un progetto, rendendo minima la spesa complessiva, e trova larga applicazione in programmi riguardanti la manutenzione periodica di grossi impianti industriali, e lavori di produzione e costruzione per cui esiste una consolidata esperienza, cosicché

si possono ritenere note con esattezza le relazioni costo-tempo di esecuzione. Base comune del PERT e del CPM è la rappresentazione del progetto mediante un grafo orientato, secondo opportune norme. Poiché in questo contesto, al grafo rappresentativo del progetto viene dato il nome di diagramma reticolare , queste tecniche vengono chiamate tecniche reticolari di programmazione delle attività . Dall’analisi del diagramma reticolare si possono ricavare molte informazioni significative sul progetto. Vogliamo qui illustrare alcune considerazioni, che possono essere fatte nell’analizzare i progetti, in cui giocano un ruolo importanti i cammini minimi.

Iniziamo la nostra analisi spiegando come sia possibile associare ad un progetto un *diagramma reticolare* (cioè un grafo orientato).

La costruzione del diagramma

Il diagramma reticolare rappresenta la successione temporale e la reciproca dipendenza delle varie attività che concorrono all’esecuzione del progetto, attività che devono essere completate prima che il progetto possa considerarsi eseguito. Il primo passo nella costruzione del diagramma reticolare consiste nell’individuazione e nell’elencazione di tutte le attività coinvolte nell’esecuzione del progetto, con un livello di disaggregazione tale per cui le si possa considerare ciascuna distinta da tutte le altre. Segue una fase di rappresentazione grafica, che dà luogo al disegno di un grafo orientato in cui ogni attività è rappresentata da un arco o ramo i cui nodi estremi rappresentano, secondo la direzione del ramo, l’inizio e il termine dell’attività in questione. Pertanto nei diagrammi reticolari un’attività A è rappresentata come in figura 6.9, ove i nodi *i* e *j* rappresentano rispettivamente l’inizio e il termine dell’attività.

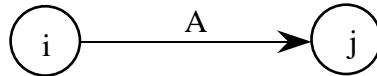


Figura 6.9: Rappresentazione grafica dell’ attività A

Naturalmente tra le varie attività esistono delle precedenze, per cui, per ciascuna attività, esisteranno altre attività che devono essere completate prima che quella in questione possa avere inizio. Il caso più semplice di precedenza quello indicato in figura 6.10a, ove l’attività A precede l’attività B, e il nodo *j* rappresenta il termine dell’attività A e l’inizio dell’attività B. In figura 6.10b è rappresentato il caso in cui l’attività A precede l’attività B che a sua volta precede l’attività C. Può però anche avvenire che due attività, la A e la B precedano una terza, la C, senza che tra A e B esista una relazione di precedenza: questo caso, in cui le attività A e B possono essere svolte in parallelo, è rappresentato in figura 6.10c. In figura 6.10d abbiamo il caso in cui le due attività B e C, tra cui non sussistono precedenze, sono entrambe precedute dall’attività A. Per esprimere il fatto che l’attività A precede l’attività B utilizziamo la notazione $A < B$, per esprimere il fatto che l’attività B è preceduta dall’attività A, utilizziamo la notazione $B > A$.

Esempio 1. In figura 6.11b) è rappresentato il caso di un progetto il cui completamento richiede l’esecuzione di 9 attività, tra cui sussistono le relazioni di precedenza:

$$A < B, C; \quad B < D, E; \quad C < F; \quad D < G; \quad E, F < H; \quad G, H < I.$$

Il progetto rappresentato in 6.11b verrà più volte riutilizzato a scopo esemplificativo; ad esso faremo pertanto riferimento con il nome di progetto P1.

Nel disegnare il diagramma reticolare si utilizzano le seguenti regole fondamentali, alcune delle quali già implicitamente enunciate:

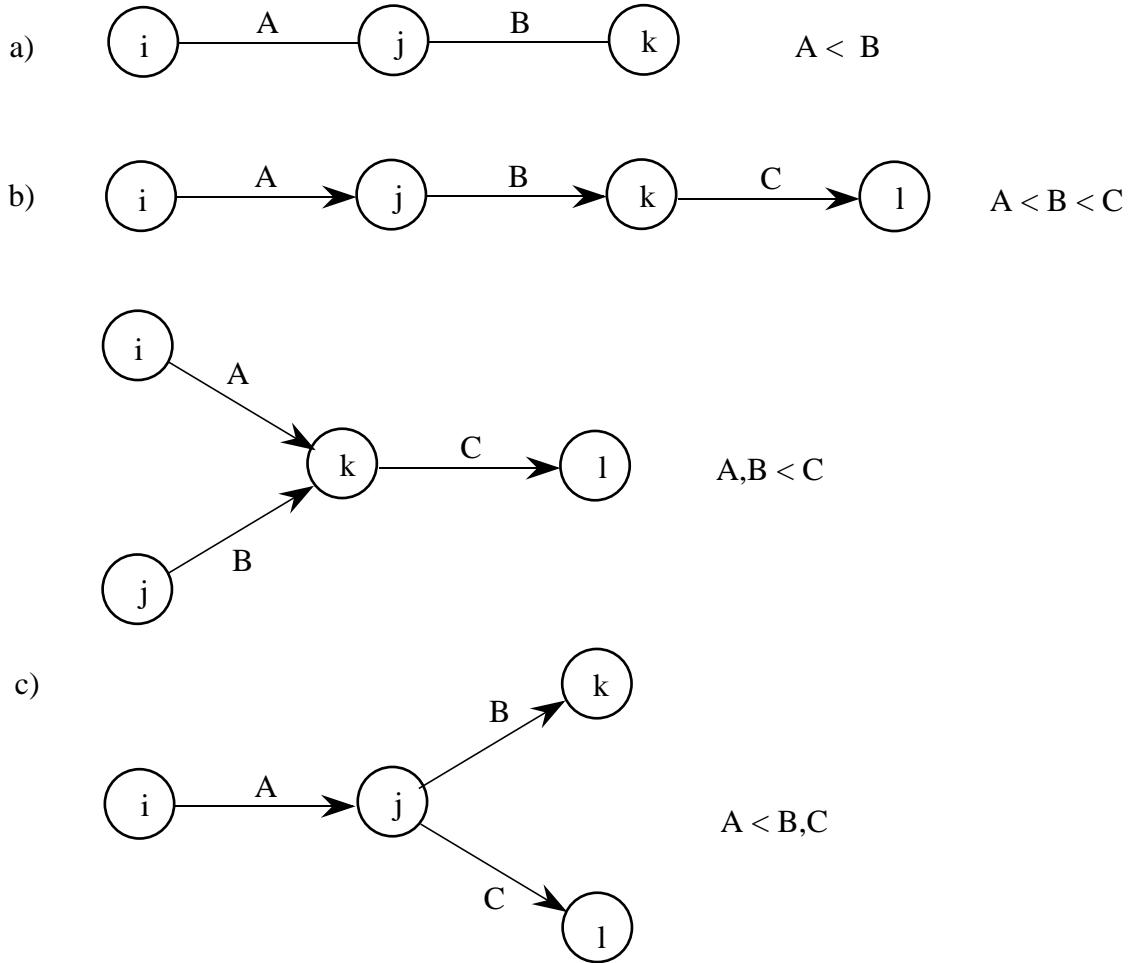


Figura 6.10: Rappresentazione grafica delle regole di precedenza

1. Le attività sono rappresentate dai rami del grafo.
2. L'inizio di un'attività è subordinato al completamento di tutte quelle che la precedono: in termini di diagramma reticolare ciò significa che rami diretti verso un nodo rappresentano attività da completare prima che abbiano inizio le attività rappresentate da rami aventi origine nel nodo stesso.
3. La lunghezza dei rami o la loro forma non hanno significato.
4. Due nodi non possono essere collegati da più di un ramo.
5. L'inizio del progetto è rappresentato da un nodo contrassegnato con zero.
6. Tutti i nodi sono numerati in modo che, se esiste un ramo diretto dal nodo i al nodo j , risulta $i < j$.
7. Il grafo può avere un solo nodo iniziale e un solo nodo finale.

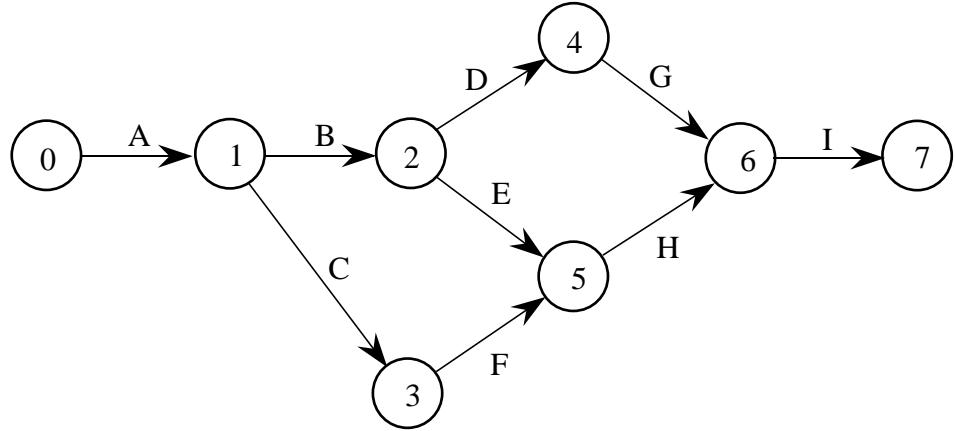


Figura 6.11: Diagramma reticolare del progetto P1

Delle suddette regole, le prime tre tengono conto della logica interna del grafo; le altre quattro sono richieste dai metodi di analisi del grafo, metodi che utilizzano tabulazioni ed uso di calcolatori.

Per quanto riguarda la regola 6, ricordiamo che una siffatta numerazione è detta numerazione topologica e che nel paragrafo precedente abbiamo visto che è possibile dare una numerazione topologica ai nodi di un grafo orientato se e solo se il grafo è aciclico. Bisogna quindi chiedersi se il diagramma reticolare di un progetto è un grafo aciclico. La risposta è ovviamente positiva. Se infatti esistesse un ciclo (orientato) nel diagramma reticolare di un progetto, questo vorrebbe dire, per come abbiamo costruito il diagramma reticolare stesso, che esistono delle attività che non possono iniziare prima di essere state concluse, e questo è ovviamente assurdo.

La regola 4, infine, ha lo scopo di rendere univoca la corrispondenza tra coppie di nodi ed attività, corrispondenza che potrebbe venire meno quando alcune attività possono essere svolte in parallelo, come accade nel seguente esempio.

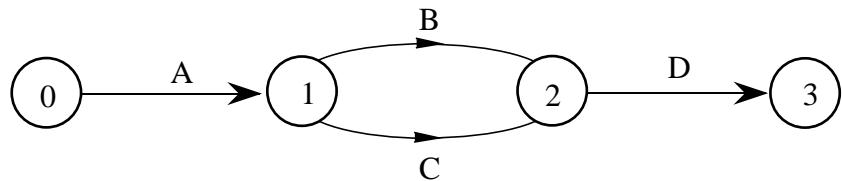


Figura 6.12: Una coppia di nodi che non individua univocamente un' attività

Esempio 2. Consideriamo le attività A, B, C, D, con le relazioni di precedenza $A < B$, $C < D$. Il grafo costruito ignorando la regola 4 è quello di 6.12, in cui alla coppia di nodi (1,2) non è associata in maniera univoca un'attività.

Quando la regola 4 non è esplicitamente soddisfatta dal progetto, come accade nell' Esempio 2, occorre ricorrere ad un artificio che consiste nell'introdurre un' *attività fittizia*, cui va associato un tempo di esecuzione nullo: nel caso dell'Esempio 2, introducendo l'attività fittizia X si ottiene il grafo della 6.13,

che rispetta la regola 4. Con l' introduzione di attività fittizie è quindi possibile individuare ogni attività

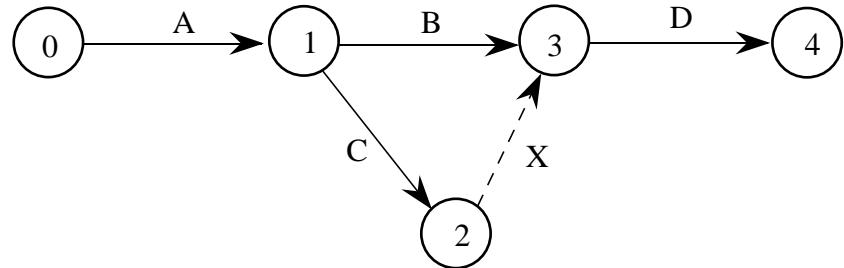


Figura 6.13: Introduzione di un' attività fittizia

mediante la coppia ordinata dei nodi estremi.

Esercizio 1. Un progetto comporta l'esecuzione delle 7 attività A, B, C, D, E, F, G, tra cui sussistono le relazioni di precedenza: A < B,C; C < D,E; D,E < F; B,F < G . Si costruisca il diagramma reticolare del progetto.

Il progetto considerato nell'esercizio precedente (il cui diagramma reticolare è dato in fondo al capitolo) sarà in seguito riutilizzato a scopo di esercizio; ad esso faremo riferimento con il nome di progetto P2.

Un altro caso in cui è richiesta l'introduzione di un'attività fittizia si verifica quando due attività precedono entrambe una terza attività, e una sola delle due ne precede una quarta. In questo caso è solo l'introduzione di un'attività fittizia che rende possibile la costruzione del grafo, come si vede nel prossimo esempio.

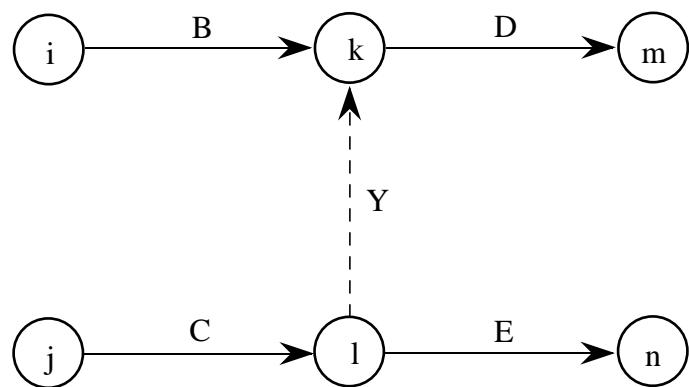


Figura 6.14: Introduzione dell' attività fittizia Y

Esempio 3. Un progetto prevede, tra le altre, le attività B, C, D, E, che devono essere svolte rispettando le precedenze: B,C < D ; C < E . Dal diagramma di figura 6.14 si rileva come solo l'introduzione dell'attività fittizia Y rende possibile la rappresentazione di questa parte del progetto.

Nel diagramma reticolare ogni nodo (ad eccezione del primo e dell'ultimo) rappresenta il termine di alcune attività e l'inizio di altre. Pertanto, in questo contesto, i nodi vengono anche chiamati eventi.

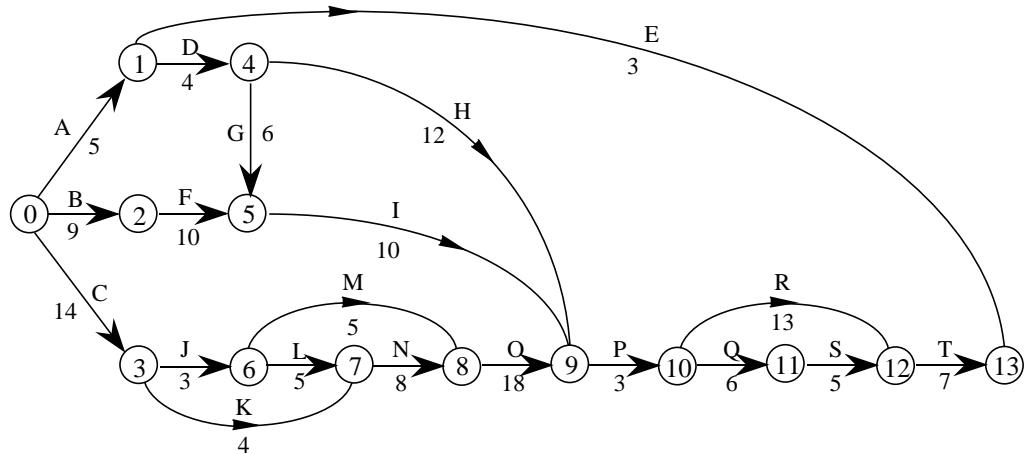


Figura 6.15: Diagramma reticolare del progetto P3

Esempio 4 Un progetto consiste nell'esecuzione di una serie di attività indicate con A, B, ..., T con le seguenti relazioni di precedenza: A, B, C possono iniziare immediatamente; D, E > A; F > B; G, H > D; I > F; G, J, K > C; M, L > J; N > K, L; O > M, N; P > H, I, O; R, Q > P; S > Q; T > R, S. Costruiamo il diagramma reticolare del progetto, numerando i nodi in modo che se il ramo (i, j) rappresenta un'attività, risulta $i < j$. Utilizzando le regole prima elencate, otteniamo il grafo di figura 6.15.

Anche il progetto considerato nell'Esempio 4 verrà riutilizzato nel seguito; ad esso faremo riferimento con il nome di progetto P3.

Domanda 1. Sai descrivere a cosa corrisponde il verificarsi dell'evento 9 nel diagramma reticolare del progetto P3 rappresentato in figura 6.15?

Domanda 2. Nel costruire il diagramma reticolare del progetto P3 è stato necessario introdurre attività fittizie?

Esercizio 2. Supponiamo che alle relazioni di precedenza del progetto P3 debba essere aggiunta la C < F. Come si modifica il grafo di figura 6.15?

Esercizio 3. Supponi che il progetto P3 sia ampliato con l'introduzione dell'attività U, per cui sussistono le relazioni di precedenza U > M, N; U < Q. Come si modifica il grafo di figura 6.15?

Il percorso critico

Abbiamo finora visto come sia possibile costruire un grafo che rappresenti l'esecuzione di un progetto, dopo che il progetto stesso è stato decomposto in attività, o fasi, di cui si siano analizzate le relazioni di precedenza. Non abbiamo però finora tenuto conto del tempo richiesto per l'esecuzione delle varie attività che compongono il progetto, e che ovviamente condizionano il tempo di esecuzione complessivo. Poiché i metodi reticolari di programmazione hanno, tra gli altri, lo scopo di controllare i tempi di esecuzione delle attività al fine di ottenere il rispetto del tempo di completamento del progetto, occorre aggiungere alla analisi qualitativa delle precedenze già effettuata anche un'analisi quantitativa che determini i valori

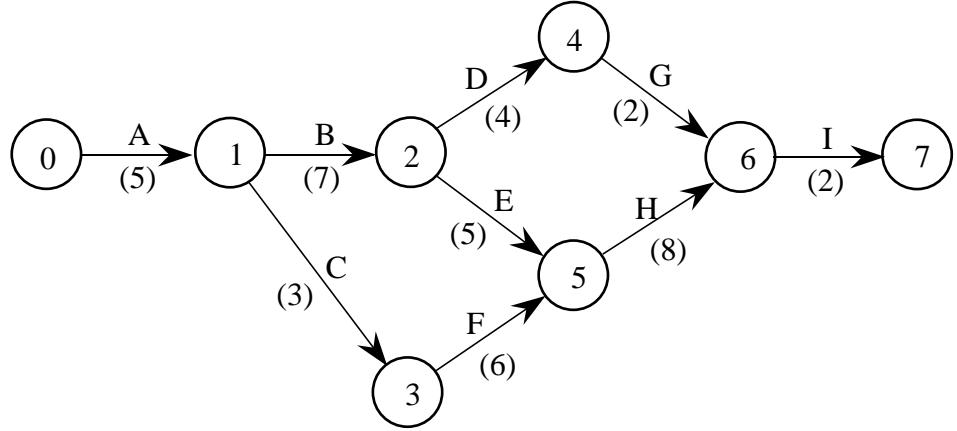


Figura 6.16: Tempi di esecuzione delle attività del progetto P1

temporali corrispondenti agli eventi descritti dal grafo, e individui i limiti entro cui tali valori temporali possono variare senza pregiudicare il valore del tempo complessivo di completamento. Per effettuare quest'analisi associamo ad ogni attività (i, j) un tempo di esecuzione t_{ij} . Il tempo di esecuzione t_{ij} può essere assunto come variabile certa, il che avviene nel CPM, o come variabile aleatoria, il che avviene nel PERT. In entrambi i casi il metodo di analisi è fondamentalmente lo stesso, per cui in questo paragrafo facciamo riferimento alla situazione in cui i tempi di esecuzione sono noti con certezza. In ogni progetto esiste un certo insieme di attività che sono di particolare importanza ai fini della determinazione del tempo di completamento dell'intero progetto, nel senso che se si verifica un ritardo nel completamento di una di queste attività, si verifica un ritardo anche nel completamento del progetto. Altre attività invece sono meno importanti, nel senso che possono anche subire un ritardo, entro certi limiti, senza che l'intero progetto ne risenta. È evidente l'importanza di distinguere tra questi due tipi di attività, così come quella di determinare il tempo minimo entro il quale certe attività intermedie possono essere completate. Quanto esposto in questo paragrafo serve proprio a consentire questa analisi. Supponiamo dunque che ad ogni attività (i, j) sia associato il tempo di esecuzione t_{ij} ; per le attività fittizie il tempo di esecuzione è ovviamente nullo.

Definizione 6.4.1 Si definisce tempo minimo di raggiungimento del nodo i , e si indica con t_i , il minimo tempo entro cui possono essere terminate tutte le attività afferenti al nodo i .

Data la definizione precedente viene del tutto naturale definire il tempo di completamento minimo del progetto nel seguente modo.

Definizione 6.4.2 Si definisce tempo minimo di completamento dell'intero progetto, e si indica con T , il tempo minimo di raggiungimento del nodo finale $T = t_f$.

In base alle regole di costruzione del diagramma reticolare di un progetto è facile convincersi che il tempo minimo di raggiungimento di un nodo i coincide con il peso del cammino massimo dal nodo iniziale al nodo i , dove i pesi degli archi sono dati dalle durate t_{ij} delle attività che essi rappresentano. Poiché il diagramma reticolare è aciclico e i nodi sono già numerati in modo topologico, è immediato applicare l'algoritmo per i cammini massimi su grafi aciclici al fine di calcolare i tempi di raggiungimento minimi.

Esempio 3. Consideriamo il diagramma reticolare del progetto P1, e associamo alle attività A, B, ..., I i seguenti tempi di esecuzione, espressi in giorni lavorativi :

tempo di esecuzione di A: $t_{01} = 5$
tempo di esecuzione di B: $t_{12} = 7$
tempo di esecuzione di C: $t_{13} = 3$
tempo di esecuzione di D: $t_{24} = 4$
tempo di esecuzione di E: $t_{25} = 5$
tempo di esecuzione di F: $t_{25} = 6$
tempo di esecuzione di G: $t_{46} = 2$
tempo di esecuzione di H: $t_{56} = 8$
tempo di esecuzione di I: $t_{67} = 2$.

Nella figura 6.4.1 i tempi di esecuzione delle attività sono stati associati ai rami del diagramma reticolare del progetto. Posto $t_0 = 0$, possiamo calcolare per i successivi nodi i tempi minimi di raggiungimento, espressi in giorni, utilizzando l' algoritmo dei cammini massimi.

per il nodo 1, $t_1 = t_0 + t_{01} = 0 + 5 = 5$
per il nodo 2, $t_2 = t_1 + t_{12} = 5 + 7 = 12$
per il nodo 3, $t_3 = t_1 + t_{13} = 5 + 3 = 8$
per il nodo 4, $t_4 = t_2 + t_{24} = 12 + 4 = 16$
per il nodo 5, $t_5 = \max(t_2 + t_{25}, t_3 + t_{35}) = \max(12 + 5, 8 + 6) = 17$
per il nodo 6, $t_6 = \max(t_4 + t_{46}, t_5 + t_{56}) = \max(16 + 2, 17 + 8) = 25$
per il nodo 7, $t_7 = t_6 + t_{67} = 25 + 2 + 27$;

avremo inoltre per il tempo minimo di completamento del progetto, $T = t_7 = 27$ giorni.

Esercizio 4. Consideriamo nuovamente il progetto P2 e associamo alle attività i seguenti tempi di esecuzione, espressi in settimane: A, 3 settimane; B, 2; C, 1; D, 4 ; E, 1; F, 2; G, . Determinare il tempo minimo di completamento del progetto.

Domanda 3 Con riferimento al progetto P1, supponiamo che il tempo di esecuzione dell'attività H passi da 8 a 10 giorni. Come varia il tempo minimo di completamento del progetto?

Domanda 4 Sempre con riferimento al progetto P1, supponiamo ora che il tempo di esecuzione dell'attività G passi da 2 a 6 giorni. Come varia il tempo minimo di completamento del progetto?

Oltre al tempo minimo di completamento dell'intero progetto, è utile introdurre anche la nozione di tempo minimo di completamento per ogni attività. Ovviamente un'attività (i, j) può avere inizio, al più presto, dopo un tempo t_i dall'inizio dell'esecuzione del progetto, in quanto perché l'attività possa avere inizio deve essere stato raggiunto il nodo i ; di conseguenza se l'attività richiede un tempo di esecuzione pari a t_{ij} , non potrà essere completata prima di un tempo pari a $t_i + t_{ij}$. Possiamo quindi dare la definizione seguente:

Definizione 6.4.3 Si definisce tempo minimo di completamento dell'attività (i, j) , e si indica con C_{ij} , il valore $C_{ij} = t_i + t_{ij}$.

Esempio 6 Consideriamo ancora il progetto P1. Per le attività del progetto, tenendo conto dei tempi di raggiungimento dei nodi calcolati nell'Esempio 5, abbiamo i seguenti tempi minimi di completamento espressi in giorni:

per l'attività A: $C_{01} = t_0 + t_{01} = 5$
per l'attività B: $C_{12} = t_1 + t_{12} = 5 + 7 = 12$
per l'attività C: $C_{13} = t_1 + t_{13} = 5 + 3 = 8$
per l'attività D: $C_{24} = t_2 + t_{24} = 12 + 4 = 16$
per l'attività E: $C_{25} = t_2 + t_{25} = 12 + 5 = 17$
per l'attività F: $C_{35} = t_3 + t_{35} = 8 + 6 = 14$

per l'attività G: $C_{46} = t_4 + t_{46} = 16 + 2 = 18$

per l'attività H: $C_{56} = t_5 + t_{56} = 17 + 8 = 25$

per l'attività I: $C_{67} = t_6 + t_{67} = 25 + 2 = 27$.

Nelle tecniche reticolari di programmazione sono di fondamentale importanza le cosiddette attività critica di cui diamo la definizione.

Definizione 6.4.4 *Sia T il tempo minimo di completamento di un progetto, corrispondente a un insieme $\{t_{ij}\}$ di valori prefissati dei tempi di esecuzione delle singole attività. Un'attività (h, k) viene detta attività critica se un variazione positiva comunque piccola ma non nulla del suo tempo di esecuzione comporta una variazione della stessa entità nel tempo minimo di completamento del progetto; e cioè, un'attività (h, k) è critica se, sostituito t_{hk} con $t_{hk} + \Delta t$, con $\Delta t \neq 0$ il tempo minimo di completamento del progetto diventa $T + \Delta t$, per qualunque valore positivo di Δt .*

Esempio 7 Nel progetto P1 l'attività E è critica: infatti se si pone $t_{25} = 5 + \Delta t$, si ottiene per i tempi di raggiungimento dei nodi 5, 6, 7 :

per il nodo 5, $t_5 = \max(t_2 + t_{25}, t_3 + t_{35}) = \max(12 + 5 + \Delta t, 8 + 6) = 17 + \Delta t$

per il nodo 6, $t_6 = \max(t_4 + t_{46}, t_5 + t_{56}) = \max(16 + 2, 17 + \Delta t + 8) = 25 + \Delta t$

per il nodo 7, $t_7 = t_6 + t_{67} = 25 + \Delta t + 2 = 27 + \Delta t$;

invece l'attività F non critica; infatti posto $t_{35} = 6 + \Delta t$, si ottiene per il tempo di raggiungimento del nodo 5 :

$$t_5 = \max(t_2 + t_{25}, t_3 + t_{35}) = \max(12 + 5, 8 + 6 + \Delta t) = 17$$

e cioè lo stesso valore di prima, almeno fintanto che Δt non supera i 3 giorni; e ovviamente se t_5 non varia, non variano neanche i tempi minimi di raggiungimento dei nodi successivi.

La determinazione dei percorsi critici è evidentemente di fondamentale importanza nelle tecniche reticolari di programmazione. Infatti le attività critiche sono quelle su cui più stretto deve essere il controllo di chi gestisce l'esecuzione del progetto, nei casi in cui un ritardo dell'esecuzione complessiva comporta una penalità, che può essere sia economica, sia di immagine. Da quanto abbiamo visto finora, l'individuazione delle attività critiche è molto semplice: è evidente che le attività critiche sono tutte e sole le attività che appartengono ad almeno un cammino massimo dal nodo iniziale al nodo finale.

Per quanto riguarda le attività non critiche è anche possibile stimare di quanto esse possano essere ritardate senza aumentare il tempo di completamento del progetto. Un'attività non critica, per esempio l'attività F nel progetto P1, non appartiene a un cammino massimo dal nodo iniziale al nodo finale, ma appartiene comunque a cammini che vanno dal nodo iniziale al nodo finale (nel caso specifico considerato uno solo, in generale più di uno). Sia allora L_{\max} la lunghezza massima di un percorso dal nodo iniziale al nodo finale *che passa per F*. È chiaro che l'attività F è l'unica a subire un ritardo, il ritardo massimo che è possibile tollerare senza che aumenti la durata del progetto è $T - L_{\max}$. Questo tempo è chiamato tempo di slittamento dell'attività. In generale:

Definizione 6.4.5 *Si definisce tempo di slittamento, o margine di tempo dell'attività (i, j) il valore che indica di quanto tempo può essere ritardato il completamento dell'attività (i, j) senza che si determini un aumento del tempo minimo di completamento dell'intero progetto.*

Ovviamente le attività critiche hanno tempo di slittamento nullo, le attività non critiche hanno un tempo di slittamento positivo. Segnaliamo che è possibile determinare in maniera molto semplice e efficiente i tempi di slittamento di tutte le attività di un progetto, noi non approfondiamo qui ulteriormente la questione per mancanza di tempo.

6.4.2 Gestione delle scorte.

Il problema di cui ci occuperemo in questo capitolo è il cosiddetto problema di Gestione delle Scorte. Si tratta di un problema di programmazione della produzione industriale che consiste nel decidere, dato un

particolare prodotto, le quantità da produrre o da immagazzinare in modo da soddisfare una domanda presente o futura. Tipicamente per la pianificazione si considera un numero ristretto di periodi di tempo, ad esempio i prossimi dodici mesi o le prossime 20 settimane. In generale indicheremo con $\{1, \dots, T\}$ l'orizzonte temporale, che è composto di T periodi detti *periodi di controllo*. Per ogni periodo è nota una domanda del bene d_t . Ad esempio, se il bene consiste in tondini metallici, per ogni giorno (o settimana o mese) del nostro orizzonte di pianificazione sono note le tonnellate richieste dal mercato. Ora, la domanda giornaliera di tondini deve essere integralmente soddisfatta. Io posso scegliere se produrre quotidianamente la quantità richiesta, oppure produrre solo in alcuni giorni (ad esempio una volta a settimana) una quantità sufficiente a soddisfare la domanda di più giorni successivi, mettendo in magazzino ciò che invierò al mercato nei giorni futuri. Ovviamente, sia produrre che immagazzinare costa, e il costo può variare da periodo a periodo. Di seguito indicheremo con c_t il costo unitario di produzione nel periodo t , mentre indicheremo con h_t il costo unitario di immagazzinamento, per $t = 1 \dots T$. Infine, iniziare un lotto di produzione presenta i cosiddetti costi di *setup*, ovvero costi amministrativi e di avviamento. Questi costi sono fissi, non dipendono cioè dalle quantità prodotte, e vengono effettivamente sostenuti solo se si produce qualcosa. Chiameremo con f_t i costi fissi di set-up relativi al periodo t , per $t = 1 \dots, T$.

Il problema è quello di scegliere quando (in che giorni) e quanto produrre in modo da minimizzare i costi di produzione (fissi e variabili) e i costi di immagazzinamento.

Una formulazione di PL01.

Introduciamo innanzitutto una variabile reale non-negativa p_t che indica il livello di produzione e una variabile reale non-negativa s_t che indica il livello delle scorte di magazzino per ogni periodo $t = 1, \dots, T$. Nel primo periodo il magazzino è vuoto (per ipotesi): quindi la domanda d_1 deve essere soddisfatta dalla produzione p_1 . Se $p_1 > d_1$, cioè produciamo più di quanto domandato, la parte residua finirà nel magazzino per servire la domanda dei periodi successivi. In particolare, avendo indicato con s_1 il livello delle scorte accumulate nel primo periodo, si avrà:

$$p_1 = d_1 + s_1 \quad (6.1)$$

Per i periodi successivi al primo, la quantità prodotta p_t nel periodo t si va ad aggiungere alle scorte accumulate nel periodo precedente s_{t-1} . Quindi, nel periodo t avremo a disposizione una quantità di bene pari a $p_t + s_{t-1}$. Una parte di questa quantità servirà a soddisfare la domanda d_t nel periodo t , mentre il resto rimarrà in magazzino e cioè sarà il livello di scorte s_t nel periodo t (utilizzabili nel periodo successivo). Quindi possiamo scrivere

$$p_t + s_{t-1} = d_t + s_t \quad t = 1, \dots, T \quad (6.2)$$

Per tenere conto del costo fisso di produzione f_t che va pagato solo nei periodi per cui si abbia $p_t > 0$, dobbiamo aggiungere una variabile booleana x_t per $t = 1, \dots, T$ con il seguente significato

$$x_t = \begin{cases} 1 & \text{se si produce nel periodo } t \ (p_t > 0) \\ 0 & \text{altrimenti.} \end{cases}$$

Per esprimere con un vincolo che $x_t = 1$ quando $p_t > 0$, o, equivalentemente, che possiamo produrre nel periodo t solo se $x_t = 1$ introduciamo il seguente vincolo:

$$p_t \leq M \cdot x_t \quad t = 1, \dots, T \quad (6.3)$$

dove M è una costante sufficientemente grande. Il ruolo del vincolo 6.3 è il seguente: se $x_t = 0$, allora non è possibile produrre ($p_t \leq 0 \rightarrow p_t = 0$). Se $x_t = 1$ allora $p_t \leq M$ e noi possiamo produrre una quantità compresa fra 0 e M . Come si è detto M deve essere scelto opportunamente in modo da rendere il vincolo sempre soddisfatto quando $x_t = 1$. Ad esempio, poiché in ogni periodo non ha senso produrre più di quanto viene richiesto durante tutto l'orizzonte temporale, un valore ammissibile per M può essere $M = \sum_{t=1}^T d_t$.

Siamo ora in grado di scrivere la funzione obiettivo, che si compone di tre termini:

- Il costo variabile di produzione, pari a $\sum_{t=1}^T c_t p_t$.
- Il costo fisso di produzione, pari a $\sum_{t=1}^T f_t x_t$.
- Il costo (variabile) di immagazzinamento, pari a $\sum_{t=1}^T h_t s_t$.

La funzione obiettivo risulta quindi essere:

$$\min \sum_{t=1}^T c_t p_t + f_t x_t + h_t s_t \quad (6.4)$$

Uno sguardo alla “struttura” del problema

Nelle righe precedenti abbiamo formulato il problema di gestione delle scorte come problema di programmazione mista. Per la soluzione del modello potremmo utilizzare algoritmi generali quali il branch-and-bound. Tuttavia, studiando più a fondo la struttura del problema, faremo vedere come sia possibile applicare algoritmi molto più efficienti per identificare una soluzione ottima.

Una prima osservazione riguarda la struttura delle soluzioni ottime. In particolare, si può facilmente dimostrare che esiste sempre una soluzione ottima per cui sia $p_t \cdot s_{t-1} = 0$ per $t = 1, \dots, T$. In altri termini, se decidiamo nel periodo t di soddisfare la domanda d_t con scorte provenienti dal periodo precedente non produrremo niente; di converso, se produciamo qualcosa ($p_t > 0$) allora il livello delle scorte precedenti s_{t-1} deve essere 0 e tutta la domanda sarà soddisfatta dalla produzione attuale. Quindi, possiamo distinguere i periodi di controllo in periodi in cui si produce (per cui $p_t > 0$ e $s_{t-1} = 0$) e periodi in cui non si produce (per cui $s_{t-1} > 0$ e $p_t = 0$). La domanda di ciascun periodo produttivo è soddisfatta dalla produzione, mentre nei periodi non produttivi la domanda è soddisfatta dalle scorte. Se i è un periodo produttivo e j è il prossimo periodo produttivo, la domanda relativa ai periodi compresi fra i e $j-1$ è servita dalla produzione realizzata in i . Quindi, in i io dovrò produrre $d_i + d_{i+1} + d_{i+2} \dots + d_{j-1} = \sum_{t=i}^{j-1} d_t$. Calcoliamo ora il costo complessivo $C(i, j)$ di produrre in i per soddisfare la domanda fino a j . Poichè la produzione in i sarà $p_i = \sum_{t=i}^{j-1} d_t$, il suo costo ammonterà a $c_i * \sum_{t=i}^{j-1} d_t$; a questo costo va' aggiunto il costo fisso di produzione del periodo i , f_i ; infine, bisogna calcolare periodo per periodo il livello delle scorte. Ricordiamo che le scorte nel periodo k sono determinate dall'equazione $s_k = p_k + s_{k-1} - d_k$ (e cioè le scorte che accumulo in un certo periodo sono pari alle scorte del periodo precedente, più la produzione nel periodo, meno ciò che se ne va per soddisfare la domanda del periodo). Quindi, nel periodo i le scorte saranno pari a $s_i = p_i + s_{i-1} - d_i$; essendo $p_i = d_i + d_{i+1} + \dots + d_{j-1} = \sum_{t=i}^{j-1} d_t$, ed avendo inoltre $s_{i-1} = 0$, sarà: $s_i = \sum_{t=i}^{j-1} d_t - d_i = \sum_{t=i+1}^{j-1} d_t$ e il suo costo sarà pari quindi $h_i * \sum_{t=i+1}^{j-1} d_t$. Nel periodo successivo, $i+1$ (che supponiamo momentaneamente non essere un periodo produttivo), il livello delle scorte sarà $s_{i+1} = p_{i+1} + s_i - d_{i+1}$, ed essendo $p_{i+1} = 0$, $s_i = \sum_{t=i+1}^{j-1} d_t$ avremo $s_{i+1} = \sum_{t=i+1}^{j-1} d_t - d_{i+1} = \sum_{t=i+2}^{j-1} d_t$, ed il costo sarà $h_{i+1} * \sum_{t=i+2}^{j-1} d_t$. Quindi, il costo complessivo delle scorte mantenute dal periodo i al periodo $j-1$ sarà pari a $h_i * \sum_{t=i+1}^{j-1} d_t + h_{i+1} * \sum_{t=i+2}^{j-1} d_t + \dots + h_{j-2} * d_{j-1}$. Riassumendo, il costo complessivo di produrre in i per soddisfare la domanda fino a $j-1$ sarà pari a $C(i, j) = c_i * \sum_{t=i}^{j-1} d_t + f_i + h_i * \sum_{t=i+1}^{j-1} d_t + h_{i+1} * \sum_{t=i+2}^{j-1} d_t + \dots + h_{j-2} * d_{j-1}$.

Ora, una soluzione (livello di produzione e livello di scorte in ogni periodo) è completamente determinata una volta stabilito l'insieme dei periodi produttivi (periodi in cui la produzione è positiva). Infatti, se $I = \{1, i, j, \dots, r, q\}$ è l'insieme ordinato dei periodi produttivi, allora vuol dire che si produce in 1 per soddisfare la domanda fino a $i-1$, quindi si produce in i per soddisfare la domanda fino a $j-1$, etc. etc., quindi si produce in r per soddisfare la domanda fino a $q-1$ e si produce in q per soddisfare la domanda fino a T . Il costo della soluzione sarà $C(I) = C(1, i) + C(i, j) + C(j, k) + C(k, \dots) + \dots + C(\dots, r) + C(r, q)$.

Quindi, il problema di gestione delle scorte può essere riformulato come il problema di trovare l'insieme di periodi produttivi I^* che minimizzi il costo $C(I^*)$.

Rappresentazione del problema di gestione delle scorte come problema di cammino minimo su grafo

Vediamo ora come il problema di trovare un insieme di istanti produttivi ottimale può essere agevolmente ricondotto a un problema di cammino minimo su grafi aciclici. A tal scopo definisco un grafo orientato $G = (V, E)$. L'insieme dei nodi è l'insieme dei periodi di controllo più un nodo fittizio che denominiamo $T + 1$, e cioè $V = \{1, \dots, T, T + 1\}$. L'insieme degli archi è l'insieme di tutti gli archi "in avanti", e cioè $E = \{(i, j) : i \in V, j \in V, i < j\}$. E' facile che con questa definizione degli archi, l'insieme dei nodi risulta ordinato topologicamente, e cioè gli archi vanno sempre da nodi con indice più piccolo a nodi con indice più grande. Quindi, il grafo G è privo di cicli orientati (vedi il teorema 6.2.1). Associamo inoltre a ogni arco (i, j) il peso $p(i, j) = C(i, j)$.

Ora, a ogni cammino $P = \{1, i, j, k, \dots, r, q, T + 1\}$ dal nodo 1 al nodo $T + 1$ associamo l'insieme di periodi produttivi $I(P) = \{1, i, j, \dots, r, q\}$. Per quanto visto sopra, il costo $C(I) = C(1, i) + C(i, j) + C(j, k) + C(k, \dots) + \dots + C(\dots, r) + C(r, q)$; ma, per come sono stati definiti i pesi sugli archi di G , è facile vedere che $C(I) = p(P)$. Quindi, il peso del cammino minimo da 1 a $T + 1$ in G corrisponde al costo di un insieme di istanti produttivi di costo minimo.

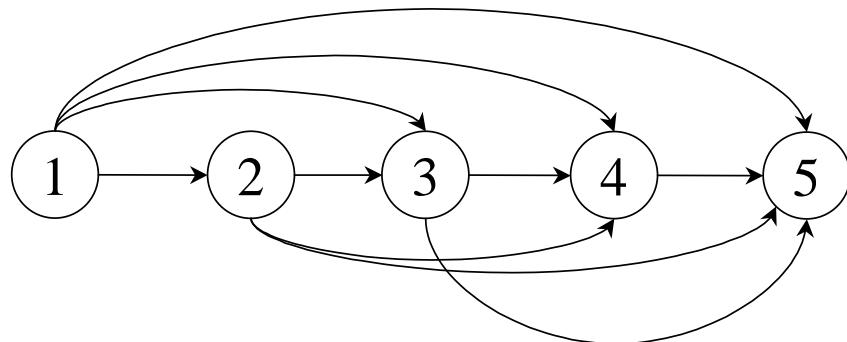
Esempio

Si consideri l'esempio riportato in tabella.

Periodo	Domanda	f	c	h
1	20	30	3	2
2	30	40	3	2
3	40	30	4	1
4	30	50	4	1

Per prima cosa calcoliamo costruiamo il grafo associato all'istanza del problema. Si tratta di un nodo con 5 nodi, uno in più dei periodi di controllo: quindi $V = \{1, 2, 3, 4, 5\}$ e tutti gli archi "in avanti".

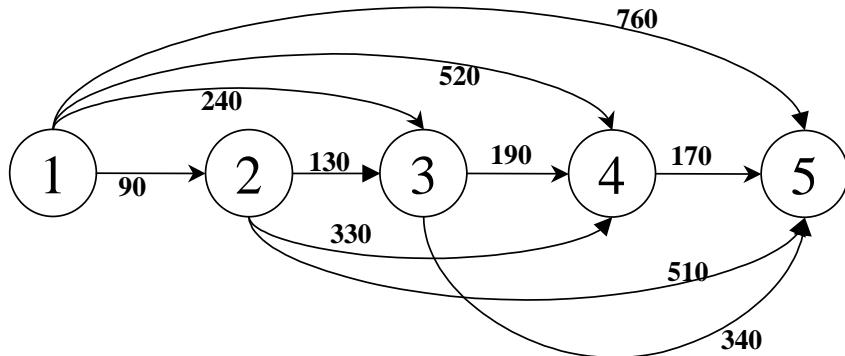
Figura 6.17: Il grafo dei periodi



Per prima cosa calcoliamo i pesi degli archi. L'arco $(1, 2)$ ha peso pari a $C(1, 2)$, cioè il costo di produrre nel periodo 1 per soddisfare la domanda fino al periodo $2 - 1 = 1$. Poichè la domanda è

pari a $d_1 = 20$, il costo variabile di produzione sarà $c_1 d_1 = 3 \times 20 = 60$; a questo va' aggiunto il costo fisso di produzione nel periodo 1, $f_1 = 30$. Non ci sono scorte residue e quindi il costo complessivo $C(1, 2) = 60 + 30 = 90$. Calcoliamo ora il costo $C(1, 3)$, cioè il costo di produrre nel periodo 1 per soddisfare la domanda fino al periodo $3 - 1 = 2$. La domanda complessiva sarà $d_1 + d_2 = 50$. Quindi il costo variabile di produzione sarà pari a $c_1(d_1 + d_2) = 3 \times 50 = 150$; a questo va' aggiunto il costo fisso di produzione nel periodo 1, $f_1 = 30$. Inoltre, una parte della produzione (d_2) dovrà essere immagazzinata per un periodo (da 1 fino a 2), e quindi pagheremo un costo d'immagazzinamento pari a $s_1 * d_2 = 2 \times 30 = 60$. Quindi, il costo complessivo $C(1, 3) = 150 + 30 + 60 = 230$. Calcoliamo ora il costo $C(1, 4)$, cioè il costo di produrre nel periodo 1 per soddisfare la domanda fino al periodo $4 - 1 = 3$. La domanda complessiva sarà $d_1 + d_2 + d_3 = 90$. Quindi il costo variabile di produzione sarà pari a $c_1(d_1 + d_2 + d_3) = 3 \times 90 = 270$; a questo va' aggiunto il costo fisso di produzione nel periodo 1, $f_1 = 30$. Inoltre, una parte della produzione ($d_2 + d_3$) dovrà essere immagazzinata nel periodo 1 fino al periodo 2, e un'altra parte d_3 dovrà essere immagazzinata nel periodo 2 fino al periodo 3 e quindi pagheremo un costo d'immagazzinamento pari a $s_1(d_2 + d_3) + s_2 d_3 = 2 \times 70 + 2 \times 40 = 220$. Quindi, il costo complessivo $C(1, 4) = 270 + 30 + 220 = 520$. A titolo esemplificativo, calcoliamo un ultimo peso, il peso dell'arco $(3, 5)$ corrispondente a $C(3, 5)$.

Figura 6.18: Calcolo dei pesi sugli archi

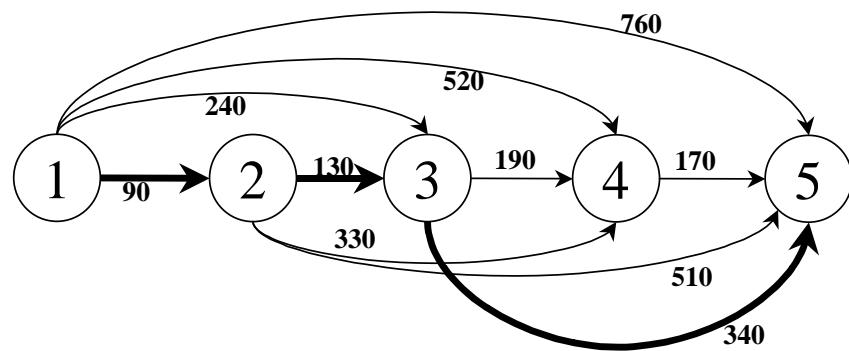


A questo punto, applichiamo l'algoritmo per il calcolo di un cammino minimo su grafi aciclici per calcolare un cammino di peso minimo dal nodo 1 al nodo 5.

- $f(1) = 0$;
 $J(1) = 1$;
- $f(2) = f(1) + p(1, 2) = 90$;
 $J[2] = 1$.
- $f(3) = \min\{f(1) + p(1, 3), f(2) + p(2, 3)\} = \min\{240, 220\} = 220$;
 $J(3) = 2$.
- $f(4) = \min\{f(1) + p(1, 4), f(2) + p(2, 4), f(3) + p(3, 4)\} = \min\{520, 420, 410\} = 410$;
 $J(4) = 4$.
- $f(5) = \min\{f(1) + p(1, 5), f(2) + p(2, 5), f(3) + p(3, 5), f(4) + p(4, 5)\} = \min\{760, 600, 560, 580\} = 560$;
 $J(5) = 3$.

A questo punto è possibile ricostruire il cammino minimo da 1 a 5 utilizzando il vettore dei predecessori. Infatti, il predecessore di 5 è il nodo 3, il predecessore di 3 è il nodo 2 e il predecessore di 3 è il nodo 1.

Figura 6.19: Cammino minimo da 1 a 5



Capitolo 7

Massimo flusso

In questo capitolo esaminiamo un importante problema di PL su grafi, il problema del *flusso massimo* in un grafo; dopo aver presentato il problema e alcune delle sue caratteristiche fondamentali, studieremo un algoritmo per la sua soluzione e considereremo inoltre alcune sue generalizzazioni ed estensioni.

In maniera molto discorsiva, possiamo immaginare che il grafo sia una rete idrica con il quale si tenta di portare dell'acqua da un nodo sorgente s ad un utilizzatore rappresentato da un nodo t . Per far questo si usa una rete di tubature rappresentata dagli archi. Le tubature hanno una certa capacità c_{uv} (cioè possono trasportare al più c_{uv} litri di acqua al secondo). Le tubature si intersecano nei nodi, dove può essere deciso, come l'acqua che arriva nel nodo deva essere distribuita ai tubi "in uscita". Si tratta di determinare come incanalare l'acqua in modo da massimizzare il flusso da s a t ; da qui il nome del problema di massimo flusso. Ovviamente, l'interpretazione idrica è solo una delle molte possibili. In fatti si potrebbe trattare di una rete di trasporto di energia elettrica (e allora i rami rappresenterebbero cavi elettrici), di persone (e allora i rami rappresenterebbero tratte servite da aerei, treni, macchine etc.), di informazioni etc.

7.1 Il problema del massimo flusso.

Sia dato un grafo orientato $G = (V, E)$ con $V = \{s, t, 3, \dots, n\}$ ed $|E| = m$ e sia data una capacità $c_{uv} \in \mathbb{R}^+$ associata a ciascun arco $(u, v) \in E$. Supponiamo che esistano

- un nodo s detto *sorgente* con solo archi uscenti ($\omega^-(s) = \{\emptyset\}$);
- un nodo t detto *pozzo* con solo archi entranti ($\omega^+(t) = \{\emptyset\}$).

I nodi distinti da s e da t sono detti *nodi intermedi*.

Un vettore $\bar{x} \in \mathbb{R}^m$ (notare che \bar{x} ha un numero di componenti pari alla cardinalità dell'insieme di archi di G) viene detto *flusso s-t ammissibile* o, per brevità *flusso ammissibile*, se soddisfa i seguenti vincoli:

$$0 \leq \bar{x}_{uv} \leq c_{uv} \quad \text{per ogni } (u, v) \in E. \quad (7.1)$$

$$\sum_{(u, v) \in \omega^-(v)} \bar{x}_{uv} - \sum_{(v, k) \in \omega^+(v)} \bar{x}_{vk} = 0, \quad \forall v \in V - \{s, t\} \quad (7.2)$$

I vincoli (7.1) indicano che il flusso su ogni arco è sempre non-negativo e non eccede la capacità dell'arco stesso. I vincoli (7.2) indicano che per ogni nodo intermedio il flusso totale entrante nel nodo eguali il flusso totale uscente dal nodo stesso. Per questo motivo, le $n - 2$ equazioni (7.2) sono dette *equazioni di conservazione* (del flusso).

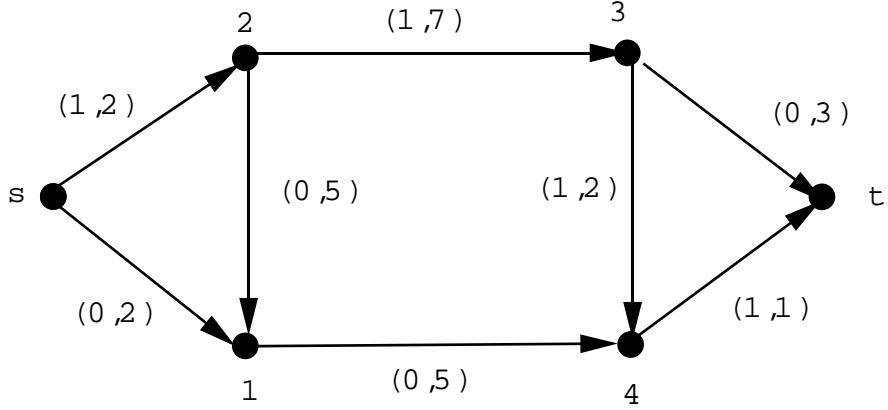


Figura 7.1: Flusso ammissibile

Consideriamo l'esempio di figura 7.1. Nella figura, fra parentesi tonde, si è indicato per ogni arco e il valore del flusso \bar{x}_e e la capacità c_e . Ad esempio, nell'arco $(s, 1)$, il flusso vale 0 mentre la capacità 2. Si può facilmente verificare che il flusso indicato è ammissibile.

Dato un flusso $s-t$, diremo *valore* del flusso, il flusso uscente dal nodo s , e cioè la quantità:

$$\bar{f} = \sum_{(s,u) \in \omega^+(s)} \bar{x}_{su} \quad (7.3)$$

Nel grafo di figura 7.1, il valore del flusso \bar{f} è uguale a 1.

Il problema del **massimo flusso** si può ora enunciare come segue:

Trovare un flusso ammissibile nel grafo G in modo che il valore del flusso \bar{f} sia massimo.

Questo problema può essere formulato come un problema di PL. Associamo una variabile x_{uv} ad ogni arco $(u, v) \in E$. Tale variabile rappresenta il flusso inviato sull'arco (u, v) . Inoltre associamo una variabile f alla quantità totale di flusso inviata da s a t . Da quanto sopra detto, il problema (MF) verrà formulato nel seguente modo:

$$\max f \quad (7.4)$$

$$- \sum_{(s,k) \in \omega^+(s)} x_{sk} + f = 0 \quad (7.5)$$

$$\sum_{(u,v) \in \omega^-(v)} x_{uv} - \sum_{(v,k) \in \omega^+(v)} x_{vk} = 0, \quad \forall v \in V - \{s, t\} \quad (7.6)$$

$$x_{uv} \leq c_{uv} \quad \text{per ogni } (u, v) \in E. \quad (7.7)$$

$$x_{uv} \geq 0 \quad \text{per ogni } (u, v) \in E. \quad (7.8)$$

Il primo vincolo (7.5) indica che la somma dei flussi sugli archi uscenti da s è uguale al flusso totale f .

Si ha quindi un vincolo per ogni nodo intermedio (7.6), che rappresenta il fatto che il flusso entrante nel nodo è uguale al flusso uscente (conservazione del flusso). Si può verificare facilmente che i vincoli (7.5) e (7.6) implicano che la somma dei flussi sugli archi entranti in t è uguale al flusso totale f .

Quindi abbiamo i vincoli (7.7) che rappresentano il fatto che il flusso su ogni arco non può eccedere la capacità dell'arco stesso.

Infine ci sono i vincoli di non negatività delle variabili (7.8).

Notiamo che la regione ammissibile di questo problema di PL è non vuota (il vettore di flusso $\bar{x} = 0$ è ovviamente ammissibile) e limitata (per la presenza dei vincoli (7.7) e (7.8)). Quindi, per il teorema fondamentale della programmazione lineare ammette una soluzione ottima e un valore ottimo della funzione obiettivo, che verrà indicato nel seguito con f^* .

Molte altre considerazioni, sia teoriche sia algoritmiche, potrebbero essere fatte sul problema del massimo flusso a partire dalla sua formulazione come problema di PL. Intanto, per fare un esempio, è evidente che il problema del massimo flusso potrebbe essere risolto con il metodo del Simplex studiato nella prima parte del corso. Questo non è tuttavia l'algoritmo migliore applicabile in questo caso (notiamo che il problema ha una "struttura" particolare che può e deve essere sfruttata).

In questo capitolo, per semplicità e brevità, noi studieremo le proprietà del problema di massimo flusso e un algoritmo per la sua soluzione a partire dalla sua formulazione come problema su grafi. Questo approccio, se da una parte non mette pienamente in luce i rapporti con la teoria della programmazione lineare, ha il vantaggio di essere più semplice e di permettere una più immediata interpretazione intuitiva dei risultati che via via esporremo.

7.2 Alcuni risultati preliminari

Nel problema del massimo flusso un ruolo centrale è svolto dal concetto di taglio $s-t$.

Definizione 7.2.1 *Dato un grafo orientato $G = (V, E)$, con $s, t \in V$, e $s \neq t$, si definisce taglio $s-t$ una partizione dei nodi (W, \bar{W}) tale che $s \in W$ e $t \in \bar{W}$.*

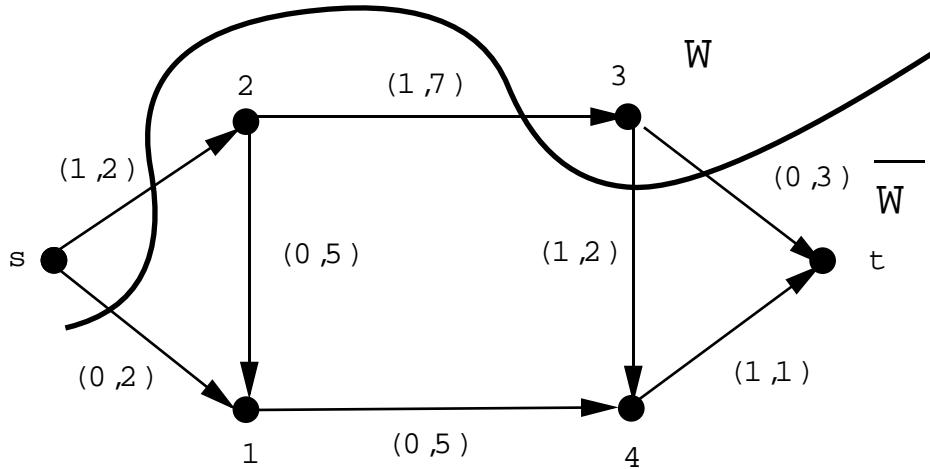


Figura 7.2: Taglio $s-t$ con $W = \{s, 3\}$ e $\bar{W} = \{1, 2, 4, t\}$

Nella figura 7.2 è illustrato un taglio $s-t$, con $W = \{s, 3\}$ e $\bar{W} = \{1, 2, 4, t\}$. È importante notare come una qualsiasi partizione dei nodi in due classi, tale che s e t appartengano a classi diverse, definisce

un taglio $s-t$. Ad ogni taglio $s-t$ (W, \bar{W}) associamo una *capacità del taglio* $C(W, \bar{W})$ che è pari alla somma delle capacità degli archi che hanno il primo estremo in W e il secondo estremo in \bar{W} , e cioè

$$C(W, \bar{W}) = \sum_{u \in W, v \in \bar{W}} c_{uv} \quad (7.9)$$

Notiamo che gli archi che attraversano il taglio in senso opposto, ovvero che abbiano il primo estremo in \bar{W} e il secondo estremo in W , non contribuiscono alla capacità del taglio. Ad esempio, dato il taglio di figura 7.2, gli archi che hanno il primo estremo in W e il secondo estremo in \bar{W} sono gli archi $(s, 1), (s, 2), (3, 4), (3, t)$, e dunque la capacità del taglio $C(W, \bar{W}) = 2 + 2 + 2 + 3 = 9$. Notiamo inoltre che l'arco $(2, 3)$ attraversa il taglio in direzione opposta, e cioè va da \bar{W} a W , e dunque la sua capacità non contribuisce alla capacità complessiva del taglio.

Dato un flusso $s-t$ \bar{x} , è possibile associare ad ogni taglio $s-t$ (W, \bar{W}) un *flusso netto di* (W, \bar{W}) , che è pari alla somma dei flussi sugli archi uscenti da W ed entranti in \bar{W} , meno la somma dei flussi sugli archi uscenti da \bar{W} ed entranti in W :

$$F(W, \bar{W}) = \sum_{u \in W, v \in \bar{W}} \bar{x}_{uv} - \sum_{u \in \bar{W}, v \in W} \bar{x}_{vu}. \quad (7.10)$$

Per il taglio di figura 7.2 è facile verificare che $F(W, \bar{W}) = 1$.

Esiste una semplice relazione tra il flusso netto di un taglio e il valore del flusso, tale relazione è data nel seguente teorema.

Teorema 7.2.2 *Dato un flusso ammissibile \bar{x} , il flusso netto $F(W, \bar{W})$ in ogni taglio (W, \bar{W}) è pari al valore del flusso \bar{f} , cioè*

$$F(W, \bar{W}) = \bar{f}.$$

Dimostrazione. Sommando le equazioni (7.2) di conservazione del flusso relative ai soli nodi dell'insieme $W - \{s\}$ e l'equazione (7.3) $\bar{f} = \sum_{(s,u) \in \omega^+(s)} \bar{x}_{su}$, otteniamo:

$$\sum_{v \in W - \{s\}} \left(\sum_{(u,v) \in \omega^-(v)} \bar{x}_{uv} - \sum_{(v,k) \in \omega^+(v)} \bar{x}_{vk} \right) - \sum_{(s,u) \in \omega^+(s)} \bar{x}_{su} = -\bar{f}$$

ovvero, inserendo il termine relativo al nodo s nella doppia sommatoria:

$$\sum_{v \in W} \left(\sum_{(u,v) \in \omega^-(v)} \bar{x}_{uv} - \sum_{(v,k) \in \omega^+(v)} \bar{x}_{vk} \right) = -\bar{f}.$$

Notiamo il flusso relativo ad un arco (u, v) con entrambi gli estremi in W compare nella sommatoria per $\omega^+(u)$ con segno negativo e in $\omega^-(v)$ con segno positivo. Pertanto il suo contributo complessivo alla sommatoria è nullo. Dunque gli archi (u, v) che contribuiscono alla sommatoria avranno necessariamente un estremo in W e l'altro estremo in \bar{W} , ovvero saranno tali che $u \in W$ e $v \in \bar{W}$, oppure $v \in W$ e $u \in \bar{W}$. Pertanto

$$\sum_{u \in \bar{W}, v \in W} \bar{x}_{uv} - \sum_{u \in W, v \in \bar{W}} \bar{x}_{vu} = -\bar{f},$$

oppure, equivalentemente:

$$F(W, \bar{W}) = \bar{f} \quad \square$$

Se $W = V - \{t\}$ e $\bar{W} = \{t\}$ abbiamo:

$$F(W, \bar{W}) = \sum_{u \in W, v \in \bar{W}} \bar{x}_{uv} = \sum_{(u,t) \in \omega^-(t)} \bar{x}_{ut} = \bar{f}.$$

Quindi il flusso uscente dalla sorgente s è uguale al flusso netto entrante nel pozzo t .

Passiamo ora ed esaminare le relazioni tra il valore del flusso e le capacità di un taglio. È estremamente semplice provare i seguenti due risultati.

Teorema 7.2.3 *Sia dato un flusso ammissibile \bar{x} in G . Il flusso netto di un taglio qualunque è minore o uguale alla capacità dello stesso taglio. Cioè, se (W, \bar{W}) è un taglio, risulta*

$$F(W, \bar{W}) \leq C(W, \bar{W}).$$

Dimostrazione. Siccome il flusso è ammissibile soddisfa, in particolare, le (7.1). Ma allora il teorema segue immediatamente dal confronto delle definizioni di capacità flusso netto di un taglio, cioè dal confronto delle (7.9) e (7.10). \square

Notiamo, per inciso, che il flusso netto di un taglio dipende dal flusso \bar{x} che stiamo considerando, mentre la capacità dello stesso taglio è indipendente dal flusso \bar{x} e dipende invece solamente dalle capacità degli archi.

Il flusso massimo f^* è ottenuto in corrispondenza ad un particolare flusso ammissibile, che indichiamo con \bar{x}^* ; quindi, come caso particolare del precedente teorema otteniamo il seguente risultato.

Teorema 7.2.4 *Il valore del massimo flusso f^* è minore o uguale alla capacità di un qualunque taglio $s-t$, cioè*

$$f^* \leq C(W, \bar{W})$$

per ogni taglio (W, \bar{W}) .

Dimostrazione. Per il Teorema 7.2.2 abbiamo, per qualunque taglio (W, \bar{W})

$$f^* = F(W, \bar{W}). \quad (7.11)$$

Ma abbiamo anche, per il teorema precedente

$$F(W, \bar{W}) \leq C(W, \bar{W}) \quad (7.12)$$

Dalle (7.11) e (7.12) segue immediatamente

$$f^* \leq C(W, \bar{W}),$$

che è quanto volevamo dimostrare. \square

7.3 Cammini aumentanti

Un secondo elemento che gioca un ruolo fondamentale nello studio di problemi di massimo flusso è il concetto di *cammino aumentante*, che è l'oggetto di questa sezione.

Per introdurre la definizione di cammino aumentante abbiamo bisogno di una definizione preliminare

Definizione 7.3.1 *Dato un cammino semplice $P = \{u_0, e_1, u_1 \dots, e_p, u_p\}$, chiamiamo arco diretto un arco di tipo $e_i = (u_{i-1}, u_i)$, mentre chiamiamo arco inverso un arco di tipo $e_i = (u_i, u_{i-1})$ ¹.*

Definizione 7.3.2 *Data una soluzione ammissibile \bar{x} del problema di massimo flusso, definiamo cammino aumentante da u_0 a u_p un cammino semplice $P = \{u_0, e_1, \dots, e_p, u_p\}$ tale che:*

- per ogni arco diretto $e_i = (u_{i-1}, u_i)$, si ha $c_{u_{i-1}, u_i} > \bar{x}_{u_{i-1}, u_i}$
- per ogni arco inverso $e_i = (u_i, u_{i-1})$, si ha $\bar{x}_{u_i, u_{i-1}} > 0$.

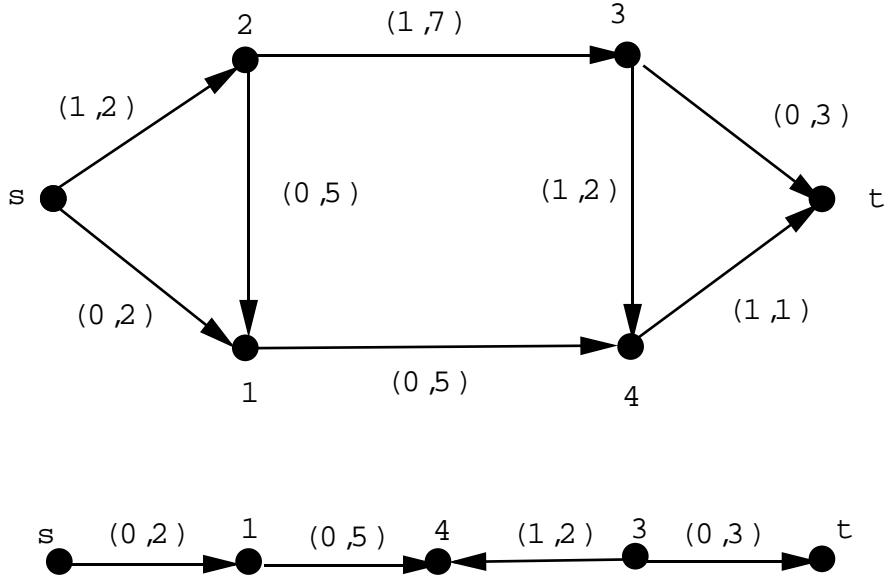


Figura 7.3: Cammino aumentante da s a t

In un cammino aumentante, dunque, gli archi diretti sono *non saturi*, mentre quelli inversi sono *non vuoti*. Consideriamo l'esempio di figura 7.3. Il cammino aumentante $\{s, 1, 4, 3, t\}$ è mostrato in basso in figura. Per ogni arco diretto (nell'esempio gli archi $(s, 1), (1, 4), (3, t)$), si ha che la capacità è strettamente maggiore del flusso. Per ogni arco inverso (nell'esempio l'arco $(3, 4)$) il flusso è strettamente positivo.

La ragione del nome di cammino aumentante deriva dal fatto che, dato un flusso ammissibile \bar{x} , una volta individuato e un cammino aumentante è facilmente definibile un nuovo flusso ammissibile \tilde{x} con un valore di flusso maggiore di quello associato ad \bar{x} . In altre parole, sfruttando un cammino aumentante è possibile aumentare il flusso inviato da s a t .

Mostriamo come ciò sia possibile. Supponiamo quindi di avere un flusso ammissibile \bar{x} e un cammino aumentante P . Indichiamo con P_+ l'insieme degli archi diretti del cammino P e con P_- l'insieme di quelli inversi. In modo formale

$$P_+ = \{e_i : e_i \in P, \text{ con } e_i \text{ arco diretto}\}$$

e

$$P_- = \{e_i : e_i \in P, \text{ con } e_i \text{ arco inverso}\}.$$

Se consideriamo l'esempio di figura 7.3 abbiamo $P_+ = \{(s1), (12), (3t)\}$ e $P_- = \{(34)\}$.

Sia $\delta_+ = \min_{(u,v) \in P_+} (c_{uv} - \bar{x}_{uv})$. È facile vedere che $\delta_+ > 0$, in quanto per ogni arco diretto si ha $c_{uv} > \bar{x}_{uv}$. Sia $\delta_- = \min_{(u,v) \in P_-} \bar{x}_{uv}$. Anche in questo caso è facile vedere che $\delta_- > 0$, in quanto per ogni arco inverso si ha $\bar{x}_{uv} > 0$. Sia $\delta = \min(\delta_+, \delta_-) > 0$. Definiamo un flusso modificato \tilde{x} nel seguente modo.

$$\tilde{x} = \begin{cases} \tilde{x}_{uv} = \bar{x}_{uv} & \text{se } (u, v) \notin P \\ \tilde{x}_{uv} = \bar{x}_{uv} + \delta & \text{se } (u, v) \in P^+ \\ \tilde{x}_{uv} = \bar{x}_{uv} - \delta & \text{se } (u, v) \in P^- \end{cases}$$

¹Con linguaggio meno preciso possiamo dire che un arco diretto è un arco il cui verso è concorde con quello di percorrenza del cammino P da u_0 a u_p , mentre un arco inverso ha direzione opposta.

Notiamo che, rispetto a \bar{x} vengono modificati solo i flussi degli archi appartenenti al cammino aumentante P . Vogliamo mostrare che, come preannunciato, \tilde{x} è ammssibile e ha un valore di flusso \tilde{f} maggiore di \bar{f} .

Mostriamo dapprima che \tilde{x} è una soluzione ammssibile. Le variabili corrispondenti ad archi non appartenenti a P non vengono modificate rispetto alla soluzione ammssibile \bar{x} e dunque non violano i vincoli di capacità (7.7) e di non negatività (7.8). Le variabili corrispondenti agli archi di P^+ vengono aumentate di δ , quindi i vincoli di non negatività (7.8) non vengono violati. Inoltre, poichè $\delta \leq \delta^+ \leq \min(c_{uv} - \bar{x}_{uv})$, i vincoli di capacità (7.7) non sono violati. Le variabili corrispondenti agli archi di P^- vengono diminuite di δ , quindi i vincoli di capacità (7.7) non sono violati. Inoltre, poichè $\delta \leq \delta^- \leq \min(\bar{x}_{uv})$, i vincoli di non negatività (7.8) non vengono violati. Dobbiamo ora mostrare che i vincoli di conservazione ai nodi (7.6) non sono violati. Poichè solo le variabili che corrispondono ad archi di P vengono modificate, i vincoli che potrebbero essere violati sono quelli che corrispondono a nodi di P . Per ogni nodo $u_i \in P$, con $u_i \neq s, t$ ci sono esattamente due archi incidenti in u_i appartenenti a P , l'arco e_{i-1} e l'arco e_i . Sono possibili i seguenti quattro casi:

1. e_{i-1} è diretto ed e_i è inverso su P . In questo caso $\tilde{x}_{e_{i-1}} = \bar{x}_{e_{i-1}} + \delta$, mentre $\tilde{x}_{e_i} = \bar{x}_{e_i} - \delta$. Poichè

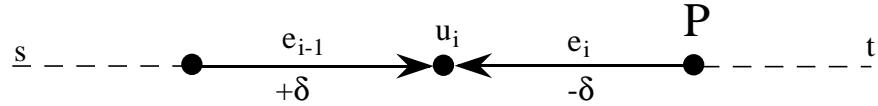


Figura 7.4: e_{i-1} diretto, e_i inverso

ambedue gli archi sono entranti in u_i , il flusso uscente da u_i resterà invariato, mentre il flusso entrante in u_i varierà di $+\delta - \delta = 0$.

2. e_{i-1} ed e_i sono ambedue diretti su P . In questo caso $\tilde{x}_{e_{i-1}} = \bar{x}_{e_{i-1}} + \delta$, mentre $\tilde{x}_{e_i} = \bar{x}_{e_i} + \delta$.

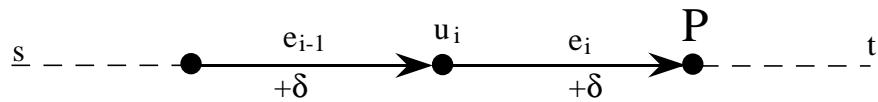


Figura 7.5: e_{i-1} diretto, e_i diretto

Poichè e_{i-1} è entrante in u_i , mentre e_i è uscente, il flusso entrante aumenta di δ e il flusso uscente aumenta di δ : l'uguaglianza fra flusso entrante e flusso uscente è preservata.

3. e_{i-1} ed e_i sono ambedue inversi su P . In questo caso $\tilde{x}_{e_{i-1}} = \bar{x}_{e_{i-1}} - \delta$, mentre $\tilde{x}_{e_i} = \bar{x}_{e_i} - \delta$.

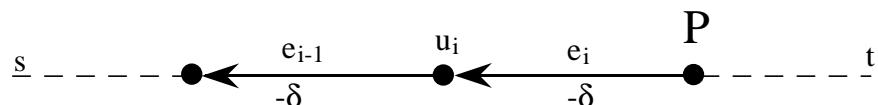


Figura 7.6: e_{i-1} inverso, e_i inverso

Poichè e_{i-1} è uscente da u_i , mentre e_i è entrante, il flusso entrante diminuisce di δ e il flusso uscente diminuisce di δ : l'uguaglianza fra flusso entrante e flusso uscente è preservata.

4. e_{i-1} è inverso ed e_i è diretto su P . In questo caso $\tilde{x}_{e_{i-1}} = \bar{x}_{e_{i-1}} + \delta$, mentre $\tilde{x}_{e_i} = \bar{x}_{e_i} - \delta$. Poichè

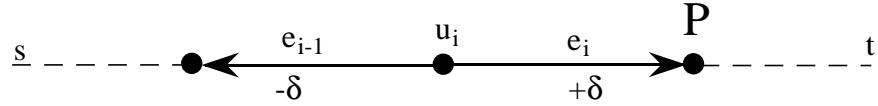


Figura 7.7: e_{i-1} inverso, e_i diretto

ambedue gli archi sono uscenti in u_i , il flusso entrante in u_i resterà invariato, mentre il flusso uscente da u_i varierà di $-\delta + \delta = 0$.

Dunque \tilde{x} è ammissibile. Inoltre, poichè il primo arco di P è sempre diretto ed uscente dal nodo s , il flusso totale uscente dal nodo s viene aumentato di δ e risulta quindi

$$\tilde{f} = \bar{f} + \delta.$$

7.4 Condizioni di ottimalità

In questa sezione ci proponiamo di trovare delle condizioni che, dato un flusso ammissibile \bar{x} , ci permettano di capire se \bar{x} sia una soluzione ottima del problema di massimo flusso o meno. La prima condizione che consideriamo è legata all'esistenza di cammini aumentanti ed ha implicazioni algoritmiche immediate. La seconda, equivalente, coinvolge la capacità dei tagli ed è interessante perché mostra come risolvendo un problema di massimo flusso possiamo contemporaneamente risolvere un altro problema apparentemente molto diverso: il "problema del minimo taglio".

Passiamo ora a considerare la prima delle due condizioni preannunciate.

Teorema 7.4.1 *Un flusso $s-t$ \bar{x} di valore \bar{f} è massimo se e solo se non esiste nessun cammino aumentante $P = \{s = u_0, e_1, \dots, u_p = t\}$ da s a t in G rispetto a \bar{x} .*

Dimostrazione.

Necessità. Se \bar{x} è ottimo non possono esistere cammini aumentanti da s a t in G rispetto a \bar{x} . Infatti, se ne esistesse uno sappiamo (vedi sezione precedente) che è possibile trovare un nuovo flusso \tilde{x} con un valore maggiore di \bar{f} . Ciò è assurdo perché \bar{x} è ottimo.

Sufficienza. Consideriamo il seguente taglio:

$$W = \{u : \text{esiste un cammino aumentante da } s \text{ a } u \text{ rispetto ad } \bar{x}\} \cup \{s\}$$

$$\bar{W} = V - W$$

Notiamo che deve risultare $t \notin W$. Infatti, se risultasse $t \in W$ esisterebbe, per definizione di W , un cammino aumentante da s a t . Ma questo implicherebbe, per quanto visto nella sezione precedente, che sarebbe possibile trovare un nuovo flusso \tilde{x} con un valore di flusso \tilde{f} maggiore di \bar{f} . Siccome questo è assurdo perchétiamo supponendo che \bar{f} sia massimo, deve essere, come affermato $t \notin W$.

Quindi (W, \bar{W}) è un taglio $s-t$. Consideriamo adesso il flusso sugli archi che attraversano il taglio da W a \bar{W} e sugli archi che attraversano il taglio da \bar{W} a W .

- Per ogni $(u, v) \in E$ tale che $u \in W$, e $v \in \bar{W}$ si ha che il flusso $\bar{x}_{uv} = c_{uv}$. Infatti, supponiamo per assurdo che $\bar{x}_{uv} < c_{uv}$. Essendo $u \in W$ esiste un cammino aumentante P' da s ad u . Se $\bar{x}_{uv} < c_{uv}$, allora il cammino $P = P' \cup (u, v)$ è un cammino aumentante da s a v . Infatti lungo l'arco diretto (u, v) si verifica la condizione $\bar{x}_{uv} < c_{uv}$. Ma allora v dovrebbe appartenere a W contraddicendo l'ipotesi che $v \in \bar{W}$.
- Per ogni $(v, u) \in E$ tale che $u \in W$, e $v \in \bar{W}$ si ha che il flusso $\bar{x}_{vu} = 0$. Infatti, supponiamo per assurdo che $\bar{x}_{vu} > 0$. Essendo $u \in W$ esiste un cammino aumentante P' da s ad u . Se $\bar{x}_{vu} > 0$, allora il cammino $P = P' \cup (v, u)$ è un cammino aumentante da s a v . Infatti lungo l'arco inverso (v, u) si verifica la condizione $\bar{x}_{vu} > 0$. Ma allora v dovrebbe appartenere a W contraddicendo l'ipotesi che $v \in \bar{W}$.

Per il Teorema (7.2.2), il flusso totale \bar{f} da s a t è pari al flusso attraverso il taglio (W, \bar{W}) che vale, per quanto appena visto,

$$\bar{f} = \sum_{u \in W, v \in \bar{W}} \bar{x}_{uv} = \sum_{u \in W, v \in \bar{W}} c_{uv} = C(W, \bar{W}).$$

Possiamo allora scrivere

$$\bar{f} = C(W, \bar{W}).$$

Ma per il Teorema 7.2.4 sappiamo che una qualunque distribuzione di flusso ammissibile non può generare un flusso totale maggiore di $C(W, \bar{W})$ e quindi il flusso \bar{f} è massimo. \square

Nel paragrafo successivo vedremo come questo teorema sia alla base di un metodo per il calcolo del massimo flusso. Prima di occuparci di questo, però, vediamo come il teorema appena visto possa essere riformulato in un modo che metta in luce le connessioni tra il problema di trovare il massimo flusso e il problema di trovare il taglio a capacità minima.

Per introdurre questo risultato facciamo delle considerazioni preliminari. Nella seconda sezione di questo capitolo abbiamo associato ad ogni taglio (W, \bar{W}) un ben definito numero: la sua capacità $C(W, \bar{W})$. Siccome il numero di tagli possibili è finito (anche se, eventualmente, molto grande) possiamo considerare tra tutti i tagli quello che ha capacità minore di tutti gli altri (ovviamente possono esistere più tagli diversi che forniscono la stessa capacità minima). Indichiamo con C_{min} il valore di questa capacità minima. Il Teorema 7.2.4 è valido per qualunque taglio, in particolare per il taglio di capacità minima. Quindi possiamo affermare che

$$f^* \leq C_{min}$$

La capacità del taglio minimo costituisce quindi un limite superiore per il valore del flusso massimo, e se troviamo un flusso \bar{x} per cui $\bar{f} = C_{min}$, \bar{x} è ovviamente un flusso ottimo².

In generale però, il valore del flusso massimo f^* potrebbe risultare inferiore o uguale a C_{min} . Il teorema del Massimo flusso - Minimo taglio (più noto, con terminologia inglese, come Max flow - Min cut theorem) stabilisce invece che un flusso \bar{x} è ottimo se e solo se il valore del flusso è esattamente C_{min} .

Teorema 7.4.2 (Massimo flusso - Minimo taglio) *Una distribuzione di flusso è ottima se e solo se risulta*

$$\bar{f} = C_{min}$$

dove \bar{f} è il valore di flusso di \bar{x} .

²Cambiando “punto di vista” possiamo anche dire che il flusso massimo è un limite inferiore per la capacità di un taglio. Quindi, se per qualche taglio (W, \bar{W}) risulta $C(W, \bar{W}) = f^*$, (W, \bar{W}) è un taglio minimo e $C(W, \bar{W}) = C_{min}$. Questo fatto verrà usato spesso nel seguito.

Dimostrazione.

Sufficienza. Mostriamo che se $\bar{f} = C_{min}$ allora \bar{x} è ottimo. Procediamo per assurdo. Supponiamo che risulti $\bar{f} = C_{min}$, ma \bar{x} non sia ottimo. Allora esiste un altro flusso, \tilde{x} il cui valore di flusso è maggiore di \bar{f} . Possiamo quindi scrivere

$$\tilde{f} > \bar{f} = C_{min}$$

che è ovviamente assurdo perché contraddice il Teorema 7.2.4.

Necessità. Supponiamo che \bar{x} sia ottimo e mostriamo che deve risultare $\bar{f} = C_{min}$. Procediamo per assurdo e supponiamo che sia $\bar{f} \neq C_{min}$. Più in particolare, siccome sappiamo che non può essere $\bar{f} > C_{min}$ per il Teorema 7.2.4, deve risultare

$$\bar{f} < C_{min}. \quad (7.13)$$

Definiamo ora lo stesso taglio considerato nel teorema precedente. Poniamo cioè $W = \{u : \text{esiste un cammino aumentante da } s \text{ a } u \text{ rispetto ad } \bar{x}\} \cup \{s\}$ e $\bar{W} = V - W$.

Ragionando come nella prova del teorema precedente, abbiamo che (W, \bar{W}) è un taglio s -te che

$$\bar{f} = C(W, \bar{W}). \quad (7.14)$$

Per le (7.13) e (7.14) possiamo allora scrivere

$$C(W, \bar{W}) \geq C_{min} > \bar{f} = C(W, \bar{W})$$

che è ovviamente assurda e così la prova del teorema è completa. □

Nella dimostrazione della necessità del teorema appena dimostrato, abbiamo anche implicitamente trovato un modo per determinare un taglio di capacità minima. Infatti, se x^* è un flusso ottimo, il taglio definito da

$$W = \{u : \text{esiste un cammino aumentante da } s \text{ a } u \text{ rispetto ad } \bar{x}\} \cup \{s\}$$

$$\bar{W} = V - W$$

è tale che $f^* = C(W, \bar{W})$ (equazione (7.14)) ed è quindi minimo per il Teorema (7.2.2). Quindi, se sappiamo risolvere un problema di massimo flusso, sappiamo anche risolvere il problema di trovare un taglio di capacità minima!

Il fatto che il massimo flusso sia minore o uguale alla capacità di ogni taglio e quindi anche alla capacità del taglio minimo è abbastanza ovvio. Meno ovvio è che all' ottimo valga sempre l' uguaglianza. Questa proprietà permette, concettualmente, di verificare se un dato flusso (ammissibile) è ottimo. Infatti per fornire la prova di ottimalità basta esibire un taglio con capacità pari al flusso inviato da s a t . Nel caso tale taglio non esista, sappiamo che deve esistere un flusso che permette di inviare un flusso maggiore da s a t .

Il teorema del Massimo flusso - Minimo taglio mette quindi in relazione un problema sostanzialmente continuo (trovare il massimo flusso inviabile da s a t) con un problema intrinsecamente combinatorio (trovare il taglio di minima capacità che separa s a t). I due problemi sono di natura diversa. In particolare, il primo cerca una soluzione in un insieme infinito di possibili soluzioni. Il secondo cerca una soluzione in un insieme finito di possibili soluzioni. Questo fatto potrebbe portare all'impressione (errata) che il primo problema sia più difficile del secondo, in quanto nel secondo è sempre possibile enumerare tutte le possibili soluzioni e scegliere la migliore. Questa impressione è, però, come ormai ben sappiamo, errata, in quanto l'insieme di tutte le soluzioni è sì finito, ma talmente grande, anche per grafi di piccole dimensioni, che non è pensabile esaminarlo tutto. Per esempio, un grafo con 300 nodi (piccolo per le applicazioni normalmente affrontate con questa metodologia) porta a 2^{288} possibili tagli separatori di due nodi assegnati. Alcuni ricercatori stimano che il numero di atomi dell'universo osservabile sia inferiore a 2^{263} .

Prima di chiudere questa sezione, vogliamo dare una formulazione equivalente del teorema del Massimo flusso - Minimo taglio che può aiutare a chiarire ulteriormente le relazioni tra i vari concetti introdotti.

7.5 L'algoritmo di Ford e Fulkerson.

I precedenti risultati ci permettono di definire un algoritmo per il calcolo del flusso massimo da s a t in un grafo orientato. Sia dato un flusso \bar{x} ammissibile. Per il Teorema 7.4.1, se non esiste in G un cammino aumentante da s a t rispetto a \bar{x} , allora \bar{x} è ottimo. Supponiamo invece che esista un cammino aumentante $P = \{s = u_0, e_1, \dots, u_p = t\}$. Il flusso \bar{x} non è ottimo, e quindi può essere "aumentato", nel senso che è possibile modificare \bar{x} in modo da aumentare il flusso totale inviato da s a t come visto nella sezione "Cammini aumentanti".

Viene allora spontaneo definire un algoritmo per il calcolo del massimo flusso nel seguente modo.

Algoritmo di Ford e Fulkerson

Inizializzazione Poni $x^0 := 0$; $i := 0$.

Passo i -esimo Cerca un cammino aumentante da s a t rispetto al flusso x^i .

Se non esiste: STOP. x^i è la soluzione ottima.

Altrimenti poni:

$$x^{i+1} = \begin{cases} x_{uv}^{i+1} = x_{uv}^i & \text{se } (u, v) \notin P \\ x_{uv}^{i+1} = x_{uv}^i + \delta & \text{se } (u, v) \in P^+ \\ x_{uv}^{i+1} = x_{uv}^i - \delta & \text{se } (u, v) \in P^- \end{cases}$$

Poni $i := i + 1$; vai al passo i .

Notiamo subito che in questo algoritmo c'è una parte non completamente specificata. Infatti all'inizio del Passo i viene detto "Cerca un cammino aumentante...", ma non viene detto né come trovare un tale cammino aumentante né quale scegliere nel caso ne esista più di uno. Queste questioni verranno affrontate nella prossima sezione. Per quel che ci interessa qui, supponiamo di essere in grado di trovare i cammini aumentanti e, nel caso ne esista più di uno, di sceglierne uno a caso.

Per chiarire meglio il procedimento illustriamolo subito con un esempio. Consideriamo di nuovo il grafo di figura 7.2. Al passo 0 avremo $x_{s1}^0 = 0, x_{s2}^0 = 0, x_{14}^0 = 0, x_{21}^0 = 0, x_{23}^0 = 0, x_{34}^0 = 0, x_{3t}^0 = 0, x_{4t}^0 = 0$. Un possibile cammino aumentante sarà: $P = \{s, (s, 2), 2, (2, 3), 3, (3, 4), 4, (4, t), t\}$. $P_+ = \{(s, 2), (2, 3), (3, 4), (4, t)\}$, mentre $P_- = \emptyset$. Quindi il primo cammino aumentante possiede solo archi diretti. $\delta = \delta_+ = \min(2 - 0, 7 - 0, 2 - 0, 1 - 0) = 1$. Quindi tutte le variabili relative agli archi del cammino aumentante vengono aumentate di una unità. Le altre resteranno invariate. Si avrà: $x_{s1}^1 = 0, x_{s2}^1 = 1, x_{14}^1 = 0, x_{21}^1 = 0, x_{23}^1 = 1, x_{34}^1 = 1, x_{3t}^1 = 0, x_{4t}^1 = 1$. Il flusso è aumentato da 0 a 1. Un nuovo cammino aumentante sarà $P = \{s, (s, 1), 1, (1, 4), 4, (3, 4), 3, (3, t), t\}$. Risulta quindi $P_+ = \{(s, 1), (1, 4), (3, 4), (3, t)\}$ e $P_- = \{(3, 4)\}$. Quindi $\delta_+ = \min(2 - 0, 5 - 0, 3 - 0) = 2$, mentre $\delta_- = \min(1) = 1$ e $\delta = \min(\delta_+, \delta_-) = 1$.

Le variabili corrispondenti ad archi diretti vengono quindi aumentate di una unità, mentre le variabili corrispondenti ad archi inversi (in questo caso la variabile x_{34}) vengono diminuite di un'unità. $x_{s1}^2 = 1, x_{s2}^2 = 1, x_{14}^2 = 1, x_{21}^2 = 0, x_{23}^2 = 1, x_{34}^2 = 0, x_{3t}^2 = 1, x_{4t}^2 = 1$. Il flusso è aumentato da 1 a 2.

Infine troviamo il cammino aumentante: $P = \{s, (s, 2), 2, (2, 3), 3, (3, t), t\}$. Sono tutti archi diretti e per essi vale $\delta = 1$. Quindi si avrà:

$x_{s1}^3 = 1, x_{s2}^3 = 2, x_{14}^3 = 1, x_{21}^3 = 0, x_{23}^3 = 2, x_{34}^3 = 0, x_{3t}^3 = 2, x_{4t}^3 = 1$. Il flusso finale vale 3, cioè la somma dei flussi uscenti da s . Per il teorema 7.4.2, per dimostrare l'ottimalità del flusso basta mostrare un taglio s -tla cui capacità è pari al valore del flusso. Bisogna cioè trovare il taglio a capacità minima.

Nell'esempio il taglio a capacità minima sarà $W = \{s, 1, 4\}$ e $\bar{W} = \{2, 3, t\}$, ed è illustrato in figura 7.8. Gli archi che hanno la coda in W e la testa in \bar{W} sono gli archi $(s, 2)$ e $(4, t)$, e la somma delle loro capacità vale 3. Per trovare questo taglio basta usare la costruzione suggerita nella dimostrazione della sufficienza del teorema 7.4.1, ovvero basta mettere in W il nodo s e tutti i nodi raggiungibili da s con un cammino aumentante rispetto alla soluzione ottima \bar{x} . Nell'esempio appena visto l'algoritmo di Ford e Fulkerson

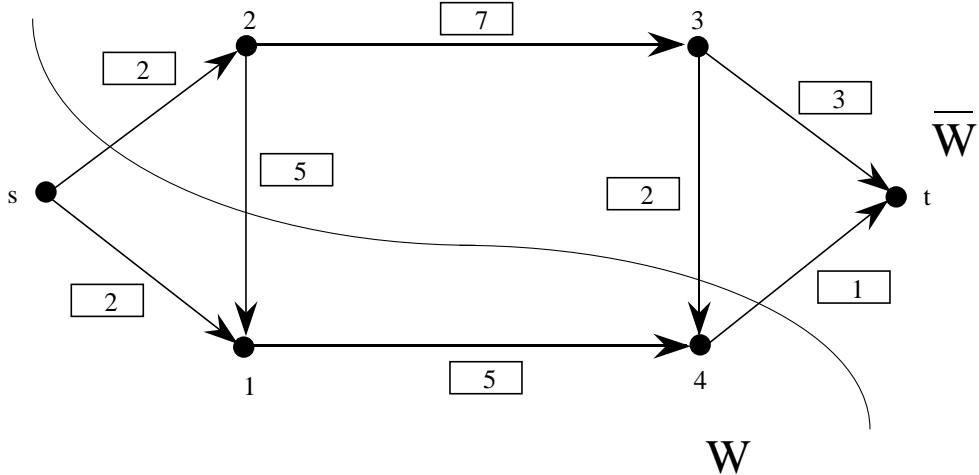


Figura 7.8: Taglio a capacità minima

ha trovato la soluzione ottima in un numero finito di iterazioni. Ma quale garanzia abbiamo che ciò accada sempre? Più in particolare, se non possiamo trovare un cammino aumentante sappiamo di essere all'ottimo, mentre in caso contrario sappiamo che l'algoritmo aumenta il valore del flusso. Ma, per esempio, possiamo pensare di poter trovare cammini aumentanti con i quali è possibile aumentare il valore del flusso di quantità δ sempre più piccole cosicché se, poniamo, il flusso ottimo vale 10, i nostri flussi abbiano valori di 9, 9.9, 9.99, 9.999 ... e non si arrivi mai al flusso ottimo.

In proposito vale il seguente risultato.

Teorema 7.5.1 *Supponiamo che tutte le capacità siano numeri interi. Allora l' algoritmo di Ford e Fulkerson trova l'ottimo in un numero finito di passi.*

Dimostrazione. Sappiamo che il flusso ottimo esiste sempre e ha quindi una valore (finito) f^* . Il flusso da cui parte l' algoritmo è quello nullo, che ha un valore di 0. Siccome le capacità sono tutte intere è facile convincersi che ad ogni passo il flusso viene aumentato di un numero intero (δ è un numero intero). Quindi ad ogni iterazione il flusso viene aumentato di almeno un'unità (δ è maggiore o uguale a 1, il più piccolo intero positivo). Questo vuol dire che in un numero finito di passi (pari, nel caso peggiore, al più piccolo intero superiore a f^*) l'algoritmo trova (= “raggiunge”) l'ottimo. \square

Con ragionamenti molto simili si può dimostrare che anche se le capacità sono numeri razionali³ l'algoritmo termina in un numero finito di passi. Invece se alcune delle capacità sono numeri irrazionali è possibile fornire degli esempi (piuttosto artificiosi in verità) in cui l'algoritmo non riesce a trovare la soluzione ottima in un numero finito di passi.

È importante sapere che esistono delle semplici varianti dell' algoritmo di Ford e Fulkerson che garantiscono la convergenza in un numero finito di passi qualunque siano i valori delle capacità.

7.6 Calcolo e scelta dei cammini aumentanti

In questa sezione affrontiamo la questione che era stata lasciata in sospeso nella descrizione dell'algoritmo di Ford e Fulkerson: come possiamo, in maniera sistematica, trovare un cammino aumentante o, nel caso

³Ricordiamo che i numeri razionali sono quelli esprimibili come rapporto di due numeri interi. Esempi di numeri irrazionali sono invece π e $\sqrt{2}$.

non ne esista uno, trovare un taglio minimo? A questo fine introduciamo un semplice algoritmo noto come *algoritmo di etichettatura*. L'algoritmo assegna ad ogni nodo un'etichetta a partire da s fino a raggiungere, se possibile, t . Se riesce effettivamente a raggiungere t è poi possibile ricostruire un cammino aumentante a partire dalle etichette; se invece non riesce a raggiungere t allora non esistono cammini aumentanti e l'insieme dei nodi etichettati e quello dei nodi non etichettati costituiscono un taglio di capacità minima.

Ogni etichetta è costituita da due numeri; la generica etichetta del nodo i verrà indicata con $(E_1(i), E_2(i))$, dove $E_1(i)$ è il primo numero dell'etichetta e $E_2(i)$ il secondo.

Sia allora \bar{x} un flusso $s-t$ ammissibile. L'algoritmo procede nel seguente modo.

Assegna alla sorgente l'etichetta $(0, \infty)$. Poi seguita ad etichettare i nodi, finché possibile, secondo le seguenti regole.

(a) Se

- i è un nodo etichettato,
- j è un nodo non etichettato,
- esiste l'arco (i, j) ,
- risulta $\bar{x}_{ij} < c_{ij}$,

assegna a j l'etichetta $(i, \min\{c_{ij} - \bar{x}_{ij}, E_2(i)\})$.

(b) Se

- i è un nodo etichettato,
- j è un nodo non etichettato,
- esiste l'arco (j, i) ,
- risulta $\bar{x}_{ji} > 0$,

assegna a j l'etichetta $(i, \min\{x_{ji}, E_2(i)\})$.

Commenti. A un certo stadio dell'algoritmo ci possono essere più modi in cui un nodo non etichettato può essere etichettato seguendo le regole (a) e (b). In questo caso basta scegliere uno qualsiasi dei modi possibili e proseguire. Per esempio consideriamo la parte di un grafo parzialmente etichettato, mostrata in figura 7.9a). Il nodo 9 può essere etichettato indifferentemente a partire dal nodo 5, 7.9b), o dal nodo 11, 7.9c).

È importante tener presente che una volta attribuita un'etichetta a un nodo questa non va più cambiata.

La procedura termina quando o viene etichettato il pozzo t o non è più possibile etichettare nessun nodo.

Significato delle etichette Le etichette indicano un possibile cammino aumentante dalla sorgente s al nodo etichettato. Per ricostruire il cammino basta usare le $E_1(i)$ e leggerle all'indietro fino ad arrivare alla sorgente s . È infatti facile convincersi, tenendo presente le regole (a) e (b) adottate nel processo di etichettatura, che il cammino dal nodo s al nodo i costruito in questo modo è in effetti un cammino aumentante e che il flusso aggiuntivo che è possibile inviare da s a i è esattamente pari a $E_2(i)$.

Quindi se etichettiamo il nodo t abbiamo trovato un cammino aumentante e possiamo procedere con l'algoritmo di Ford e Fulkerson.

Consideriamo invece il caso in cui l'algoritmo di etichettatura si ferma, senza aver etichettato il pozzo t , perché non è più possibile etichettare nessun nodo. Definiamo W come l'insieme dei nodi che sono stati etichettati e indichiamo con \bar{W} i rimanenti nodi del grafo. Risulta ovviamente $s \in W$, per costruzione, e $t \in \bar{W}$ perché abbiamo supposto che il pozzo non sia stato etichettato. (W, \bar{W}) costituisce quindi un taglio $s-t$.

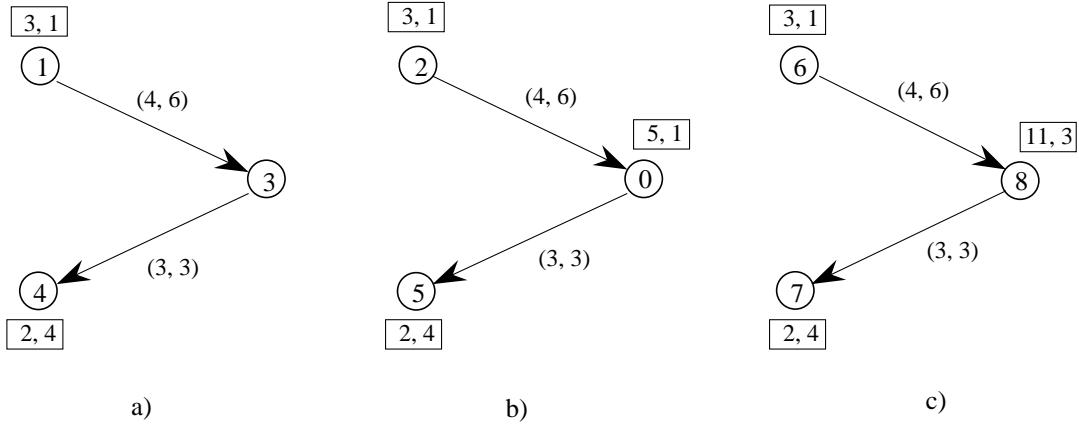


Figura 7.9: Esempio di etichettatura

Calcoliamo la capacità di questo taglio. Poiché l'algoritmo di etichettatura si è fermato senza essere in grado di etichettare nessun nodo in \bar{W} abbiamo, per le regole (a) e (b), che tutti gli archi (i, j) che attraversano il taglio sono o saturi ($x_{ij} = c_{ij}$), se $i \in W$ e $j \in \bar{W}$, o vuoti ($x_{ij} = 0$) se $i \in \bar{W}$ e $j \in W$. Ma allora, tenendo presente il Teorema 7.2.2 e le definizioni di capacità e flusso netto di un taglio, abbiamo

$$\bar{f} = F(W, \bar{W}) = C(W, \bar{W}).$$

Quindi (W, \bar{W}) è un taglio minimo e \bar{x} è ottimo.

Notiamo che l'algoritmo di etichettatura proposto trova, se ne esiste almeno uno, un cammino aumentante qualunque. Ovviamente il comportamento dell'algoritmo di Ford e Fulkerson sarà diverso se si scelgono cammini aumentanti diversi. In particolare la scelta del cammino aumentante da usare ad ogni iterazione può avere una influenza critica sul numero di iterazioni necessarie per raggiungere l'ottimo. A questo scopo può essere istruttivo considerare il semplice esempio riportato nella figura 7.10 dove i numeri sugli archi indicano le capacità. Applichiamo l'algoritmo di Ford e Fulkerson. Se scegliamo i cammini

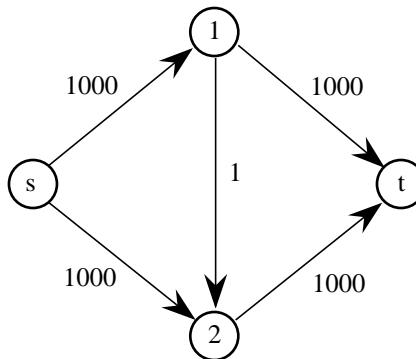


Figura 7.10: Importanza della scelta dei cammini aumentanti

aumentanti $((s, 1), (1, t))$ e $((s, 2), (1, t))$ si vede facilmente che l'algoritmo trova l'ottimo in due sole iterazioni. Se invece scegliamo come cammini aumentanti i cammini $((s, 1), (1, 2), (2, t))$, $((s, 2), (2, 1), (1, t))$,

$((s, 1), (1, 2), (2, t))$, $((s, 2), (2, 1), (1, t))$, ..., si vede che sono necessarie 2000 iterazioni, perché attraverso ogni cammino aumentato indicato si può aumentare il valore del flusso di una sola unità alla volta.

Esistono dei modi “intelligenti” di scegliere i cammini aumentanti che assicurano che situazione disastrose come quella evidenziata nell’esempio appena visto non possono accadere. Non abbiamo qui lo spazio per approfondire l’argomento. Diciamo solo che due “buone” scelte sono quelle di scegliere ad ogni passo, tra tutti i cammini aumentanti possibili, un cammino composto dal minore numero possibili di archi o un cammino che assicuri di poter aumentare il valore del flusso della più grande quantità possibile. Notiamo che usando uno qualunque di questi due criteri nell’ esempio della figura 7.10 porta alla scelta dei cammini $((s, 1), (1, t))$ e $((s, 2), (1, t))$ e fa quindi terminare l’algoritmo di Ford e Fulkerson in 2 sole iterazioni.

Esistono dei modi efficienti e esistematici, diversi dalla procedura di etichettatura illustrata in questa sezione, per calcolare cammini aumentanti che soddisfino uno dei due criteri appena indicati. Qui, per ragioni di spazio, diciamo solo che sono basati sulle tecniche di calcolo dei cammini minimi del tipo illustrato nel capitolo precedente.

7.7 Esempi

Dato il grafo di figura 7.11, trovare il valore del massimo flusso inviabile da s a t (la capacità associata a ciascun arco è mostrata in figura in prossimità dell’arco stesso).

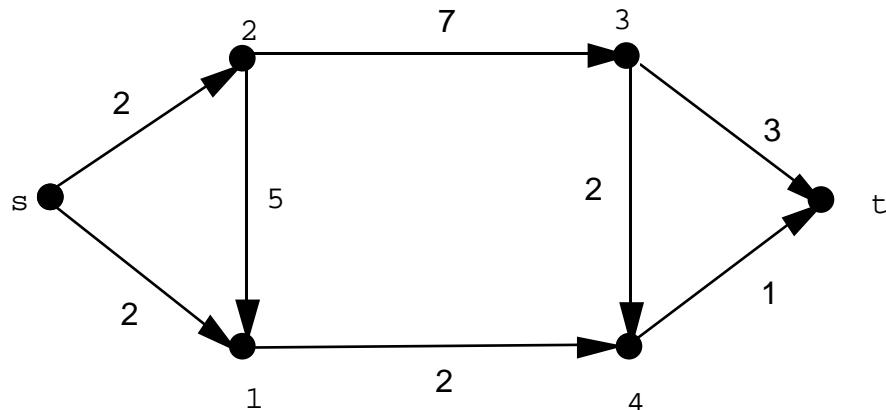


Figura 7.11:

Soluzione.

Per calcolare il flusso utilizzeremo l’algoritmo di Ford e Fulkerson, con flusso iniziale nullo. Ricordiamo infatti che l’algoritmo di Ford e Fulkerson richiede in ingresso un flusso ammissibile, e che il flusso nullo è un flusso ammissibile per il nostro problema.

- *Iterazione 0.*

Al passo 0 avremo $x_{s1}^0 = 0, x_{s2}^0 = 0, x_{14}^0 = 0, x_{21}^0 = 0, x_{23}^0 = 0, x_{34}^0 = 0, x_{3t}^0 = 0, x_{4t}^0 = 0$. Un possibile cammino aumentante da s a t sarà: $P = \{s, (s, 2), 2, (2, 3), 3, (3, 4), 4, (4, t), t\}$. $P_+ = \{(s, 2), (2, 3), (3, 4), (4, t)\}$, mentre $P_- = \emptyset$. Ci sono solo archi diretti e dunque $\delta = \delta_+ = \min(2-0, 7-0, 2-0, 1-0) = 1$. Quindi tutte le variabili relative agli archi del cammino aumentante vengono aumentate di un'unità. Le altre resteranno invariate. Si avrà: $x_{s1}^1 = 0, x_{s2}^1 = 1, x_{14}^1 = 0, x_{21}^1 = 0, x_{23}^1 = 1, x_{34}^1 = 1, x_{3t}^1 = 0, x_{4t}^1 = 1$. Il flusso è aumentato da 0 a 1.

- *Iterazione 1.*

Un nuovo cammino aumentante sarà: $P = \{s, (s, 1), 1, (1, 4), 4, (3, 4), 3, (3, t), t\}$.

Risulta $P_+ = \{(s, 1), (1, 4), (3, t)\}$, mentre $P_- = \{(3, 4)\}$. Inoltre $\delta_+ = \min(2-0, 2-0, 3-0) = 2$, mentre $\delta_- = \min(1) = 1$. Quindi $\delta = \min(\delta_+, \delta_-) = 1$.

Le variabili corrispondenti ad archi diretti vengono quindi aumentate di un'unità, mentre le variabili corrispondenti ad archi inversi (in questo caso la variabile x_{34}) vengono diminuite di un'unità. Sarà dunque: $x_{s1}^2 = 1, x_{s2}^2 = 1, x_{14}^2 = 1, x_{21}^2 = 0, x_{23}^2 = 1, x_{34}^2 = 0, x_{3t}^2 = 1, x_{4t}^2 = 1$. Il flusso è aumentato da 1 a 2.

- *Iterazione 2.*

Infine troviamo il cammino aumentante: $P = \{s, (s, 2), 2, (2, 3), 3, (3, t), t\}$. Sono tutti archi diretti e per essi vale $\delta = 1$. Quindi si avrà:

$$x_{s1}^3 = 1, x_{s2}^3 = 2, x_{14}^3 = 1, x_{21}^3 = 0, x_{23}^3 = 2, x_{34}^3 = 0, x_{3t}^3 = 2, x_{4t}^3 = 1.$$

Il flusso finale vale 3.

- *Iterazione 3.*

Non è possibile trovare un cammino aumentante rispetto al flusso x^3 , che è dunque ottimo. \square

Ovviamente, se è noto un flusso iniziale ammissibile, diverso da 0, possiamo tranquillamente applicare l'algoritmo a partire dal flusso dato. Consideriamo anche un'altro esempio.

Dato il grafo e il flusso ammissibile rappresentati in Fig. 7.12, trovare il flusso massimo inivibile da s a t , certificando l'ottimalità della soluzione ottenuta con l'esibizione di un taglio di capacità uguale al flusso massimo trovato.

Soluzione.

Per trovare il flusso massimo utilizzeremo l'algoritmo di Ford e Fulkerson, con flusso iniziale uguale a quello in figura. Il flusso iniziale vale 10.

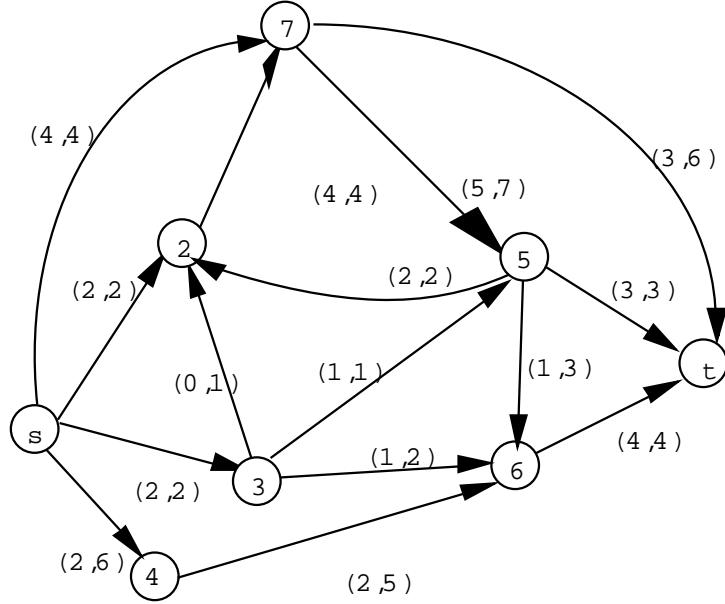


Figura 7.12:

- *Iterazione 0.*

Al passo 0 avremo $x_{s2}^0 = 2$, $x_{s3}^0 = 2$, $x_{s4}^0 = 2$, $x_{s7}^0 = 4$, $x_{27}^0 = 4$, $x_{32}^0 = 0$, $x_{35}^0 = 1$, $x_{36}^0 = 1$, $x_{46}^0 = 2$, $x_{52}^0 = 2$, $x_{56}^0 = 1$, $x_{5t}^0 = 3$, $x_{6t}^0 = 4$, $x_{75}^0 = 5$, $x_{7t}^0 = 3$. Un possibile cammino aumentante da s a t sarà: $P = \{s, (s, 4), 4, (4, 6), 6, (5, 6), 5, (7, 5), 7, (7, t), t\}$. $P_+ = \{(s, 4), (4, 6), (7, t)\}$, mentre $P_- = \{(5, 6), (7, 5)\}$. $\delta_+ = \min(4, 3, 3) = 3$, mentre $\delta_- = \min(1, 3) = 1$. Quindi $\delta = \min(\delta_+, \delta_-) = 1$. Si avrà quindi (scriviamo solo le variabili da aggiornare): $x_{s4}^1 = 3$, $x_{46}^1 = 3$, $x_{56}^1 = 0$, $x_{75}^1 = 4$, $x_{7t}^1 = 4$. Il flusso è aumentato da 10 a 11.

- *Iterazione 1.*

Un nuovo cammino aumentante sarà: $P = \{s, (s, 4), 4, (4, 6), 6, (3, 6), 3, (3, 2), 2, (5, 2), 5, (7, 5), 7, (7, t), t\}$. $P_+ = \{(s, 4), (4, 6), (3, 2), (7, t)\}$, mentre $P_- = \{(3, 6), (5, 2), (7, 5)\}$. $\delta_+ = \min(3, 2, 1, 2) = 1$, mentre $\delta_- = \min(1, 2, 2) = 1$. Quindi $\delta = \min(\delta_+, \delta_-) = 1$. Si avrà: $x_{s4}^2 = 4$, $x_{46}^2 = 4$, $x_{36}^2 = 0$, $x_{32}^2 = 1$, $x_{52}^2 = 1$, $x_{75}^2 = 5$, $x_{7t}^2 = 5$. Il flusso è aumentato da 11 a 12.

- *Iterazione 2.*

Non è possibile trovare un cammino aumentante rispetto a x^2 , che è dunque ottimo.

Un modo per rappresentare graficamente le iterazioni dell'algoritmo è illustrato in Figura 7.13. Ogni volta che il valore del flusso su un arco viene aggiornato, si sbarra il precedente valore del flusso sovrapponendogli il nuovo valore.

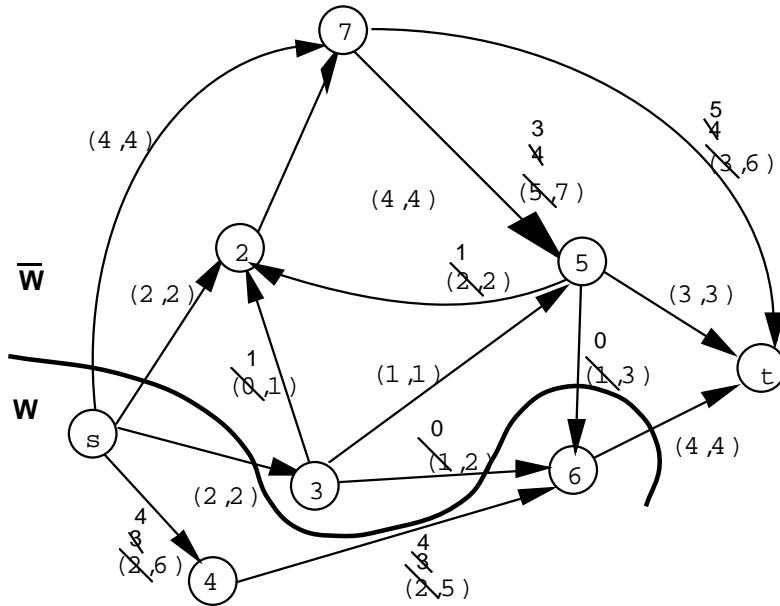


Figura 7.13:

Nell'esempio un taglio a capacità minima sarà $W = \{s, 4, 6\}$ e $\bar{W} = \{2, 3, 5, 7, t\}$, ed è mostrato in figura 7.13 con una linea che separa i due insiemi. Per calcolare la capacità del taglio bisogna sommare le capacità degli archi che hanno la coda in W e la testa in \bar{W} , e cioè gli archi $(s, 2)$, $(s, 3)$, $(s, 7)$, e $(6, t)$, e la somma delle loro capacità vale 12. Per trovare questo taglio si può usare la costruzione suggerita dall'enunciato del Teorema ??, ovvero basta mettere in W il nodo s e tutti i nodi raggiungibili da s con un cammino aumentante rispetto alla soluzione ottima x^2 . In questo esempio, i nodi raggiungibili sono appunto il nodo 4 (cammino aumentante $\{s, (s, 4), 4\}$) e il nodo 6 (cammino aumentante $\{s, (s, 4), 4, (4, 6), 6\}$).

7.8 Accoppiamento bipartito

Sia dato un grafo bipartito $G(S, D, E)$ (con $|S| = l, |D| = r, l+r = n, |E| = m$) non orientato. Il problema che si vuole affrontare è quello di determinare un sottoinsieme A di archi di G di cardinalità massima, tale che due archi di A non abbiano mai un nodo in comune. In altre parole, si tratta di assegnare il massimo numero di nodi dell'insieme S a nodi dell'insieme D (o viceversa), in modo che ogni nodo sia di S sia di D sia assegnato al più ad un solo altro nodo, con il vincolo che una assegnazione di un nodo i a un nodo j possa avvenire solo se nel grafo G esiste l'arco (i, j) .

Questo problema, noto come problema di accoppiamento di cardinalità massima, ha moltissime applicazioni, e si presenta tutte le volte in cui bisogna assegnare in maniera esclusiva i membri di un insieme

(nodi S) a quelli di un altro insieme (nodi D). Per esempio, supponiamo che i nodi S rappresentino i dipendenti di un'azienda e i nodi D i lavori che devono essere svolti. Supponiamo inoltre che un arco collega un nodo di S a un nodo di D se e solo se la persona rappresentata dal nodo di S è in grado di svolgere il lavoro rappresentato dal nodo di D . In questo caso allora, risolvere il problema di accoppiamento di cardinalità massima vuol dire assegnare ai dipendenti i lavori in modo da massimizzare il numero di lavori svolti.

In questa sezione presentiamo un algoritmo per la risoluzione del problema di accoppiamento di cardinalità massima basato sulla riduzione del problema a un problema di massimo flusso, che può quindi essere risolto con l'algoritmo di Ford e Fulkerson. Più precisamente mostreremo come sia possibile formulare il problema come problema di massimo flusso; ossia mostreremo come sia possibile formulare un problema di massimo flusso la cui soluzione ottima fornisca immediatamente la soluzione ottima del problema di accoppiamento bipartito.

Per raggiungere questo scopo, costruiamo il seguente grafo “stratificato” H con $(n+2)$ nodi, $(m+n)$ archi e 4 “strati” (vedi la figura 7.14).

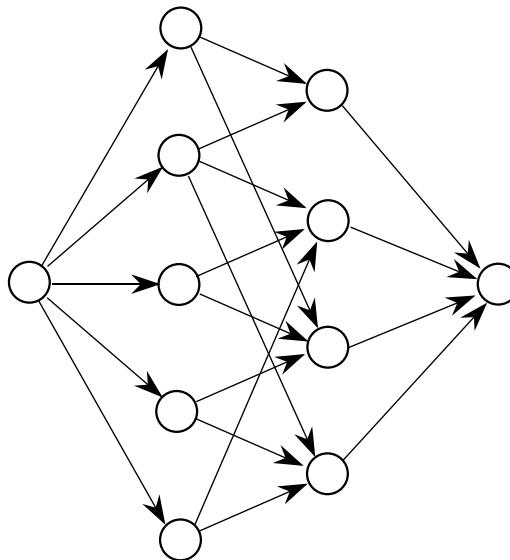


Figura 7.14: Grafo H

- Il primo strato è formato da un solo nodo (detto sorgente e indicato con s).
- Il secondo strato è formato da l nodi (corrispondenti ai nodi dell'insieme S).
- Il terzo strato è formato da r nodi (corrispondenti ai nodi dell'insieme D).
- Il quarto strato è formato da un solo nodo (detto pozzo e indicato con p).

Il nodo s è collegato a tutti i nodi del secondo strato con archi, di capacità pari a 1, orientati da s verso gli altri nodi. I nodi del secondo strato sono collegati ai nodi del terzo strato con archi, di capacità pari a 1, orientati dai nodi del secondo strato verso i nodi del terzo. Questi archi corrispondono agli archi dell'insieme E (ossia, sia i un nodo del secondo strato e dell'insieme S , j un nodo del terzo strato e dell'insieme D , esiste un arco nel grafo H tra i e j se e solo se (i, j) appartiene all'insieme E).

Tutti i nodi del terzo strato sono collegati con il nodo p con archi, di capacità pari a 1, orientati dai nodi del terzo strato verso p .

Si noti che s ha solo archi uscenti, i nodi del secondo strato hanno tutti un solo arco entrante, i nodi del terzo strato hanno tutti un solo arco uscente, p ha solo archi entranti.

Risolvendo il problema del massimo flusso dal nodo s al nodo p sul grafo H , viene anche risolto il problema di accoppiamento massimo sul grafo G .

In effetti, per la particolare struttura del grafo, è facile convincesi che una distribuzione di flusso che massimizza il flusso inviato da s a p inviando complessivamente un flusso pari a f^* , (e ottenuta con l'algoritmo di Ford e Fulkerson) è formata da un insieme di f^* cammini in cui scorre un flusso unitario, tutti con origine nel nodo s , arrivo nel nodo p , formati da tre archi (uno tra gli strati 1 e 2, uno tra gli strati 2 e 3, uno tra gli strati 3 e 4) e senza altri nodi comuni oltre a s e p . Accoppiando i nodi associati agli archi con flusso pari a 1, si ottiene un accoppiamento ammissibile di cardinalità pari a f^* . D'altra parte, se, per assurdo, esistesse un accoppiamento di cardinalità maggiore di f^* , allora sarebbe anche possibile, per la struttura del problema, costruire una distribuzione di flussi maggiore di f^* (e pari alla cardinalità dell'accoppiamento). Infatti basterebbe inviare un flusso pari a 1: da s a tutti i nodi del secondo strato accoppiati, fra i nodi accoppiati, e dai nodi del terzo strato accoppiati al nodo p ; questo è ovviamente assurdo. Quindi l'accoppiamento di cardinalità f^* , trovato in base alla risoluzione del problema di massimo flusso, è massimo e risolve il problema di accoppiamento bipartito posto all'inizio.

7.9 Il problema di distribuzione di flusso a costo minimo

In questo ultimo pragrafo vogliamo mostrare come tutti i problemi su grafi considerati fin qui (cammini minimi, flusso massimo, accoppiamento bipartito), siano in verità casi particolari di un unico problema: Il problema della distribuzione di flusso a costo minimo.

Descriviamo dapprima questo problema. Problemi di distribuzione di flusso a costo minimo nascono naturalmente quando si voglia determinare il modo più economico di trasportare merci da alcune origini a delle destinazioni attraverso una *rete fisica*, e in effetti modelli di questo tipo sono stati già incontrati quando abbiamo presentato modelli di PL. Consideriamo una grafo orientato $G = (V, E)$ con associato ad ogni arco $(i, j) \in E$ un costo c_{ij} (negativo, positivo o nullo), una capacità $u_{ij} \geq 0$ ed un limite inferiore l_{ij} , con $0 \leq l_{ij} \leq u_{ij}$. Supponiamo inoltre che a ciascun nodo i di V sia associato un numero intero $b(i)$, con la convenzione che:

- $b(i) > 0$ indica la presenza di offerta (il nodo i viene detto *origine*);
- $b(i) < 0$ indica la presenza di domanda (il nodo i viene detto *destinazione*);
- $b(i) = 0$ indica l'assenza di offerta e di domanda (il nodo i viene detto *di trasferimento*).

Nel seguito supporremo valida la seguente ipotesi:

Ipotesi di ammissibilità. L'offerta totale uguaglia la domanda totale; ovvero, formalmente:

$$\sum_{i \in V} b(i) = 0. \quad (7.15)$$

Definizione 7.9.1 Un flusso ammissibile in una rete G è un vettore non negativo $x \in \mathbb{R}^{|E|}$ che soddisfa i seguenti vincoli:

- vincoli di capacità: per ogni arco $(ij) \in E$:

$$l_{ij} \leq x_{ij} \leq u_{ij}.$$

- vincoli di bilanciamento della massa: per ogni nodo $k \in V$

$$\sum_{(jk) \in \omega^-(k)} x_{jk} - \sum_{(ki) \in \omega^+(k)} x_{ki} = b(k).$$

Il *costo di un flusso* è definito come la somma

$$\sum_{(ij) \in E} c_{ij} x_{ij}.$$

Il problema del flusso a costo minimo, si può enunciare come segue:

trovare una distribuzione ammissibile di flusso sulla rete G tale in modo da minimizzare il costo totale.

Se si indica con A_G la matrice d'incidenza della rete, con $b = (b(1), \dots, b(n))^T$ il vettore della domanda/offerta, con $c \in \mathbb{R}^m$ il vettore dei costi e con $l \in \mathbb{R}_+^m$ ed $u \in \mathbb{R}_+^m$ i vettori le cui componenti rappresentano il limite inferiore e la capacità dell'arco corrispondente, il problema di flusso a costo minimo può essere scritto nel seguente modo:

$$\begin{aligned} & \min c^T x \\ & A_G x = b \\ & l \leq x \leq u \end{aligned} \tag{P}_C$$

Nel caso di rete non capacitata, cioè quando si abbia $l_{ij} = 0$ e $u_{ij} = \infty$, allora il problema di flusso a costo minimo si scrive:

$$\begin{aligned} & \min c^T x \\ & A_G x = b \\ & x \geq 0 \end{aligned} \tag{Q}.$$

In questo caso il problema è anche noto in letteratura come *problema di trasferimento* (transhipment problem).

L'ipotesi di ammissibilità consente di assicurare l'esistenza di una soluzione ammissibile per il problema di trasferimento (\tilde{P}) . Si ha infatti, il seguente teorema:

Teorema 7.9.2 (Ipotesi di ammissibilità) *Condizione necessaria e sufficiente per l'esistenza di una soluzione ammissibile per problema di trasferimento (Q) è che valga (7.15).*

Una delle proprietà più interessanti della classe di problemi che stiamo analizzando è espressa dal seguente teorema.

Teorema 7.9.3 (Proprietà di interezza) *Dato un problema di flusso a costo minimo che soddisfa l'ipotesi di ammissibilità, se i vettori l , u e b sono interi non negativi, se il problema ha una soluzione ottima, allora ha una soluzione intera dello stesso valore.*

In particolare se si utilizasse il metodo del simplex (cosa che in genere, bisogna dire, non viene fatta, poiché si possono sviluppare sue specializzazioni che risultano più efficienti), si può mostrare che la soluzione ottima trovata, se sono soddisfatte le condizioni del teorema precedente, sono intere.

RiconSIDERIAMO ora i problemi su grafi già studiati e mostriAMO che si tratta di casi particolari del problema di flusso a costo minimo.

Il problema del massimo flusso

Un problema di massimo flusso può essere ricondotto a un problema di flusso a costo minimo nel seguente modo:

- si aggiunga un arco dal nodo t al nodo s di limite inferiore $l_{ts} = 0$, capacità u_{ts} infinita e costo $c_{ts} = -1$;
- per tutti gli altri archi $ij \in E$, $(ij \neq ts)$, si ponga $c_{ij} = 0$, $l_{ij} = 0$, lasciando la capacità invariata;

- per ogni nodo $i \in V$, si ponga $b(i) = 0$ (vincoli di conservazione del flusso).

È facile vedere che minimizzare il costo di un flusso su una rete così definita, corrisponde a massimizzare la quantità di flusso sull'arco fittizio ts , e quindi a massimizzare la quantità di flusso entrante in t .

Il problema del cammino minimo

Per capire come un problema di cammino minimo possa essere visto, in effetti, come un problema di flusso a costo minimo, dobbiamo prima considerare una riformulazione del problema del cammino minimo come problema di PLI. A tal fine ricordiamo dapprima la definizione del problema.

Dato un grafo orientato $G = (N, A)$, associamo a ciascun arco $e = (u, v) \in A$ un numero reale p_e , detto *peso* dell'arco. Per ogni cammino *orientato* $C = \{v_1, e_1, \dots, e_p, v_p\}$, denotiamo con $p(C)$ il *peso* di C , ossia la somma dei pesi degli archi di PC . Il problema del *cammino minimo* è:

dati due nodi $s \in V$ e $t \in V$, trovare un cammino orientato P^ in G da s a t di peso minimo.*

Si osservi che (i) se non esiste un cammino orientato da s a t in G , il problema non ha soluzioni ammissibili; (ii) se esiste un ciclo orientato in G di peso negativo la soluzione del problema è illimitata (poiché conterrà tale ciclo ripetuto un numero illimitato di volte).

Una formulazione del problema che utilizza variabili binarie ed è descritta di seguito.

– *Variabili*. Si definiscono le variabili booleane

$$x_{ij} = \begin{cases} 1 & \text{se } (i, j) \in P \\ 0 & \text{altrimenti} \end{cases}$$

Il vettore $x \in R^m$ si chiama vettore di incidenza del cammino P .

– *Funzione obiettivo*. La funzione obiettivo è il peso del cammino:

$$\sum_{(i,j) \in E} c_{ij} x_{ij}.$$

– *Vincoli*. I vincoli devono imporre che la soluzione corrisponda al vettore d'incidenza di un cammino semplice da s a t , cioè:

- in ciascun nodo k che non sia s o t incidono esattamente due archi del cammino, uno entrante e l'altro uscente. Questo si esprime con il vincolo:

$$\begin{aligned} \sum_{(ik) \in \omega^-(k)} x_{ik} &= 1 & \forall k \in V - \{s, t\} \\ \sum_{(kj) \in \omega^+(k)} x_{kj} &= 1 & \forall k \in V - \{s, t\} \end{aligned}$$

- dalla sorgente s esce esattamente un arco e nel pozzo t entra esattamente un arco; cioè:

$$\begin{aligned} \sum_{(it) \in \omega^-(t)} x_{it} &= 1 \\ \sum_{(sj) \in \omega^+(s)} x_{sj} &= 1 \end{aligned}$$

Se poniamo $b = (1 \ -1 \ 0 \ \dots \ 0)^T$, e consideramo la matrice di incidenza A della rete G , allora questi vincoli si possono esprimere in forma matriciale come:

$$Ax = b.$$

Se si eccettua la richiesta che le variabili siano binarie, si tratta ovviamente di un problema di flusso a costa minimo in cui

- la rete è non-capacitata;
- per ogni nodo $i \in V - \{s, t\}$, si pone $b(i) = 0$;
- si pone $b(s) = 1$ e $b(t) = -1$.

È facile vedere che ogni flusso ammissibile corrisponde a inviare un flusso unitario dal nodo s al nodo t ; se si suppone che il flusso ottimo sia intero, un flusso unitario corrisponde a un cammino diretto da s a t (formato dagli archi il cui flusso corrispondente è uguale a uno). Dunque, il flusso di costo minimo corrisponderà al cammino di costo minimo da s a t . Per il Teorema 7.9.3 sappiamo che in effetti, almeno una soluzione di questo problema problema è intera. Quindi se utilizziamo un algoritmo che ci garantisce di trovare una soluzione intera (per esempio il simplex), nella formulazione precedente possiamo trascurare il vincolo (difficile da trattare) che impone che le variabili siano 0-1 e ridurci a un puro problema di flusso a costo minimo.

I problemi di flusso a costo minimo costituiscono una delle più importanti classi di problemi su grafi, e anche se non possiamo approfondire i metodi di soluzione per questa classe di problemi, abbiamo voluto almeno introdurre il problema e alcune sue importanti proprietà, quali l'esistenza di soluzioni intere.

L'aver fatto vedere che il problema dei cammini minimi e quello del massimo flusso sono casi particolari del problema di flusso a costo minimo non vuole assolutamente suggerire che i metodi di risoluzione per quest'ultima classe di problemi dovrebbero essere usati per la risoluzione di problemi di cammino minimo o di massimo flusso. È piuttosto vero il contrario: gli algoritmi che abbiamo studiato per risolvere i problemi di massimo flusso e di cammino minimo contengono in nuce le idee di base e, in alcuni casi, possono essere direttamente utilizzati come "mattoncini" per sviluppare algoritmi efficienti per la risoluzione dei problemi di flusso a costo minimo.

Capitolo 8

Euristiche per la soluzione di problemi di ottimizzazione combinatoria

Il termine *Euristica*, dal greco *euristikein* = scoprire, indica un metodo (algoritmico) per la ricerca di soluzioni ammissibili (non necessariamente ottime) di un problema di ottimizzazione. Se i vincoli o la funzione obiettivo del problema hanno una qualche particolare struttura si è talvolta in grado di sviluppare algoritmi efficienti che trovano la soluzione ottima in tempi ragionevoli anche per problemi di grande dimensione. Esempi di siffatti problemi sono il problema del cammino minimo in un grafo e il problema del massimo flusso. In molti casi, tuttavia, non sono noti metodi efficienti, e ci si deve accontentare di applicare metodi generali quali ad esempio il branch-and-bound. Purtroppo, quando la dimensione del problema cresce, il metodo del branch-and-bound potrebbe richiedere la generazione di un numero troppo elevato di sottoproblemi, e quindi non convergere in tempo ragionevole.

Per problemi di questo tipo, all'aumentare delle dimensioni ci si deve spesso accontentare di metodi che non assicurano l'ottimalità delle soluzioni trovate - le euristiche, appunto - ma che producono soluzioni di "qualità" ragionevole. In generale, se Z^* è il valore della soluzione ottima del problema dato (supposto di minimizzazione, per fissare le idee) e Z^E è il valore della soluzione trovata dalla nostra euristica, sarà $Z^E \geq Z^*$. Il punto cruciale è che la soluzione euristica *potrebbe* essere ottima (ovvero $Z^E = Z^*$), ma noi non siamo in grado di dimostrarne l'ottimalità. Questa è la differenza sostanziale fra metodi euristicci e metodi esatti (come il branch-and-bound): se il metodo esatto ha il tempo sufficiente a terminare le sue iterazioni, la soluzione finale è una soluzione ottima.

Alcuni tipi di euristiche forniscono un risultato *garantito* nel senso che è possibile dimostrare che la soluzione prodotta ha una valore che è al più α volte la soluzione ottima (con $\alpha > 1$), ovvero $Z^E \leq \alpha Z^*$.

Più spesso il risultato delle euristiche non è garantito, e la soluzione prodotta potrebbe avere un valore molto maggiore della soluzione ottima. Per poter valutare la qualità di un'euristica per un problema dato, i ricercatori hanno messo a punto un metodo detto di *benchmarking*. Si tratta semplicemente di utilizzare delle librerie di istanze del problema considerato per le quali sono noti i risultati ottenuti dalle altre euristiche proposte nella letteratura, e verificare come si posizionano rispetto ad essi i risultati prodotti dalla nuova euristica.

Di seguito ci occuperemo di euristiche per una classe particolare di problemi di ottimizzazione, introdotta nel prossimo paragrafo, e che comprende numerosissimi esempi di problemi reali.

8.1 L'ottimizzazione combinatoria

Una particolare classe di problemi di ottimizzazione è rappresentata dai cosiddetti *problem di ottimizzazione combinatoria*. A questa classe appartengono, fra l'altro, i problemi di PL01, il problema dell'accoppiamento e il problema del cammino minimo su grafi visti nei capitoli precedenti.

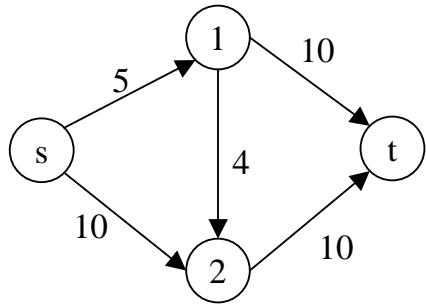


Figura 8.1: Esempio di cammino minimo

Un generico problema di ottimizzazione combinatoria può essere definito come segue. Sia $B = \{b_1, \dots, b_n\}$ un qualsiasi insieme finito detto *insieme base* ("ground set") e sia $\mathcal{S} = \{S_1, \dots, S_m\}$ una famiglia di sottoinsiemi di B ("Subset System"). Infine, sia $w : \mathcal{S} \rightarrow \mathbb{R}$ una funzione obiettivo che associa a ciascun insieme $S \in \mathcal{S}$ un numero reale $w(S)$. Allora, diremo *Problema di Ottimizzazione Combinatoria* il seguente problema:

$$\min_{S \in \mathcal{S}} w(S)$$

Si osservi la differenza fra la definizione di problema di ottimizzazione combinatoria e la più generale definizione di problema di ottimizzazione introdotta nel Capitolo 1. Nella definizione generale, l'insieme di soluzioni ammissibili \mathcal{S} può assumere qualunque forma, mentre nei problemi di ottimizzazione \mathcal{S} è una famiglia (finita) di sottoinsiemi di un insieme finito di elementi.

Ad esempio, il problema del cammino minimo da s a t in un grafo orientato $G = (V, E)$, con un peso p_e associato a ogni arco $e \in E$, visto nel capitolo precedente, è un problema di ottimizzazione combinatoria. Per questo problema, l'insieme di base è l'insieme degli archi. Per l'istanza mostrata in figura 8.1, l'insieme di base sarà $B = \{(s, 1), (s, 2), (1, 2), (1, t), (2, t)\}$. La famiglia di soluzioni è l'insieme di tutti i sottoinsiemi di archi che corrispondono a cammini orientati da s a t . In particolare, avremo 3 possibili cammini distinti da s a t e quindi $\mathcal{S} = \{S_1 = \{(s, 1), (1, t)\}, S_2 = \{(s, 1), (1, 2), (2, t)\}, S_3 = \{(s, 2), (2, t)\}\}$. La funzione obiettivo è semplicemente la somma dei pesi associati agli archi: $w(S) = \sum_{e \in S} p_e$. Quindi $w(S_1) = 15$, $w(S_2) = 19$ e $w(S_3) = 20$.

Nella maggioranza dei casi trattati, a ogni elemento dell'insieme di base è associato un peso e la funzione obiettivo è semplicemente la somma dei pesi degli elementi contenuti nella soluzione. Tuttavia la funzione obiettivo può assumere in generale qualunque forma. Un esempio di diversa funzione obiettivo verrà mostrato in seguito.

Nei prossimi paragrafi illustreremo alcuni esempi importanti di problemi di ottimizzazione combinatoria.

8.1.1 Il problema dell'accoppiamento massimo su grafi bipartiti

Sia dato un grafo $G = (V, E)$ non orientato e bipartito, ovvero $i) V = U \cup W$ e $W \cap U = \emptyset$ e ogni arco appartenente ad E ha per estremi un nodo $u \in U$ e un nodo $v \in W$ (cioè non esistono archi con ambo gli estremi in U o ambo gli estremi in W). Inoltre, con ogni arco $(u, v) \in E$ è associato un peso w_{uv} , che rappresenta il "vantaggio" di assegnare il nodo u al nodo v . Un esempio di grafo bipartito (pesato) è mostrato in figura 8.2.a. Ora, a ogni nodo dell'insieme W vogliamo assegnare al più un nodo dell'insieme U e viceversa, in modo da massimizzare la somma dei vantaggi. Si osservi che assegnare un nodo $u \in U$ a un nodo $v \in W$ è equivalente a selezionare l'arco (u, v) (di peso w_{uv}). Inoltre, se è stato selezionato l'arco (u, v) , e quindi se è stato assegnato il nodo u a v (e viceversa), nessun altro arco incidente in u può essere selezionato (perchè ciò equivalebbe ad assegnare u a più di un nodo); analogamente, nessun altro arco incidente in v può essere selezionato. Un sottoinsieme $M \subseteq E$ degli archi di G che gode della proprietà che, comunque presi due archi appartenenti a M , essi non hanno estremi in comune, è detto *accoppiamento* (in inglese "matching"). Un esempio di matching (di valore 19) è mostrato in figura 8.2.b.

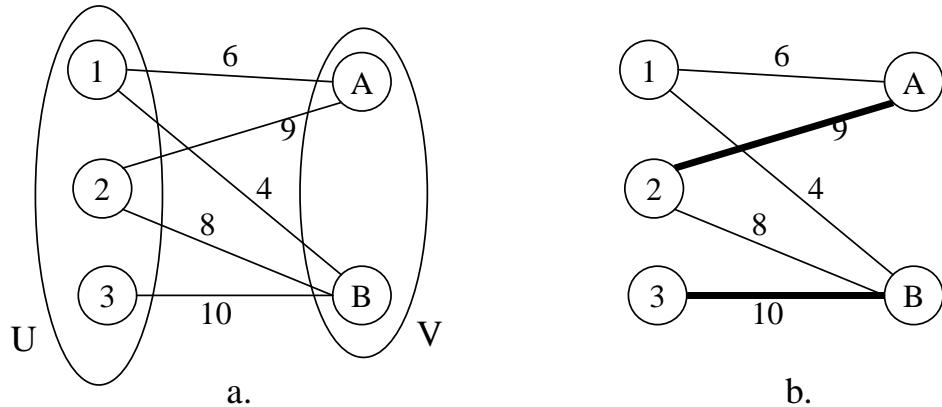


Figura 8.2: Esempio di accoppiamento bipartito

Dunque, il problema dell'accoppiamento massimo può essere così ridefinito: selezionare un sottoinsieme M degli archi di G tale che i) M sia un matching, e ii) la somma dei pesi degli archi in M sia massima.

Così formulato il problema dell'accoppiamento massimo su grafo bipartito è chiaramente un problema di ottimizzazione combinatoria, ove:

- l'insieme di base coincide con l'insieme degli archi E .
- la famiglia di sottoinsiemi di archi coincide con la famiglia di tutti i possibili matching di G
- la funzione obiettivo associa a ogni matching M la somma dei pesi degli archi in M .

8.1.2 Il Problema del Commesso Viaggiatore.

Un commesso viaggiatore deve visitare un certo numero di città, partendo dalla sua città di residenza e ritornandoci alla fine del giro. Questo problema può essere utilmente rappresentato su un grafo $G = (V, E)$, ove ogni nodo $u \in V$ rappresenta una città da visitare, mentre gli archi rappresentano strade (o voli) che collegano coppie di città. Per semplificare la trattazione supponiamo che il grafo G sia non orientato: l'estensione al caso orientato di quanto verrà detto è immediata. A ogni arco $(u, v) \in E$ si associa una distanza $d_{uv} \in \mathbb{R}_+$ che rappresenta la lunghezza del percorso fra la città u e la città v . Si osservi che d_{uv} potrebbe anche rappresentare il tempo di percorrenza, o il costo del trasferimento.

Dunque, il commesso viaggiatore, partendo dalla città in cui abita (sia per esempio la città 1), vuole effettuare tutte le visite e ritornare alla fine alla città di partenza, percorrendo una strada di lunghezza complessiva minima. Si osservi intanto che il "giro" del commesso viaggiatore corrisponde a un ciclo del grafo G che attraversi ogni nodo una e una sola volta. Un ciclo siffatto è detto *ciclo hamiltoniano*. Quindi, il *Problema del Commesso Viaggiatore* (indicato anche con l'acronimo TSP, dall'inglese *Travelling Salesman Problem*) si definisce come il problema di selezionare un particolare ciclo hamiltoniano, quello (o uno di quelli) di costo (lunghezza, peso) minimo.

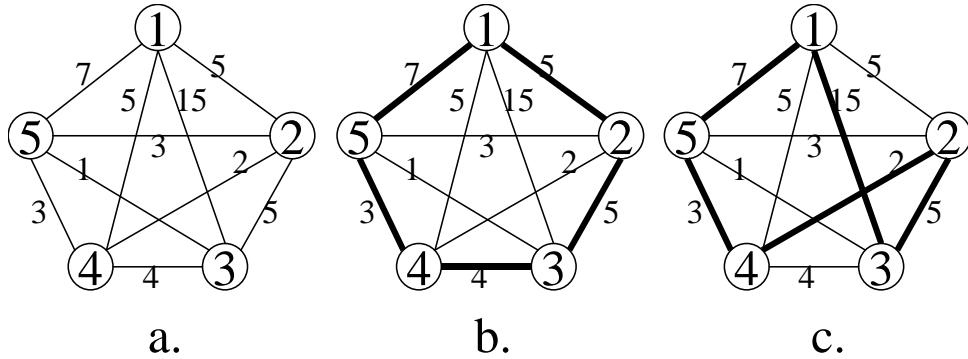


Figura 8.3: Esempi di cicli hamiltoniani

Si consideri ad esempio il grafo in Fig. 8.3. Un possibile ciclo hamiltoniano è, ad esempio, l'insieme di archi $H_1 = \{(1, 2), (2, 3), (3, 4), (4, 5), (5, 1)\}$ (Fig. 8.3.b), e il suo costo (somma dei costi degli archi) è pari a 24. Un altro ciclo hamiltoniano è $H_2 = \{(1, 3), (3, 2), (2, 4), (4, 5), (5, 1)\}$ (Fig. 8.3.c) di costo pari a 32.

Un ciclo hamiltoniano può dunque essere rappresentato come insieme di archi. Quindi, l'insieme di base del nostro problema di ottimizzazione combinatoria è l'insieme E degli archi di G . L'insieme delle soluzioni è una famiglia $\mathcal{H} = \{H_1, \dots, H_m\}$ di sottoinsiemi di E , ove H_i è l'insieme di archi di un ciclo hamiltoniano per $i = 1, \dots, m$. La funzione obiettivo è semplicemente, per ogni $H \in \mathcal{H}$, $w(H) = \sum_{(u,v) \in H} d_{uv}$.

8.1.3 Il problema del partizionamento di un grafo.

Problemi di questo tipo appaiono ogni volta si debba scegliere una partizione di un insieme di elementi in un certo numero di classi in modo da minimizzare (o massimizzare) una funzione di distanza fra gli elementi appartenenti a una stessa classe. Sia quindi dato un grafo non orientato $G = (V, E)$, e sia associato un peso w_{uv} con ogni arco $uv \in E$. Si vuole trovare la partizione di V in k classi (o *cluster*), con k prefissato, che minimizzi la somma dei pesi degli archi incidenti in nodi appartenenti a una stessa classe. Si ricordi che se $\pi = \{C_1, C_2, \dots, C_k\}$ è una k partizione di V si ha $\bigcup_{i=1}^k C_i = V$ e $C_i \cap C_j = \emptyset$ per $i, j \in \{1, \dots, k\}, i \neq j$. Il costo $w(C_i)$ di una classe di nodi C è definito come $\sum_{u \in C, v \in C} w_{uv}$. Il costo di una partizione $w(\pi) = w(\{C_1, C_2, \dots, C_k\})$ è definito come $\sum_{i=1}^k w(C_i)$. Il problema di Partizionamento di un Grafo consiste nel scegliere la k -partizione π^* che minimizza $w(\pi^*)$.

Anche il problema di partizionamento di un grafo può essere inserito nel *framework* dei problemi di ottimizzazione, anche se l'associazione non è così immediata. Esistono in realtà (almeno) due diverse rappresentazioni del problema come problema di ottimizzazione combinatoria. La prima consiste nel definire l'insieme di base come l'insieme degli archi E e le soluzioni ammissibili come quei sottoinsiemi di archi che sono archi completamente contenuti in qualche classe di una k -partizione di G . Cioè, $S \subseteq E$ è una soluzione ammissibile se e solo se esiste una k -partizione $\pi = \{C_1, \dots, C_k\}$ tale che, per ogni arco $uv \in S$, il nodo u e il nodo v sono entrambi contenuti in una stessa classe della partizione π . In questo

caso, la funzione di costo $w(S)$ è semplicemente la somma degli archi contenuti in S .

Una seconda rappresentazione è in qualche senso più naturale, perché discende direttamente dalla definizione di k -partizione. Infatti, una k -partizione può essere definita come una funzione che associa a ogni nodo v un intero appartenente all'intervallo $[1, \dots, k]$, e cioè (l'indice del) la classe di appartenza. Quindi, una soluzione è completamente definita da un insieme di n coppie $(v_1, i_1), (v_2, i_2), \dots, (v_n, i_n)$, ove con i_j si è indicato la classe di appartenenza del nodo v_j . Quindi, l'insieme di base è l'insieme $B = \{(v, i) : v \in V, i = 1, \dots, k\}$, mentre una soluzione $S \subseteq B$ è un insieme di coppie nodo/classe (v, i) con la proprietà che ogni nodo di V è contenuto in esattamente una coppia di S . La funzione obiettivo, in questo caso, non è una funzione lineare degli elementi di S : in effetti, a ciascun elemento dell'insieme di base non è associato alcun peso. Tuttavia, il costo della soluzione S dipende dalle coppie appartenenti ad S nel senso che queste ultime determinano gli archi che appartengono a una medesima classe della partizione e quindi il costo di S . La funzione di costo sarà quindi $w(S) = \sum_{i=1}^k \sum_{(u,i) \in S, (v,i) \in S} w_{uv}$. Questo è un esempio di funzione obiettivo "atipica" anticipato nella parte iniziale di questo capitolo.

Una importante famiglia di problemi di partizionamento pone una restrizione sulla dimensione delle classi. In particolare, con ogni classe viene associata una cardinalità massima, e il problema diventa: trovare la k -partizione di costo minimo, col vincolo che la cardinalità della j -esima classe non ecceda la cardinalità massima r_j , per $j = 1, \dots, k$.

8.1.4 PL01

Le soluzioni ammissibili di un problema di programmazione lineare (0,1) sono vettori booleani $x \in \{0, 1\}^n$. In genere, ogni componente di un vettore booleano rappresenta il fatto che un certo elemento di un insieme di base $B = \{b_1, \dots, b_n\}$ è scelto oppure no. Tipicamente, se la componente i -esima $x_i = 1$ allora l'elemento b_i è scelto, se $x_i = 0$ allora b_i non è scelto. Ad esempio, nel problema di scelta dei progetti più remunerativi, l'insieme di base è l'insieme dei possibili progetti attivabili e $x_i = 1$ significa che il progetto i -esimo è attivato, $x_i = 0$ significa che il progetto i -esimo non è attivato. In altri termini, un vettore booleano identifica tutti gli elementi dell'insieme di base che sono stati scelti, e cioè un particolare sottoinsieme $S \subseteq B$. Al contrario, dato un qualunque sottoinsieme $S \subseteq B$, possiamo associare a S un vettore booleano x^S detto *vettore d'incidenza di S* , tale che la componente i -esima $x_i^S = 1$ se e solo se $b_i \in S$.

In definitiva, un problema di programmazione lineare (0,1) $\min\{w^T x : Ax \geq b, x \in \{0, 1\}^n\}$ può sempre essere interpretato nel seguente modo:

- Le variabili x_1, \dots, x_n sono in corrispondenza agli elementi di un insieme di base $B = \{b_1, b_2, \dots, b_n\}$.
- I vincoli del problema definiscono la regione ammissibile e quindi ci dicono quali vettori booleani sono ammissibili e quali no. Poichè ogni vettore booleano è in corrispondenza con un sottoinsieme dell'insieme di base, i vincoli definiscono quali sottoinsiemi dell'insieme di base sono soluzioni ammissibili del nostro problema e quali no.
- La funzione obiettivo (lineare) associa un costo w_i a ogni variabile x_i per $i = 1, \dots, n$ e quindi a ciascun elemento b_i dell'insieme di base B . E' facile vedere che il costo di una soluzione x^S , e cioè di un sottoinsieme S di B , è semplicemente la somma dei costi degli elementi appartenenti a S .

8.2 Un sistema multitaxi per il servizio Areopporto Fiumicino - Roma Centro

Descriveremo di seguito un'applicazione reale e recente che ha richiesto l'utilizzazione di diverse tecniche dell'ottimizzazione combinatoria (algoritmi greedy e ricerca locale) e dell'ottimizzazione su reti (cammini minimi e assegnamento). In effetti, gli algoritmi di soluzione descritti nei prossimi paragrafi sono stati scelti come "esempi" di algoritmi proprio perché necessari alla soluzione dell'applicazione descritta. Altre applicazioni avrebbero ovviamente richiesto algoritmi diversi. Si osservi inoltre che la maggior parte dei

problemi di ottimizzazione presenti in "natura" non sono puri e la scelta dei modelli da utilizzare non è univoca dipendendo dalla dimensione delle istanze d'interesse, dalla disponibilità di algoritmi efficienti, etc.

Nel 1999 l'Ente Nazionale per le Energie Alternative (ENEA) è stato incaricato dal ministero dell'ambiente di sviluppare sistemi alternativi per il trasporto urbano. Investigando una serie di strategie per la politica dei trasporti, l'ENEA si è resa conto dell'esistenza di alcuni problemi di ottimizzazione connessi alle proposte allo studio. Si tratta di un buon esempio di "sinergia" fra diverse competenze, e soprattutto si tratta di un buon esempio di utilizzazione delle metodologie dell'ottimizzazione intese non come solutori universali, ma come "mattoncini" (in inglese "building block") per sviluppare sistemi complessi.

Fra le varie strategie individuate dall'ENEA, una consisteva nello sviluppo di un sistema integrato di *taxi collettivo* (detto anche *multitaxi*) per il trasporto di persone dall'aeroporto di Fiumicino alle loro destinazioni cittadine e viceversa.

Il taxi collettivo è un'interessante alternativa ai tradizionali sistemi di trasporto pubblico. La principale caratteristica di questo tipo di servizio risiede nel fatto che i percorsi dei veicoli non sono prefissati (come nel caso delle linee degli autobus), ma sono calcolati di volta in volta in base alla domanda; questa flessibilità è propria del servizio taxi "classico", ma a differenza di quest'ultimo, il servizio multitaxi non è individuale. Un sistema di trasporto multitaxi è composto da:

- Una flotta di vetture, non necessariamente uguali tra loro, il cui compito è prelevare dei clienti dai loro punti di partenza e trasportarli ciascuno nelle rispettive destinazioni.
- Uno o più depositi in cui gli automezzi ritornano alla fine del turno di servizio.
- Una centrale di gestione che raccoglie le richieste di servizio degli utenti e comunica alle vetture i percorsi da seguire e i clienti da servire.

Il servizio può essere di tipo immediato o a prenotazione. Nel primo caso le richieste dei clienti sono soddisfatte appena possibile, a partire dal momento della loro formulazione, oppure sono subito rifiutate se non è possibile (o conveniente) soddisfarle. Nel secondo caso le domande devono pervenire alla centrale di gestione con un certo anticipo per essere servite in un secondo momento.

Le richieste degli utenti sono espresse in termini di località di prelievo, località di destinazione e, nel caso di servizio a prenotazione, orario desiderato (di partenza o di arrivo).

Il servizio, inoltre, può essere dei seguenti tipi:

- Molti a molti. In questo modo si indica l'esistenza di molteplici (e distinti) punti di raccolta e di consegna. Questo implica che le vetture, durante i loro viaggi, effettuano sia prelievi che consegne di clienti.
- Uno a molti. In questo caso, invece, si indica l'esistenza di un unico punto di raccolta e di molteplici (e distinti) punti di consegna; di conseguenza le vetture effettuano un solo prelievo di clienti, all'inizio del viaggio, dopodiché si limitano a portare a destinazione tutti i passeggeri a bordo.
- Molti a uno. Le vetture effettuano una serie di prelievi e concludono il loro viaggio nell'unico punto di discesa per i clienti.

Un sistema di trasporto multitaxi è particolarmente adatto in situazioni di utenza debole (anziani e disabili) e nel caso di domanda scarsa (trasporto urbano notturno, servizio in aree poco abitate).

Nel caso specifico del servizio multitaxi dall'aeroporto di Fiumicino a Roma centro, si tratta di un servizio immediato di tipo uno a molti. Coloro che intendono avvalersi di questo servizio si presentano ad uno sportello di accettazione dove vengono registrati i loro dati (nominativo, ora di arrivo allo sportello, indirizzo di destinazione,...). Terminata la fase di registrazione, i clienti restano in attesa fino al momento in cui viene loro comunicato di salire su una vettura per partire.

Per semplicità supporremo che tutte le autovetture abbiano la stessa capacità (ad esempio, possono trasportare al massimo 5 passeggeri). Il sistema automatico, ad intervalli regolari (ad esempio ogni

minuto), suddivide i clienti in attesa in gruppi composti al massimo da cinque persone, assegna ciascun gruppo a un taxi specifico, decide quali gruppi debbano partire e, infine, calcola per ogni taxi l'itinerario da seguire, ovvero la sequenza delle consegne. I clienti che compongono i gruppi selezionati per la partenza vengono fatti salire sulle vetture e queste ultime iniziano il loro viaggio, seguendo le indicazioni di percorso fornite dalla centrale di gestione. I dati dei clienti partiti vengono quindi rimossi dal sistema. Infine, le vetture devono tornare all'areoporto per accogliere nuovi passeggeri. Bisogna quindi decidere a) come si compongono i gruppi (cioè gli equipaggi dei singoli taxi) e b) quale sia, per ogni vettura, il percorso migliore da seguire. Benché gli obiettivi possono essere molteplici, in prima analisi potremmo assumere che il gestore debba soddisfare le richieste dei clienti cercando di minimizzare il tempo medio (o le distanze) percorse dai suoi taxi. Si tratta evidentemente di un problema di ottimizzazione. Esiste una naturale decomposizione del problema in due fasi: 1. fase di clustering, o di partizionamento, in cui i clienti attualmente in attesa vengono suddivisi in un numero di gruppi di dimensione al più pari alla capacità dei veicoli. Tipicamente si preferisce aggregare clienti con destinazioni il più possibile vicine fra loro. 2. fase di istradamento (in inglese *"routing"*). In questa fase si decidono i percorsi che i taxi devono seguire per scaricare i passeggeri. I percorsi devono essere tali da minimizzare, ad esempio, le distanze percorse o i tempi di viaggio.

Per poter modellare quanto sopra descritto in termini di problema di ottimizzazione è conveniente definire una struttura matematica adeguata per rappresentare strade, incroci e destinazioni.

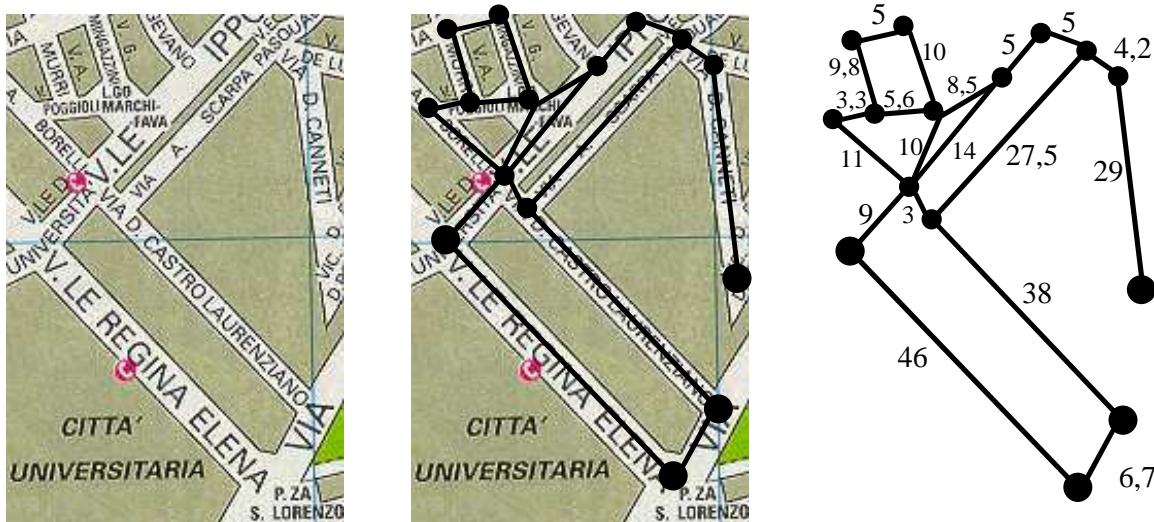


Figura 8.4: Dettaglio del grafo di Roma

Il grafo di Roma Tutte le possibili destinazioni di clienti sono rappresentate dal grafo di Roma. Si tratta di un grafo orientato $R = (W, A)$ dove l'insieme dei nodi W rappresenta gli incroci (o piazze) della città mentre gli archi sono i tratti di strada compresi fra coppie di incroci (vedi Fig. 8.4). La lunghezza c_{uv} di ogni arco $(u, v) \in A$ rappresenta la lunghezza effettiva del tratto di strada corrispondente.

Per semplificare la trattazione successiva, supporremo sempre che la destinazione dei clienti coincida con una piazza o un incrocio, cioè con un nodo del grafo di Roma. Qualora questa ipotesi sia troppo restrittiva, in corrispondenza cioè a tratti di strada molto lunghi fra due incroci adiacenti, si può sempre aggiungere un incrocio *fittizio* suddividendo il tratto in due tratti consecutivi separati (vedi figura 8.5).

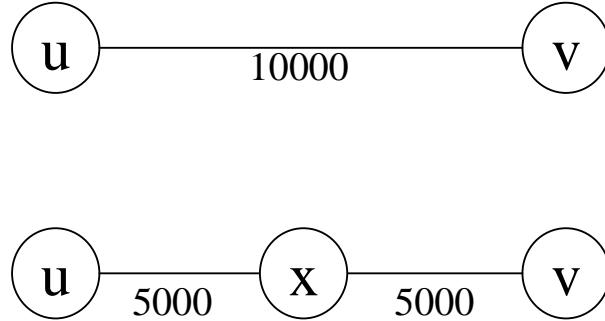


Figura 8.5: Aggiunta di nodi fintizi

Poichè ogni cliente ha una destinazione e ogni destinazione è un nodo del grafo di Roma, i clienti attualmente in attesa sono in corrispondenza con un sottoinsieme di nodi del grafo di Roma. Quindi, il problema di partizionare i clienti in modo che le destinazioni di clienti appartenenti a una stessa classe siano fra loro il più vicino possibile può essere riformulato come un problema di clustering (partizionamento dei nodi) nel sottografo del grafo di Roma indotto dai clienti in attesa.

Una volta che i clienti siano stati assegnati ai veicoli si devono ottimizzare i percorsi dei taxi. E' facile vedere che il problema di minimizzare la lunghezza del percorso di un taxi che deve scaricare i suoi passeggeri in alcuni nodi del grafo di Roma e quindi tornare al (nodo di) Fiumicino è esattamente il problema di calcolare un ciclo hamiltoniano di lunghezza minima nel grafo indotto dai nodi di destinazione dei passeggeri e dal nodo di Fiumicino.

Quindi, il problema di gestione di un servizio multitaxi può essere affrontato risolvendo in sequenza due problemi di ottimizzazione su grafi: il problema di clustering (sul grafo indotto da tutti i clienti in attesa) e il problema del commesso viaggiatore (TSP) sul grafo indotto dai nodi destinazione dei clienti di un taxi. Chiaramente, questo secondo problema andrà risolto per ogni vettura in partenza.

Un'altra assunzione che faremo di seguito per semplificare le descrizioni degli algoritmi è l'ipotesi di *grafo completo*.

Definizione 8.2.1 *Un grafo $R = (W, A)$ si dice completo se, per ogni coppia di nodi distinti $u, v \in W$, l'arco (u, v) appartiene ad A .*

Chiaramente, il grafo di Roma non soddisfa l'ipotesi di completezza, a causa della presenza di moltissime coppie di piazze o incroci (la maggior parte) non collegati direttamente da archi. Tuttavia, per i nostri scopi è possibile completare il grafo, e cioè aggiungere gli archi mancanti, sfruttando le seguenti considerazioni. Quando un taxi si deve spostare da un generico nodo u a un nodo v distinto e non collegato direttamente a u , dovrebbe preferibilmente scegliere il percorso più breve che da u porta a v , e cioè il cammino orientato di lunghezza minima da u a v . Nell'ipotesi di taxi "intelligente", quindi, possiamo rimpiazzare l'arco mancante con un arco fittizio di lunghezza pari alla lunghezza del cammino minimo da u a v . In altri termini, prima di applicare gli algoritmi di ottimizzazione, il grafo di Roma subirà una fase di *pre-processamento* nella quale:

1. per ogni coppia (orientata) di nodi distinti u e v viene calcolato (e memorizzato in opportune strutture dati) il cammino minimo da u a v : sia c_{uv}^* la lunghezza di tale cammino. A tal scopo può essere adoperato l'algoritmo di Dijkstra descritto nel Capitolo ??.

2. per ogni coppia (orientata) di nodi distinti u e v si aggiunge l'arco (u, v) all'insieme di archi A : la lunghezza associata sarà posta uguale a c_{uv}^* .

8.3 Euristiche di tipo "Greedy"

Una prima classe di heuristiche per la soluzione di problemi di ottimizzazione combinatoria va sotto il nome di *euristiche di tipo "greedy"*. Il termine inglese "greedy" può essere tradotto come *avid* ed è stato scelto per rappresentare una caratteristica fondamentale dell'algoritmo greedy, appunto la cosiddetta *scelta greedy*. Per costruire la soluzione finale, che ricordiamo essere un sottoinsieme S dell'insieme di base B , l'algoritmo parte in genere da una soluzione parziale che estende aggiungendo un elemento per volta fino a costruire la soluzione finale. L'elemento scelto di volta in volta è quello che minimizza il costo della funzione obiettivo (da cui "avid"). Inoltre, una volta scelto un elemento, questo non viene mai rimosso dalla soluzione (altra caratteristica dell'avidità). Di seguito, denoteremo con $B = \{b_1, \dots, b_n\}$ l'insieme di base e con $\mathcal{S} = \{S_1, \dots, S_m\}$ l'insieme delle soluzioni ammissibili del problema di ottimizzazione combinatoria. Inoltre, supponiamo che un costo sia associato a *qualsiasi* sottoinsieme di B (non solo alle soluzioni ammissibili). E cioè, indicando con \mathcal{T} la famiglia di tutti i sottoinsiemi di B , supporremo che sia definita una funzione $w : \mathcal{T} \rightarrow \mathbb{R}$ per ogni sottoinsieme $T \in \mathcal{T}$. L'obiettivo è sempre quello di trovare una soluzione ammissibile $S \in \mathcal{S} \subseteq \mathcal{T}$ tale che $w(S)$ sia minimo.

Per introdurre formalmente l'algoritmo greedy, abbiamo bisogno della seguente definizione:

Definizione 8.3.1 *Un sottoinsieme T dell'insieme di base B è detto soluzione parziale (o anche sottosoluzione), se esiste una soluzione $S \in \mathcal{S}$ tale che $T \subseteq S$.*

Si osservi che:

- a. Se S è una soluzione, allora S è anche una soluzione parziale. Infatti $S \subseteq S$.
- b. Se T è una soluzione parziale, allora è sempre possibile costruire una soluzione S tale che $T \subseteq S$ semplicemente inserendo elementi a T .

L'algoritmo greedy costruisce una sequenza di soluzioni parziali T_0, T_1, \dots, T_q con la proprietà che T_0 è l'insieme vuoto, T_q è una soluzione ammissibile del problema e $T_i = T_{i-1} \cup \{e\}$, ove $e \in B - T_{i-1}$ è l'elemento tale che $T_{i-1} \cup \{e\}$ sia ancora una soluzione parziale e abbia costo minimo (fra tutte le soluzioni parziali ottenibili aggiungendo un elemento a T_{i-1}). Il seguente schema fornisce una versione semplificata dell'algoritmo greedy:

Algoritmo Greedy

- a. *Inizializzazione.* Poni $T_0 = \emptyset$. Poni $i = 1$.
- b. *Iterazione i -esima.* Scegli $e \in B - T_{i-1}$ tale che $T_{i-1} \cup \{e\}$ sia una soluzione parziale e $w(T_{i-1} \cup \{e\})$ sia minimo.
- c. Poni $T_i = T_{i-1} \cup \{e\}$.
- d. *Terminazione* Se T_i è una soluzione ammissibile: STOP.
- e. Altrimenti Poni $i = i + 1$. Va al passo b.

Si osservi che il passo [b.] possiede un certo grado di indeterminatezza. In particolare, potrebbero esistere più elementi che minimizzano il costo della nuova soluzione parziale. In questo caso, è necessaria una regola per "dirimere i pareggi" (in inglese "tie breaking rule"). A volte è sufficiente scegliere a caso (*random choice*); altre volte è più opportuno definire una nuova funzione di costo. Vedremo di seguito esempi delle due alternative.

Per concludere questa sezione, si osservi che ogni volta si voglia applicare l'algoritmo greedy a un qualunque problema di ottimizzazione combinatoria, bisognerà innanzitutto definire:

- l'insieme di base
- la famiglia delle soluzioni
- la famiglia delle soluzioni parziali
- la funzione obiettivo

Poichè il nostro esempio prevede la soluzione in cascata di un problema di partizionamento di grafi e di un problema di commesso viaggiatore, considereremo appunto le applicazioni dell'algoritmo greedy a tali problemi. Consideriamo innanzitutto l'applicazione dell'algoritmo greedy al problema del TSP.

8.3.1 Applicazione dell'algoritmo greedy generico al problema del Commesso Viaggiatore.

Consideriamo il problema del Commesso Viaggiatore (TSP). Come visto nel paragrafo 8.1.2, una soluzione ammessa per il TSP è un insieme di archi che definisce un ciclo hamiltoniano. Si consideri ad esempio il grafo in Fig. 8.6. L'insieme delle soluzioni è rappresentato da tutti i possibili cicli hamiltoniani del grafo. Un possibile ciclo hamiltoniano è, ad esempio $S_1 = \{(1, 2), (2, 3), (3, 4), (4, 5), (5, 1)\}$, e il suo costo (somma dei costi degli archi) è pari a 24. Un altro ciclo hamiltoniano è $S_2 = \{(1, 3), (3, 2), (2, 4), (4, 5), (5, 1)\}$ di costo pari a 32. E' facile vedere che i cicli hamiltoniani in un grafo completo sono in corrispondenza ai possibili ordinamenti circolari dei nodi del grafo. Ad esempio, la soluzione S_1 è in corrispondenza dell'ordinamento $o_1 = (1, 2, 3, 4, 5)$, mentre S_2 è in corrispondenza a $o_2 = (1, 3, 2, 4, 5)$. Quindi, se i nodi sono 5, abbiamo $4! = 24$ cicli hamiltoniani distinti.

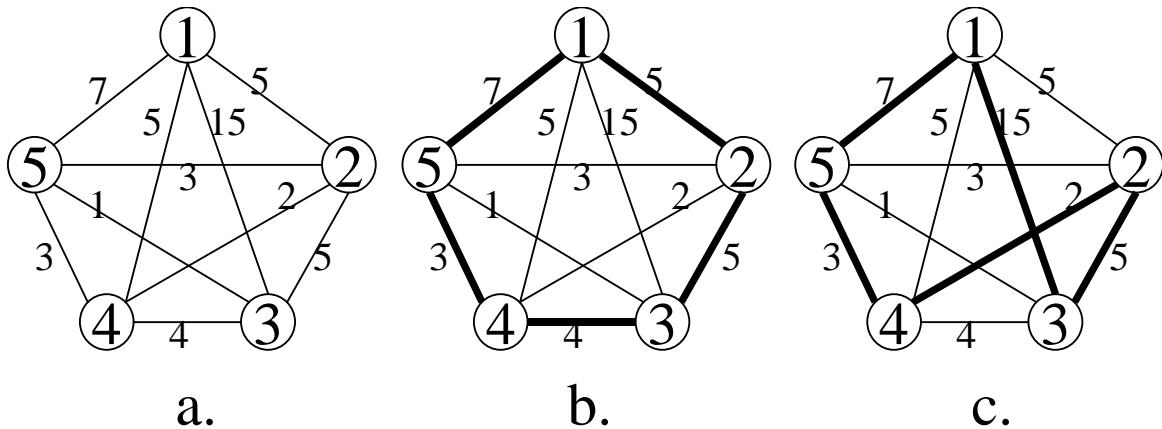


Figura 8.6: Esempi di cicli hamiltoniani

Osservazione 8.3.2 Il numero di cicli hamiltoniani in un grafo completo con n nodi è pari a $(n - 1)!$.

La precedente osservazione permette di rilevare che il numero di cicli hamiltoniani di un grafo completo cresce *esponenzialmente* col numero di nodi. In altri termini, quando il numero di nodi è elevato, l'esplorazione completa dell'insieme delle soluzioni diventa impraticabile ed è cruciale sviluppare delle buone euristiche per la soluzione del problema.

Vediamo come l'algoritmo greedy si applica al TSP. Innanzitutto dobbiamo rispondere alla domanda: chi sono le soluzioni parziali del problema del Commesso Viaggiatore? Per vedere ciò consideriamo innanzitutto le proprietà di cui deve godere un insieme di archi s perché esso sia un ciclo hamiltoniano.

Osservazione 8.3.3 *Dato un grafo $G = (V, A)$ e un insieme $S \in A$, S è l'insieme di archi di un ciclo hamiltoniano se e solo se:*

1. *in ogni nodo di G incidono esattamente due archi di S*
2. *S non contiene cicli di cardinalità inferiore a $|V|$.*

Per capire la natura della condizione 2., si osservi la figura 8.7. In grassetto è evidenziato un insieme S di archi che soddisfa la condizione 1. ma non la condizione 2. (l'insieme contiene infatti cicli di cardinalità inferiore a 6). In effetti, l'insieme S non rappresenta in questo caso un ciclo hamiltoniano.

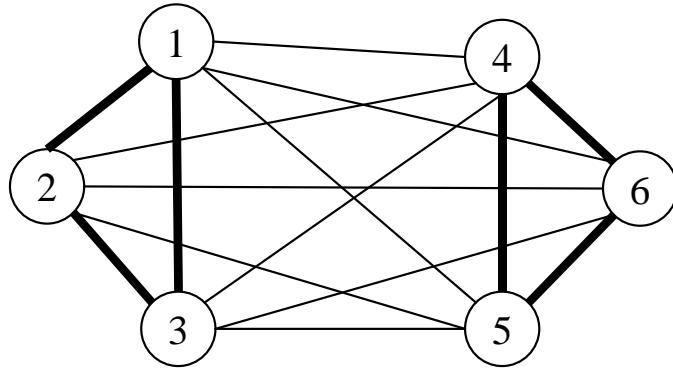


Figura 8.7: Esempi di cicli hamiltoniani

E' facile a questo punto vedere che T è una soluzione parziale per il TSP se e solo se

1. *in ogni nodo di G incidono al più due archi di T .*
2. *T non contiene cicli di cardinalità inferiore a $|V|$.*

Un altro elemento da determinare è la funzione di costo. Si è detto che il costo di un ciclo hamiltoniano è pari alla somma dei costi (o lunghezze) degli archi che lo compongono. L'immediata estensione è la seguente: il costo di un sottoinsieme di archi T è pari alla somma dei costi degli archi nel sottoinsieme, e cioè $w(T) = \sum_{e \in T} w_e$.

Siamo ora in grado di applicare l'algoritmo greedy al grafo di Fig. 8.6.

Inizializzazione.

- a. $T_0 = \emptyset$ ($w(T_0) = 0$). $i = 1$.

Iterazione 1.

- b. Poichè ogni arco del grafo può essere aggiunto a T_0 soddisfacendo le condizioni di soluzione parziale, scegliamo l'arco che costa di meno, cioè l'arco $(3, 5)$ di costo $w_{3,5} = 1$.
- c. Poni $T_1 = T_0 \cup \{(3, 5)\}$ ($w(T_1) = 1$).
- d. Poichè T_1 non è un ciclo hamiltoniano non possiamo fermarci.

- e. Poni $i = 2$.

Iterazione 2.

- b. Anche in questo caso ogni arco (in $A - \{(3, 5)\}$) può essere aggiunto a T_1 soddisfacendo le condizioni di soluzione parziale; scegliamo l'arco che costa di meno, cioè l'arco $(2, 4)$ di costo $w_{2,4} = 2$.
- c. Poni $T_2 = T_1 \cup \{(2, 4)\} = \{(3, 5), (2, 4)\}$ ($w(T_2) = 3$).
- d. T_2 non è un ciclo hamiltoniano.
- e. Poni $i = 3$.

Iterazione 3.

- b. Tutti gli archi residui possono essere aggiunti e quindi scegliamo l'arco meno costoso in $A - T_2$: l'arco $(2, 5)$ di costo $w_{2,5} = 3$. Si osservi che anche l'arco $(4, 5)$ ha costo $w_{4,5} = 3$. La scelta fra due archi equivalenti può essere fatta casualmente. Tuttavia, in altri casi si può decidere un criterio di *risoluzione dei pareggi* ("tie breaking rule"), legato ovviamente al tipo di problema.
- c. Poni $T_3 = T_2 \cup \{(2, 5)\} = \{(3, 5), (2, 4), (2, 5)\}$ ($w(T_3) = 6$).
- d. T_3 non è un ciclo hamiltoniano.
- e. Poni $i = 4$.

Iterazione 4.

- b. Non tutti gli archi in $A - T_3$ possono essere aggiunti a T_3 soddisfacendo le condizioni di soluzione parziale; infatti, se aggiungiamo l'arco $(3, 4)$ si crea il ciclo su quattro nodi $(2, 3), (3, 4), (4, 5), (5, 2)$, violando così la condizione (2) di soluzione parziale. Se aggiungiamo l'arco $(1, 2)$, invece, avremo tre archi incidenti nel nodo 2. Ragionamenti analoghi valgono per gli archi $(1, 5), (2, 3), (4, 5)$. Quindi, gli unici archi che aggiunti non violano le condizioni per le soluzioni parziali, sono l'arco $(1, 4)$ e l'arco $(1, 3)$. Fra i due, scegliamo $(1, 4)$ che ha costo minimo.
- c. Poni $T_4 = T_3 \cup \{(1, 4)\} = \{(3, 5), (2, 4), (2, 5), (1, 4)\}$ ($w(T_4) = 11$).
- d. T_4 non è un ciclo hamiltoniano.
- e. Poni $i = 5$.

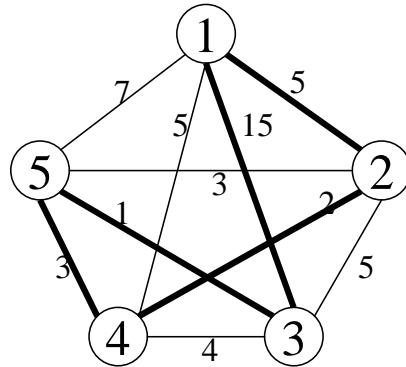
Iterazione 5.

- b. L'unico arco selezionabile è l'arco $(1, 3)$.
- c. Poni $T_5 = T_4 \cup \{(1, 3)\} = \{(3, 5), (2, 4), (2, 5), (1, 4), (1, 3)\}$ ($w(T_5) = 26$).
- d. T_5 è un ciclo hamiltoniano: STOP.

8.3.2 Applicazione dell'algoritmo greedy generico al problema del Partizionamento di Grafi (clustering).

Si è visto come il problema di clustering consiste nel partizionare i nodi di un grafo $G = (V, A)$ in k classi $\mathcal{C} = \{C_1, C_2, \dots, C_k\}$ in modo da minimizzare la somma dei costi degli archi appartenenti alla stessa classe. Una soluzione S (detta k -partizione) può essere rappresentata come un insieme di coppie (v, c) , ove $v \in V$ è un nodo del grafo e $c \in \{C_1, \dots, C_k\}$ è un possibile cluster. Quindi, l'insieme di base B è l'insieme di tutte le coppie $(nodo, cluster)$ e cioè $B = \{(v, c) : v \in V, c \in \mathcal{C}\}$. Una soluzione $S \subseteq B$ è quindi un sottoinsieme di B con la proprietà che, per ogni nodo $v \in V$ esiste una e una sola

Figura 8.8: Soluzione greedy



coppia $(u, c) \in S$ tale che $u = v$. Una conseguenza di ciò è che S conterrà esattamente $|V|$ coppie (cioè $|S| = |V|$). Inoltre, per la nostra applicazione (ogni classe corrisponde a una vettura), le classi hanno una cardinalità massima prefissata. Le soluzioni parziali sono tutti i sottoinsiemi T di B tali che, per ogni nodo $v \in V$, esisterà al più una coppia $(u, c) \in T$ tale che $u = v$. In altri termini, le soluzioni parziali sono k -partizioni parziali, cioè k -partizioni di un sottoinsieme dei nodi di G . Il costo di una soluzione parziale (più in generale, di un sottoinsieme di B) è calcolato ancora una volta come la somma dei costi degli archi appartenenti a una stessa classe.

Supponiamo adesso di avere 5 persone in attesa e due taxi disponibili. Inoltre, supponiamo che ogni taxi possa ospitare al massimo 3 persone. Supponiamo infine che il grafo dei clienti sia quello di Figura 8.6. Vediamo adesso come sia possibile calcolare una 2-partizione (partizione in due classi, bi-partizione) del grafo in Figura 8.6.

Poichè $V = \{1, \dots, 5\}$ e $\mathcal{C} = \{C_1, C_2\}$, l'insieme di base sarà $B = \{(1, C_1), (1, C_2), (2, C_1), (2, C_2), (3, C_1), (3, C_2), (4, C_1), (4, C_2), (5, C_1), (5, C_2)\}$. E' importante osservare che in questo caso i pareggi vengono risolti individuando una funzione di costo ad hoc per motivi che verranno discussi di seguito. Infine, poichè scegliere una coppia di B corrisponde ad assegnare un nodo a un cluster, di seguito adotteremo una terminologia (semplificata) e a fianco della locuzione "scegliere la coppia (v, C_j) ", diremo anche "assegnare il nodo v al cluster C_j ", e via di seguito.

Inizializzazione.

- $T_0 = \emptyset$ ($w(T_0) = 0$). $i = 1$.

(Equivalentemente: crea 2 classi vuote C_1, C_2).

Iterazione 1.

- Poichè ogni coppia può essere aggiunta a T_0 senza incrementare il costo di T_0 (ovvero, ogni nodo può essere assegnato a una qualunque classe a costo nullo), ho bisogno di una regola per decidere quale nodo assegnare a quale classe. Nell'esempio del TSP si era optato per una scelta di tipo "random". In questo esempio si preferisce una regola che tenga in conto della particolare struttura del problema. Una considerazione euristica è la seguente: conviene innanzitutto "sistemare" in qualche classe della partizione quei nodi su cui incidono gli archi più costosi in modo da poter scegliere per essi classi abbastanza (se non del tutto) vuote. Quindi la regola di tie breaking sarà: scegli il nodo per cui la somma degli archi incidenti è massima. Di seguito, indicheremo con $g(v)$ la somma dei costi degli archi incidenti nel nodo v . E' facile verificare che $g(1) = 32$, $g(2) = 17$, $g(3) = 25$, $g(4) = 14$, $g(5) = 14$. Scegliamo quindi il nodo 1. Poichè le coppie $(1, 1)$ e $(1, 2)$ sono equivalenti dal punto di vista del costo delle partizioni corrispondenti (e cioè è indifferente

assegnare il nodo 1 alla classe C_1 o alla classe C_2 , possiamo scegliere in modo casuale. Scegliamo quindi la coppia $(1, C_1)$ (assegnamo cioè il nodo 1 alla classe C_1).

- c. Poni $T_1 = T_0 \cup \{(1, C_1)\}$ ($w(T_1) = 0$).

Equivalentemente $C_1 = \{1\}$, $C_2 = \emptyset$.

- d. Poichè T_1 non è una partizione (completa) dei nodi di G , si continua.

- e. Poni $i = 2$.

Iterazione 2.

- b. Ogni coppia in B che non contiene il nodo 1 può essere aggiunta a T_1 soddisfacendo le condizioni di soluzione parziale. Tuttavia, ogni coppia del tipo (v, C_1) , che corrisponde ad assegnare il nodo v alla classe C_1 , produce un incremento di costo pari al costo dell'arco $(1, v)$ (infatti il nodo 1 appartiene alla classe 1); al contrario, ogni nodo può essere assegnato alla classe 2 senza incrementi di costo (infatti la classe C_2 è ancora vuota). Infine, poichè ogni coppia di tipo (v, C_2) produce un incremento di costo nullo, dobbiamo usare ancora una volta la regola di tie breaking e scegliere quindi il nodo di costo massimo, ovvero il nodo 3 ($g(3) = 25$). Scegliamo quindi la coppia $(3, 2)$, assegnando in questo modo il nodo 3 alla classe 2.

- c. Poni $T_2 = T_1 \cup \{(3, C_2)\} = \{(1, C_1), (3, C_2)\}$ ($w(T_2) = 0$).

$C_1 = \{1\}$, $C_2 = \{3\}$.

- d. T_2 non è una partizione di V .

- e. Poni $i = 3$.

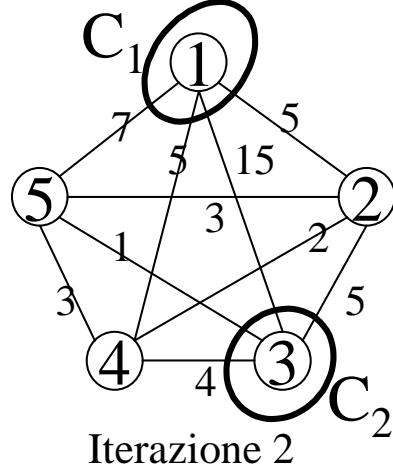
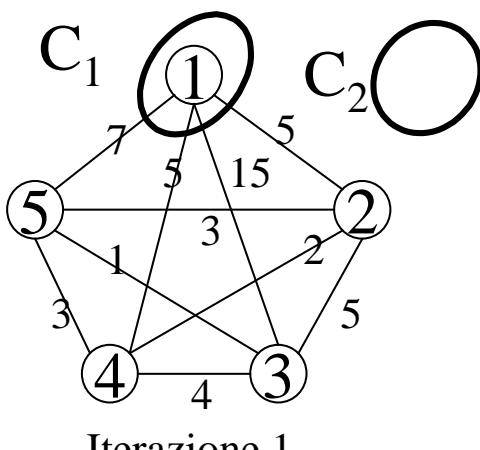


Figura 8.9: Evoluzione delle classi I

Iterazione 3.

- b. In questo caso, ogni coppia aggiungibile a T_2 , e cioè tutte quelle coppie che non contengono il nodo 1 e il nodo 3, producono un incremento di costo positivo. Ad esempio, se il nodo 2 viene assegnato alla classe 1, l'incremento di costo è pari al costo dell'arco $(1,2)$, ovvero 5. E' facile vedere che il minimo incremento di costo si ottiene scegliendo la coppia $(5, C_2)$, ovvero assegnando il nodo 5 alla classe C_2 . Non ci sono pareggi.

- c. Poni $T_3 = T_2 \cup \{(5, C_2)\} = \{(1, C_1), (3, C_2), (5, C_2)\}$ ($w(T_3) = 5$).

$$C_1 = \{1\}, C_2 = \{3, 5\}.$$

- d. T_3 non è una partizione.

- e. Poni $i = 4$.

Iterazione 4.

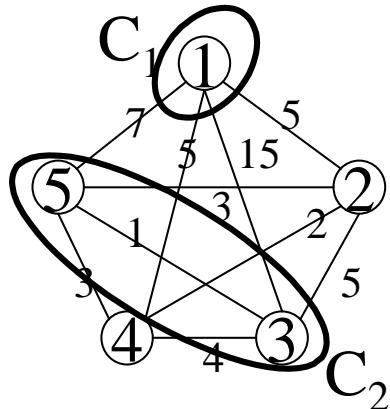
- b. Tutte le coppie contenenti i nodi 1, 3 e 5 non possono essere aggiunte. L'incremento di costo associato alla coppia $(2, C_1)$ è pari a 5, per la coppia $(2, C_2)$ è pari a $3 + 5 = 8$, per la coppia $(4, C_1)$ è pari a 5, per la coppia $(4, C_2)$ è pari a $3 + 4 = 7$. Fra la coppia $(2, C_1)$ e la coppia $(4, C_1)$ si sceglie $(2, C_1)$ perché $g(2) = 17 > g(4) = 14$.

- c. Poni $T_4 = T_3 \cup \{(2, C_1)\} = \{(1, C_1), (3, C_2), (5, C_2), (2, C_1)\}$ ($w(T_4) = 10$).

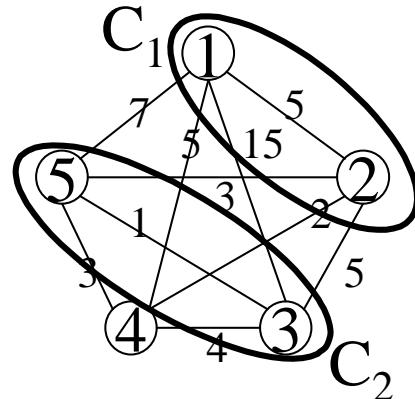
$$C_1 = \{1, 2\}, C_2 = \{3, 5\}.$$

- d. T_4 non è una partizione.

- e. Poni $i = 5$.



Iterazione 3



Iterazione 4

Figura 8.10: Evoluzione delle classi II

Iterazione 5.

- b. Le uniche coppie selezionabili sono $(4, C_1)$ e $(4, C_2)$. L'incremento di costo corrispondente a $(4, C_1)$ è pari a 7 ed è uguale all'incremento prodotto dalla coppia $(4, C_2)$. Possiamo scegliere indifferentemente.

- c. Poni $T_4 = T_3 \cup \{(2, C_1)\} = \{(1, C_1), (3, C_2), (5, C_2), (2, C_1), (4, C_2)\}$ ($w(T_5) = 17$)

$$C_1 = \{1, 2\}, C_2 = \{3, 4, 5\}.$$

- d. T_5 è una partizione: STOP.

Un'ultima osservazione riguarda la forma con cui vengono generalmente rappresentati gli algoritmi greedy. Infatti, raramente si fa esplicito riferimento all'insieme di base (o alle soluzioni ammissibili come sottinsiemi dell'insieme di base). Normalmente gli algoritmi vengono presentati in una forma molto

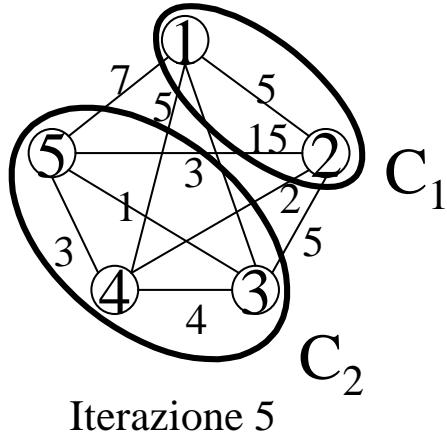


Figura 8.11: Evoluzione delle classi III

più vicina alla descrizione "naturale" del problema. Come visto nell'esempio del problema di clustering, ogni coppia dell'insieme di base rappresenta in realtà l'assegnazione di un nodo a un cluster. Una forma alternativa e più leggibile dell'algoritmo greedy generico applicato al problema di clustering è la seguente.

Problema. Sia dato un grafo $G = (V, A)$, con costi $c_a \in \mathbb{R}$ associati a ciascun arco $a \in A$. Trovare la partizione dell'insieme dei nodi V in k classi C_1, \dots, C_k che minimizza il costo $\sum_{i=1}^k \sum_{u \in C_i, v \in C_i} c_{uv}$.

Algoritmo Greedy particolarizzato per il problema di clustering

Note: l'insieme W utilizzato nell'algoritmo rappresenta l'insieme dei nodi già assegnati (nelle iterazioni precedenti a quella corrente) a qualche cluster.

- Inizializzazione.* Poni $C_i = \emptyset$ per $i = 1, \dots, k$. Poni $W = \emptyset$. Poni $i = 1$.
- Iterazione i -esima.* Scegli un nodo $u \in V - W$ e assegna u a un cluster C_j in modo da minimizzare l'incremento di costo della partizione parziale, e cioè la quantità $\sum_{v \in C_j} c_{uv}$. In caso di pareggi, scegli il nodo u che massimizza $g(u)$.
- Poni $C_j = C_j \cup \{u\}$.
- Terminazione* Se $W = V$ ogni nodo è stato assegnato a un cluster: STOP.
- Altrimenti Poni $i = i + 1$. Va al passo b.

E' in questa forma che verranno di seguito descritti gli algoritmi di ricerca locale per l'ottimizzazione combinatoria.

8.3.3 Una diversa forma del generico algoritmo greedy.

Gli esempi di applicazione dell'algoritmo greedy sopra elencati hanno una specifica struttura delle soluzioni e delle soluzioni parziali. In particolare, tutte le soluzioni parziali generate dal greedy - tranne l'ultima - non sono soluzioni ammissibili per il problema. Inoltre la funzione obiettivo peggiora a ogni iterazione, ma noi siamo comunque obbligati a continuare i passi finché la sottosoluzione diventa una soluzione del problema.

Per altri problemi di ottimizzazione, come il problema dell'accoppiamento, le soluzioni parziali sono anche soluzioni del problema originario. Infatti, se $M = \{e_1, \dots, e_q\}$ è un matching, allora ogni sottosinsieme di M è ancora un matching (e cioè una soluzione ammissibile). In questo caso si tende in generale

a estendere le soluzioni parziali perché tipicamente la funzione obiettivo migliora all'aumentare del numero di elementi contenuti nella soluzione. Per problemi di questo tipo, il generico algoritmo greedy (per problemi di minimizzazione) viene riscritto nel modo seguente:

Algoritmo Greedy II

- Inizializzazione.* Poni $T_0 = \emptyset$. Poni $i = 1$.
- Iterazione i -esima.* Scegli $e \in B - T_{i-1}$ tale che $T_{i-1} \cup \{e\}$ sia una soluzione parziale e $w(T_{i-1} \cup \{e\})$ sia minimo.
- Se per ogni $e \in B - T_{i-1}$, $T_{i-1} \cup \{e\}$ non è una soluzione ammissibile: STOP. T_{i-1} è la soluzione greedy.
- Altrimenti se $w(T_{i-1}) < w(T_{i-1} \cup \{e\})$: STOP. T_{i-1} è la soluzione greedy.
- Altrimenti Poni $T_i = T_{i-1} \cup \{e\}$. Poni $i = i + 1$. Vai al passo b.

Il passo [c.] serve ad assicurare che l'algoritmo termini qualora ogni elemento residuo non può essere aggiunto alla soluzione parziale corrente senza violare il vincolo di essere soluzione parziale. Il passo [d.] assicura invece che se la prossima soluzione parziale è peggiore di quella attuale essa non venga generata e l'algoritmo termini. Si osservi infine che in questa forma l'algoritmo greedy può essere applicato anche a problemi in cui le sottosoluzioni via via generate non sono soluzioni ammissibili. Si consideri ad esempio il problema del Commesso Viaggiatore. A ogni iterazione generiamo sottosoluzioni che non sono cicli hamiltoniani finché, all'iterazione n -esima, genereremo il nostro ciclo hamiltoniano. Secondo lo schema qui sopra riportato, all'iterazione $n + 1$ l'algoritmo terminerà in quanto l'aggiunta di un qualunque arco a un ciclo hamiltoniano non produce una sottosoluzione: quindi sarà verificato il test di terminazione al passo [c.].

E' facile vedere che per adattare il precedente algoritmo a problemi di massimizzazione è sufficiente cambiare il passo [b.] scegliendo l'elemento e tale che $w(T_{i-1} \cup \{e\})$ sia massimo e invertendo il senso della disequazione nel test al passo [d.] che va' riscritto come:

- Altrimenti se $w(T_{i-1}) > w(T_{i-1} \cup \{e\})$: STOP. T_{i-1} è la soluzione greedy.

Si consideri l'esempio di Fig. 8.12, ove si voglia calcolare l'accoppiamento di peso massimo.

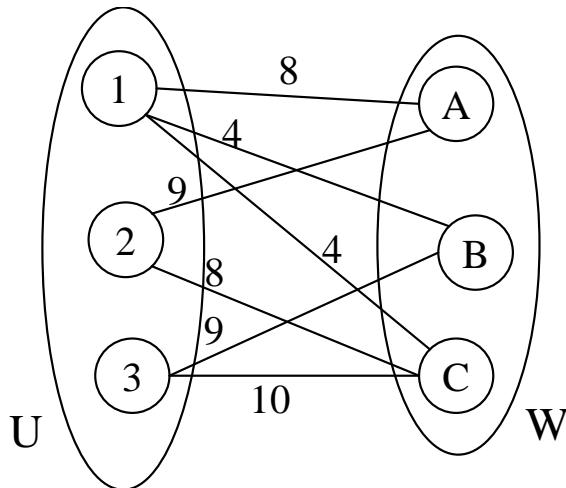


Figura 8.12: Un problema di accoppiamento massimo

Inizializzazione.

- a. $T_0 = \emptyset$ ($w(T_0) = 0$). $i = 1$.

Iterazione 1.

- b. Poichè ogni arco può essere aggiunto a T_0 mantenendo la proprietà di essere una soluzione parziale, scegliamo l'arco di peso massimo $(3, C)$.
- c. L'arco esiste, il test di terminazione non è soddisfatto.
- d. $10 = w(T_0 \cup \{(3, C)\}) > w(T_0)$ e il secondo test di terminazione non è soddisfatto (si ricordi che si tratta di un problema di massimizzazione).
- e. Poni $T_1 = T_0 \cup \{(3, C)\}$ ($w(T_1) = 10$). Poni $i = 2$.

Iterazione 2.

- b. Gli archi $(1, C)$ e $(2, C)$ non possono essere aggiunti a T_1 in quanto incidenti in C (esiste già un arco in T_1 incidente in C). Analogamente, l'arco $(3, B)$ non può essere aggiunto perchè incidente in 3. Fra gli archi residui, quello di peso massimo è l'arco $(2, A)$, di peso 9.
- c. L'arco esiste, il test di terminazione non è soddisfatto.
- d. $18 = w(T_1 \cup \{(2, A)\}) > w(T_1)$ e il secondo test di terminazione non è soddisfatto.
- e. Poni $T_2 = T_1 \cup \{(2, A)\}$ ($w(T_2) = 19$). Poni $i = 3$.

Iterazione 3.

- b. L'unico arco che può essere aggiunto a T_2 è l'arco $(1, B)$, di peso 4.
- c. L'arco esiste, il test di terminazione non è soddisfatto.
- d. $23 = w(T_2 \cup \{(1, B)\}) > w(T_2)$ e il secondo test di terminazione non è soddisfatto.
- e. Poni $T_3 = T_2 \cup \{(1, B)\}$ ($w(T_3) = 23$). Poni $i = 4$.

Iterazione 4.

- b. Nessun arco può essere aggiunto a T_3 .
- c. L'arco non esiste, il test di terminazione è soddisfatto. T_3 è la soluzione greedy.

La soluzione trovata dall'algoritmo greedy per il grafo di Fig. 8.12 è mostrata in Fig. 8.13.

8.4 Ricerca Locale

Si è osservato come uno degli elementi caratterizzanti dell'algoritmo greedy sia la natura irrevocabile della scelta greedy. Tuttavia, assai spesso apportando "piccole" modifiche alla soluzione greedy si possono avere miglioramenti nel valore della funzione obiettivo. Si consideri l'esempio del ciclo hamiltoniano di Fig. 8.14.a, che è quello prodotto dall'algoritmo greedy (si confronti con la Fig. 8.6). Il costo del ciclo è pari a 26. Per comodità gli archi non appartenenti al ciclo non sono stati rappresentati nella figura. Se rimuoviamo gli archi $(1, 3)$ e $(4, 5)$ otteniamo la soluzione parziale di Fig. 8.14.b. Esistono due soli modi di completare questo insieme di archi per renderlo un ciclo hamiltoniano. Il primo è re-inserire gli archi $(1, 3)$ e $(4, 5)$, riottenendo così il ciclo di partenza. Il secondo è invece scegliere gli archi $(3, 4)$ e $(1, 5)$, ottenendo così il ciclo hamiltoniano di Fig. 8.14.c.

Si osservi che questo ciclo ha costo 19, cioè un costo inferiore al ciclo di partenza. Quindi, con una piccola modifica della soluzione greedy (cambiando solo due archi) si è ottenuto un sostanziale

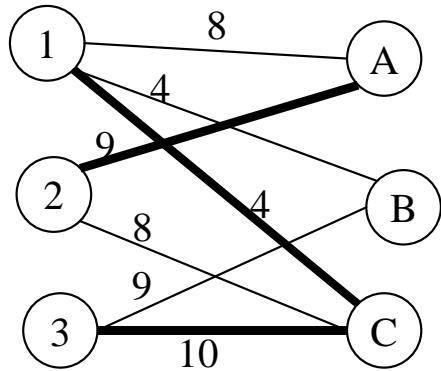


Figura 8.13: Matching massimo

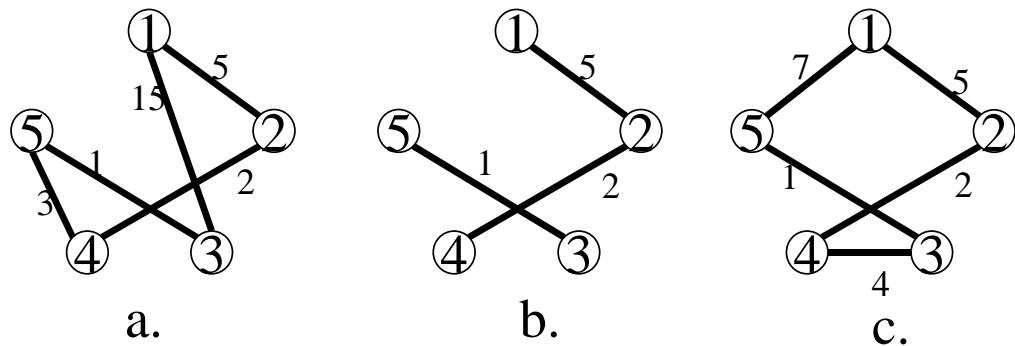


Figura 8.14: Trasformazione di un ciclo hamiltoniano

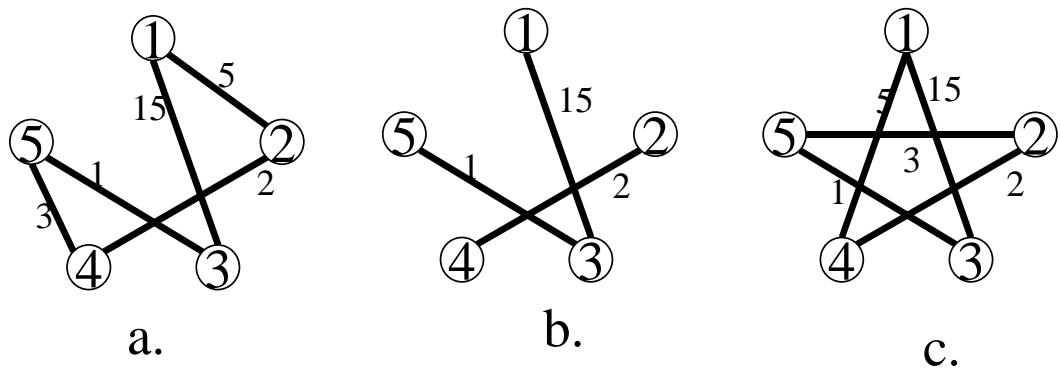


Figura 8.15: Un'altra possibile trasformazione della soluzione greedy

miglioramento della funzione obiettivo. Proviamo ora, sempre partendo dal grafo di Fig.8.14 a sostituire la coppia $(1, 2)$ e $(4, 5)$. In questo caso, l'unico modo per ricostruire un ciclo hamiltoniano diverso da quello di partenza è aggiungere gli archi $(1, 4)$, $(2, 5)$ (si veda la Fig. 8.15).

Questo nuovo ciclo tuttavia ha lo stesso costo di quello di partenza. Cerchiamo di generalizzare ciò che è stato fatto. Innanzitutto osserviamo che, se rimuoviamo dal ciclo di partenza due archi adiacenti (ovvero incidenti in uno stesso nodo), esiste un solo modo per completare la soluzione parziale così ottenuta consistente nel re-inserire gli archi appena eliminati. Al contrario, se rimuoviamo due archi non adiacenti (cioè due archi che non hanno nodi in comune), la soluzione parziale può essere completata in due modi distinti, uno dei quali produce un nuovo ciclo. In generale, se indichiamo con $H_0 = \{(u_1, u_2), (u_2, u_3), \dots, (u_{q-1}, u_q), (u_q, u_1)\}$ un generico ciclo hamiltoniano, allora il meccanismo di generazione di un nuovo ciclo hamiltoniano può essere descritto come segue (si veda la Fig. ??):

1. Scegli una coppia di archi non adiacenti (u_i, u_{i+1}) e (u_j, u_{j+1}) .
2. Rimuovi la coppia di archi dal ciclo
3. Aggiungi i due nuovi archi (u_i, u_j) e (u_{i+1}, u_{j+1}) .

Poichè l'operazione coinvolge lo scambio di due archi con altri due archi, verrà chiamata *2-scambio* (in inglese *"2-exchange"*).

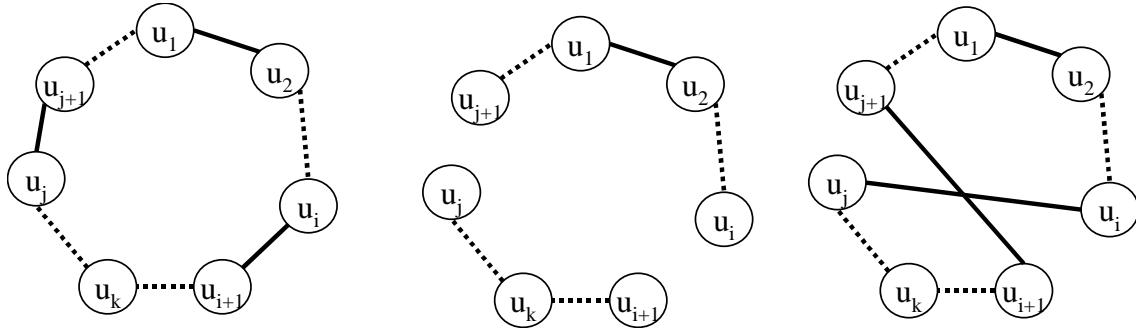


Figura 8.16: Scambio di archi generalizzato

Negli esempi delle figure 8.14 e 8.15 ci siamo limitati a provare la sostituzione di due coppie di archi del ciclo originario H_0 . Un'ovvia generalizzazione consiste nel provare tutti i possibili 2-scambi (in corrispondenza a tutte le possibili coppie di archi non adiacenti di H_0) e scegliere alla fine il 2-scambio che produce il ciclo hamiltoniano di costo minimo. Formalmente, possiamo definire la seguente procedura:

1. Per ogni coppia di archi non adiacenti di H_0 calcola la lunghezza del ciclo hamiltoniano corrispondente
2. Scegli il migliore dei cicli così prodotti.

Si osservi che il numero di cicli hamiltoniani esaminati non è molto grande. Infatti, il ciclo iniziale H_0 contiene esattamente n archi (ove $n = |V|$). Quante sono le coppie di archi non adiacenti? Per ogni arco (u, v) , ci sono esattamente due archi adiacenti (un arco incidente in u e un arco incidente in v). Quindi, il numero di archi non adiacenti è $n - 3$. Quindi, per ognuno degli n archi del ciclo, esistono esattamente $n - 3$ 2-scambi distinti. Il numero di cicli generati per 2-scambio a partire da H_0 è quindi minore o uguale a $n(n - 3)$ (esattamente è pari a $n(n - 3)/2$).

Ora, se indichiamo con H_1 il migliore dei cicli hamiltoniani ottenibili per 2-scambio da H_0 saranno possibili due casi:

- a. $w(H_1) < w(H_0)$, cioè il ciclo trovato è migliore di quello di partenza.
- b. $w(H_1) \geq w(H_0)$, cioè il ciclo trovato non è migliore di quello di partenza.

Supponiamo adesso di trovarci nel caso a., e cioè H_1 è migliore di H_0 . Nulla ci impedisce di ri-cominciare da capo, nella speranza di migliorare ulteriormente il ciclo hamiltoniano. Quindi possiamo riapplicare ad H_1 la procedura di ricerca del miglior 2-scambio, eventualmente identificando un nuovo ciclo H_2 migliore di H_1 (e quindi di H_0); infine, la procedura può essere re-iterata finché si riesce a identificare un ciclo migliore del precedente.

Siamo in grado di descrivere adesso una delle euristiche più efficienti per la ricerca di un ciclo hamiltoniano di costo (lunghezza) minimo, la cosiddetta euristica $2 - Opt$. Di seguito, denotiamo con \mathcal{T} la famiglia dei cicli hamiltoniani di G .

Algoritmo 2-opt per il TSP

Inizializzazione

- a. Scegli un ciclo hamiltoniano iniziale $H_0 \in \mathcal{T}$ (ad esempio mediante l'euristica greedy). Poni $i = 1$.

Iterazione i -esima

- b. Sia H_i il più corto ciclo hamiltoniano ottenibile da H_{i-1} per 2-scambio.
- c. Se $w(H_i) \geq w(H_{i-1})$ allora STOP. H_{i-1} è il miglior ciclo trovato fino a questo punto.
- d. Altrimenti ponи $i = i + 1$ e va al passo b.

Fine iterazione i -esima

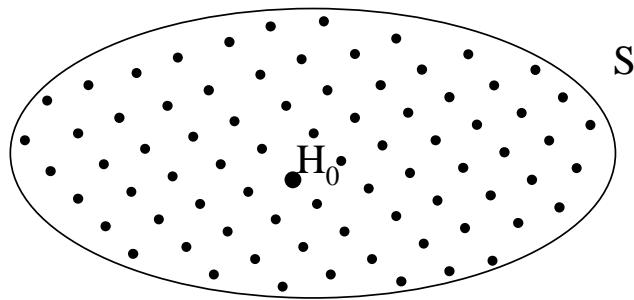


Figura 8.17: Insieme delle soluzioni ammissibili

Quanto abbiamo visto finora applicato al caso del TSP può essere generalizzato a ogni problema di ottimizzazione combinatoria. Esaminiamo più nel dettaglio ciò che è stato fatto. Siamo partiti da un ciclo hamiltoniano iniziale H_0 (*soluzione iniziale*) e abbiamo esaminato un certo numero di cicli hamiltoniani "non troppo diversi" da H_0 . In particolare, abbiamo esaminato tutti quei cicli che hanno esattamente $n - 2$ archi in comune con H_0 . Chiaramente, questo è solo un piccolo sottoinsieme dell'insieme di tutti i cicli hamiltoniani del grafo (che, come si è detto, ha dimensione $n!$). I cicli hamiltoniani appartenenti a questo sottoinsieme hanno la caratteristica di trovarsi, in qualche senso, "vicino" a H_0 . Graficamente,

possiamo rappresentare tutti i cicli hamiltoniani come dei punti in uno spazio di soluzioni S (vedi Fig. 8.17).

Il ciclo H_0 sarà un punto appartenente a questo sottoinsieme. L'insieme di tutti i cicli "vicini" ad H_0 è evidenziato in figura 8.18 da un ovale centrato in H_0 e contenuto in S . Questo insieme è detto in generale *l'intorno* di H_0 , ed è indicato come $N(H_0)$. In questo caso stiamo parlando di un particolare intorno, quello definito dal 2-scambio. Nulla ci impedisce di definire intorni più grandi (ad esempio, l'insieme di tutti i cicli hamiltoniani ottenibili sostituendo 3 archi) o più piccoli. In genere, è bene che la dimensione dell'intorno non cresca troppo per poter effettuare efficientemente la visita (e cioè la generazione) dei cicli hamiltoniani in esso contenuti, il calcolo della funzione obiettivo e quindi la scelta della migliore soluzione appartenente all'intorno.

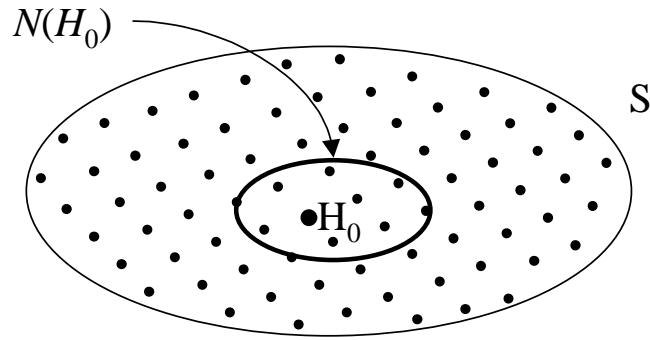


Figura 8.18: Intorno di una soluzione

Continuando con questo esempio, è semplice interpretare l'euristica 2-opt come traiettoria di punti all'interno di questo spazio di soluzioni. Infatti, il metodo può essere interpretato come segue: parto da H_0 , visito il suo intorno, mi sposto nella migliore soluzione trovata H_1 , visito l'intorno di H_1 , mi sposto nella migliore soluzione trovata H_2 , etc. La traiettoria termina quando l'ultima soluzione visitata è migliore (non peggiore) di tutte quelle nel suo intorno.

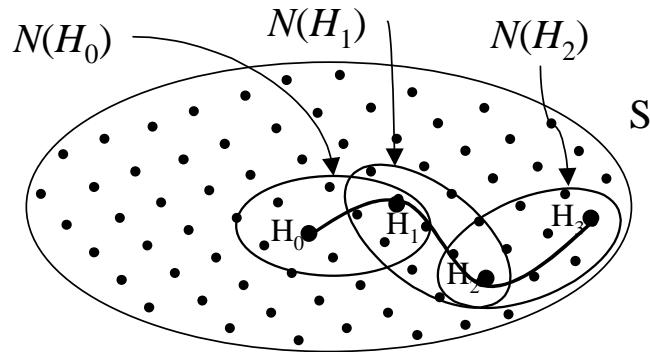


Figura 8.19: Traiettoria nello spazio delle soluzioni

La Figura 8.19 rappresenta appunto sia la traiettoria di soluzioni che la sequenza di intorni visitata. Si capisce a questo punto il significato del termine "Ricerca Locale". La ricerca (nello spazio delle

soluzioni) avviene localmente, negli intorni delle soluzioni sulla traiettoria seguita. L'ultima soluzione è detta *minimo locale*, a sottolineare il fatto che questa soluzione non è l'ottimo globale (su tutto l'insieme delle soluzioni) ma solo l'ottimo locale, relativo alla famiglia di intorni definita.

8.4.1 Algoritmo generico di Ricerca Locale

L'euristica 2-Opt per il TSP è un esempio di euristica di *Ricerca Locale*. Il primo passo per costruire un'euristica consiste nel definire un opportuno intorno delle soluzioni. Chiamando con $S \in \mathcal{S}$ una soluzione ammissibile per il problema di ottimizzazione, si tratta di stabilire come costruire a partire da S un insieme di soluzioni $N(S)$. La definizione dell'intorno avviene tipicamente mediante la specificazione di una particolare operazione, detta *mossa*, da applicare alla soluzione corrente per ottenere le nuove soluzioni. Per esempio, nell'euristica 2-Opt per il Commesso Viaggiatore, la mossa era il cosiddetto 2-scambio (sostituzione di due archi non adiacenti con due nuovi archi). Consideriamo come nuovo esempio il problema di k -partizionamento. Per semplificare la trattazione, considereremo il caso senza vincoli di cardinalità sulle partizioni. Le soluzioni del problema di k -clustering sono dunque tutte le k -partizioni dell'insieme di nodi V del grafo. Quindi, una soluzione è una specifica k -partizione, che possiamo indicare come $S = \{C_1, \dots, C_k\}$. Un modo per definire l'intorno di una partizione è il seguente: l'intorno $N(S)$ di una partizione S è l'insieme di tutte le partizioni ottenibili da S spostando uno e un solo nodo dalla sua classe di appartenenza a una nuova classe. La mossa in questo caso consiste in:

- scegli un nodo $v_j \in V$. Sia C_l la classe di appartenenza di v_j .
- genera una nuova partizione $S' = \{C'_1, C'_2, \dots, C'_l, \dots, C'_q\}$ nel seguente modo:
 - Scegli un indice r con $r \in \{1, \dots, q\}$ e $r \neq l$.
 - Poni $C'_i = C_i$ per $i = 1, \dots, q$ e $i \neq r$, $i \neq l$.
 - Poni $C'_l = C_l - \{v_j\}$, $C'_r = C_r \cup \{v_j\}$.

Se consideriamo ancora l'esempio di Fig. 8.11, la soluzione prodotta dall'algoritmo greedy è: $S_0 = \{C_1 = \{1, 2\}, C_2 = \{3, 4, 5\}\}$. L'intorno di S_0 è formato da cinque soluzioni $N(S_0) = \{S_1, S_2, S_3, S_4, S_5\}$, ove $S_1 = \{C_1 = \{1, 2, 3\}, C_2 = \{4, 5\}\}$, $S_2 = \{C_1 = \{1, 2, 4\}, C_2 = \{3, 5\}\}$, $S_3 = \{C_1 = \{1, 2, 5\}, C_2 = \{3, 4\}\}$, $S_4 = \{C_1 = \{2\}, C_2 = \{1, 3, 4, 5\}\}$, $S_5 = \{C_1 = \{1\}, C_2 = \{2, 3, 4, 5\}\}$. E' facile anche verificare che $w(S_0) = 13$, $w(S_1) = 18$, $w(S_2) = 13$, $w(S_3) = 21$, $w(S_4) = 35$. $w(S_5) = 16$.

Introdotta la definizione di intorno di una soluzione, siamo finalmente in grado di descrivere formalmente il generico algoritmo di ricerca locale.

Algoritmo di Ricerca Locale

Inizializzazione

- a. Scegli una soluzione iniziale $S_0 \in \mathcal{S}$ (ad esempio mediante l'euristica greedy). Poni $i = 1$.

Iterazione i -esima

- b. Sia $S_i \in N(S_{i-1})$ la migliore soluzione nell'intorno di S_{i-1} .
- c. Se $w(S_i) \geq w(S_{i-1})$ allora STOP. S_{i-1} è il minimo locale.
- d. Altrimenti pon $i = i + 1$ e va al passo b.

Fine iterazione i -esima

Ovviamente è sempre possibile interrompere la ricerca locale prima di aver raggiunto un ottimo locale, ad esempio dopo aver eseguito un prefissato massimo numero di iterazioni.

8.5 Estensione del modello al caso Roma Centro - Fiumicino Areoperto

Finora abbiamo trattato il caso del servizio da Fiumicino Areoperto al centro città. In questo contesto le vetture, una volta lasciati i passeggeri alle loro destinazioni, tornano all'areoperto per prelevare nuovi clienti. Tuttavia nulla impedisce loro di trasportare passeggeri da Roma centro all'areoperto di Fiumicino effettuando il servizio anche nel viaggio di ritorno. In effetti, un servizio del genere è allo studio. Il modo più semplice per affrontare il problema consiste nel considerare le due tratte (Fiumicino-Roma Centro e Roma Centro - Fiumicino) come due problemi distinti.

Il caso Roma Centro - Fiumicino presenta qualche elemento di complessità addizionale. Infatti, in questo caso il servizio viene effettuato su prenotazione e i clienti vanno raccolti rispettando il più possibile gli orari stabiliti. In ogni caso, ancora una volta di stratta di un problema di clustering (assegnazione dei clienti alle vetture) e di un problema di Comesso Viaggiatore (scelta del percorso ottimo) risolti in sequenza. Quindi, alla fine del processo di ottimizzazione avremo un certo numero di cicli hamiltoniani $H = \{H_1, H_2, \dots, H_q\}$ ciascuno rappresentante i clienti da prelevare e la sequenza da seguire per ciascun taxi. A ogni ciclo avremo inoltre associato un tempo d'inizio raccolta t_i per $i = 1, \dots, q$, ovvero il primo cliente di ogni ciclo H_i deve essere prelevato al tempo t_i . Si tratta ora di assegnare ciascun ciclo a un taxi proveniente dall'areoperto e che abbia già concluso il suo giro di consegne. Quale vettura conviene assegnare al ciclo H_j (per $j = 1, \dots, q$)? Sicuramente un taxi che abbia finito le consegne in tempo per prelevare il primo cliente al tempo t_j (ma non troppo prima, altrimenti il taxi dovrà aspettare a lungo). Inoltre, ci verrà assegnare un taxi che consegnerà il suo ultimo passeggero in un nodo del grafo di Roma abbastanza vicino al nodo corrispondente al primo cliente del ciclo di ritorno. Allora, se chiamiamo con $A = \{A_1, A_2, \dots, A_r\}$ l'insieme dei cicli hamiltoniani che corrispondono a taxi provenienti dall'areoperto di fiumicino che possono essere assegnati ai cicli di ritorno $\{H_1, H_2, \dots, H_q\}$, si tratterà innanzitutto di definire un insieme di pesi p_{ij} che rappresentino il vantaggio di assegnare il taxi che serve A_i al percorso di ritorno H_j . Ad esempio, se d_{ij} è la distanza fra l'ultimo passeggero di A_i e il primo passeggero di H_j , potremmo porre $p_{ij} = -d_{ij}$. Potremmo aggiungere nel peso anche un fattore relativo al tempo di attesa - minori tempi di attesa per il conducente sono da preferire). A questo punto vogliamo scegliere come assegnare ogni vettura che arriva dall'areoperto a un ciclo verso l'areoperto. Costruiamo quindi un grafo bipartito $G = (V, E)$ ove l'insieme dei nodi sarà $V = H \cup A$, ed esiste un arco fra in nodo $H_j \in H$ e il nodo $A_i \in A$ se e solo se l'ultimo cliente di A_i viene lasciato in tempo utile per prelevare il primo cliente di H_j . Inoltre, associamo a ogni arco $(i, j) \in E$ il peso p_{ij} che rappresenta il vantaggio di assegnare il taxi del ciclo A_j al ciclo di ritorno H_i . E' facile convincersi che cercare l'assegnamento migliore corrisponde a risolvere un problema di accoppiamento massimo nel grafo G .