

# INTRODUZIONE ALLA RICERCA OPERATIVA

Lecture Notes, v. 7.0  
27 ottobre 2023

Daniele Vigo<sup>1</sup>  
Marco Antonio Boschetti<sup>2</sup>

Alma Mater Università di Bologna

<sup>1</sup> Dip. di Ingegneria dell’Energia Elettrica e  
dell’Informazione “Guglielmo Marconi” e CIRI-ICT

<sup>2</sup> Dip. di Matematica



# Indice

<b>1</b>	<b>Introduzione alla Ricerca Operativa</b>	<b>1</b>
1.1	La Ricerca Operativa . . . . .	1
1.2	Sistemi organizzati e problemi decisionali . . . . .	2
1.3	Problemi, Modelli e Algoritmi . . . . .	3
1.4	Approssimazione di un modello . . . . .	6
1.5	Complessità computazionale . . . . .	9
1.6	Problemi vs Algoritmi . . . . .	11
1.7	Ottimizzazione non vincolata . . . . .	12
1.7.1	Esempio: Domanda e Ricavo . . . . .	12
1.8	Ottimizzazione vincolata . . . . .	14
1.8.1	Esempio: Miscelazione . . . . .	14
1.8.2	Esempio: Flussi di Traffico . . . . .	15
1.8.3	Esempio: Ridistribuzione auto a noleggio . . . . .	16
<b>2</b>	<b>Programmazione Matematica</b>	<b>21</b>
2.1	Problemi di ottimizzazione matematica . . . . .	21
2.2	Classificazione dei problemi di ottimizzazione . . . . .	24
2.2.1	Problemi di Programmazione Non Lineare (PNL) . . . . .	25
2.2.2	Problemi di Programmazione Convessa (PC) . . . . .	25
2.2.3	Problemi di Programmazione Lineare (PL) . . . . .	27
2.2.4	Problemi di Programmazione Lineare Intera (PLI) . . . . .	28
2.2.5	Problemi di Programmazione Lineare Mista (PLM) . . . . .	29
2.2.6	Esempio 1: Problema lineare continuo . . . . .	29
2.2.7	Esempio 2: Problema lineare intero . . . . .	30
2.2.8	Esempio 3: Problema nonlineare continuo . . . . .	31
<b>3</b>	<b>Programmazione Lineare</b>	<b>33</b>
3.1	Formulazione di problemi PL . . . . .	34
3.1.1	Problema di produzione di sedie . . . . .	34
3.1.2	Problema di produzione di vasche per idromassaggio . . . . .	36
3.1.3	Problemi di mix di produzione . . . . .	37
3.1.4	Problema della dieta . . . . .	38
3.2	Forme dei Problemi PL . . . . .	39
3.2.1	Trasformazione della funzione obiettivo da max a min . . . . .	40

3.2.2	Trasformazione di disequazioni in equazioni . . . . .	41
3.2.3	Trasformazione di equazioni in disequazioni . . . . .	41
3.2.4	Trasformazione di variabili libere . . . . .	42
3.2.5	Esempio di trasformazione di forma . . . . .	42
3.2.6	Equivalenza tra le forme di PL . . . . .	44
3.3	Interpretazione Geometrica dei Problemi PL . . . . .	44
3.3.1	Disegno della Regione Ammissibile di un Problema PL . .	44
3.3.2	Poliedri e vertici . . . . .	47
3.3.3	Vincoli Ridondanti . . . . .	49
3.3.4	Risoluzione Grafica di un problema PL . . . . .	49
3.3.5	Considerazioni sulla soluzione ottima di un PL . . . . .	51
3.4	Assunzioni implicite della PL <b>NEW: (Ver 3.0)</b> . . . . .	54
3.4.1	Proporzionalità . . . . .	54
3.4.2	Additività . . . . .	55
3.4.3	Divisibilità . . . . .	56
3.4.4	Certezza . . . . .	56
<b>4</b>	<b>L'Algoritmo del Simplex</b> . . . . .	<b>57</b>
4.1	Versione intuitiva dell'algoritmo del simplex . . . . .	57
4.2	Versione algebrica dell'algoritmo del simplex . . . . .	60
4.2.1	Soluzioni del problema PL in forma standard . . . . .	61
4.3	Versione algebrica dell'algoritmo . . . . .	68
4.4	Test di Ottimalità . . . . .	69
4.5	Spostamento da SBA a SBA (Pivoting) . . . . .	72
4.6	Soluzioni base degeneri . . . . .	77
4.6.1	Interpretazione geometrica della degenerazione . . . . .	78
4.6.2	Regola di Bland . . . . .	79
4.7	Determinazione della base iniziale: Metodo delle due fasi . . . . .	80
4.7.1	Fase 1 . . . . .	80
4.7.2	Fase 2 . . . . .	82
4.8	Versione completa dell'algoritmo del simplex . . . . .	82
4.9	Esempi di applicazione dell'algoritmo del simplex . . . . .	83
4.9.1	Esempio 1 . . . . .	83
4.10	La forma tableau . . . . .	87
4.10.1	Inversa della base ricavabile dal tableau <b>NEW: (Ver 5.0)</b> .	92
4.10.2	Algoritmo del simplex in forma tableau . . . . .	93
4.10.3	Esempio di risoluzione in forma tableau con solo Fase 2 .	94
4.10.4	Esempio di risoluzione in forma tableau con Fase 1 . . . . .	97
<b>5</b>	<b>Analisi di sensitività per PL</b> . . . . .	<b>101</b>
5.1	Definizione Analitica degli intervalli <b>NEW: (Ver. 4.0)</b> . . .	105
5.1.1	Variazione di un termine noto . . . . .	106
5.1.2	Variazione del costo di una variabile fuori base . . . . .	107
5.1.3	Variazione del costo di una variabile base . . . . .	107
5.1.4	Analisi di sensitività dal tableau ottimo . . . . .	108

<b>6 Esercizi di Programmazione Lineare</b>	<b>111</b>
Esercizio 6.1. Produzione di fertilizzanti . . . . .	111
Esercizio 6.2. Miscelazione di mangimi . . . . .	114
Esercizio 6.3. . . . .	117
Esercizio 6.4. Mix di Produzione . . . . .	120
Esercizio 6.5. . . . .	123
Esercizio 6.6. . . . .	126
Esercizio 6.7. Miscelazione di caffè . . . . .	129
Esercizio 6.8. Mix di produzione parametrico . . . . .	132
Esercizio 6.9. Vincolo ridondante in Fase 1 . . . . .	135
Esercizio 6.10. Problema illimitato . . . . .	138
Esercizio 6.11. . . . .	140
Esercizio 6.12. . . . .	145
Esercizio 6.13. . . . .	145
Esercizio 6.14. . . . .	145
<b>7 Programmazione Lineare Intera</b>	<b>147</b>
7.1 Difficoltà dei problemi PLI . . . . .	148
7.1.1 Proprietà dei problemi PLI . . . . .	148
7.2 Formulazioni equivalenti di PLI . . . . .	150
<b>8 Modelli di Programmazione Lineare Intera NEW: ((Ver. 6.0))</b>	<b>155</b>
8.1 Introduzione . . . . .	155
8.2 Indagine di Mercato . . . . .	156
8.3 Noleggio di Dispositivi . . . . .	157
8.4 Campagna pubblicitaria . . . . .	158
8.5 Turnazione del Personale . . . . .	160
8.6 Il problema dello zaino (Knapsack Problem) . . . . .	161
8.6.1 Varianti del problema . . . . .	163
8.7 Problema del Knapsack Multiplo . . . . .	168
8.8 Problema del Bin Packing . . . . .	170
8.8.1 Varianti del problema . . . . .	173
8.9 Problema del Set Covering . . . . .	174
8.10 Problema del Set Partitioning . . . . .	177
8.11 Problema del Set Packing . . . . .	179
8.12 Problema dell'Assegnamento . . . . .	181
8.12.1 Problema dell'Assegnamento Generalizzato . . . . .	181
8.13 Problemi di facility location . . . . .	182
8.14 Problemi di scheduling . . . . .	185
8.15 Problema del lot sizing . . . . .	187
8.16 Problema del fixed charge . . . . .	188
8.17 Disattivazione di vincoli . . . . .	189

<b>9 Esercizi: modelli di Programmazione Lineare Intera NEW: ((Ver. 6.0))</b>	<b>191</b>
Esercizio 9.15. Taglio di una barra. . . . .	191
Esercizio 9.16. Assemblaggio elettrodomestici. . . . .	193
Esercizio 9.17. Caricamento di camion. . . . .	194
Esercizio 9.18. Localizzazione ambulatori. . . . .	196
Esercizio 9.19. Assegnamento di lavori. . . . .	198
Esercizio 9.20. Assegnamento di lavori 2. . . . .	199
Esercizio 9.21. Localizzazione magazzini. . . . .	200
Esercizio 9.22. Test software. . . . .	201
Esercizio 9.23. Mix di produzione. . . . .	202
Esercizio 9.24. Posizionamento prodotti. . . . .	204
Esercizio 9.25. Assegnamento posti barca. . . . .	205
Esercizio 9.26. Localizzazione vigili del fuoco. . . . .	207
Esercizio 9.27. . . . .	208
Esercizio 9.28. . . . .	209
Esercizio 9.29. Assegnamento lavori. . . . .	210
Esercizio 9.30. Piano lavorazione. . . . .	212
Esercizio 9.31. Flusso massimo. . . . .	213
Esercizio 9.32. Assegnamento con conflitti. . . . .	214
<b>10 Algoritmi per PLI</b>	<b>215</b>
10.1 Introduzione . . . . .	215
10.2 Algoritmo Cutting Plane NEW: (Ver. 6.0) . . . . .	215
10.3 L'Algoritmo Branch and Bound . . . . .	217
<b>11 Esercizi di Programmazione Lineare Intera</b>	<b>229</b>
Esercizio 11.1. . . . .	229
Esercizio 11.2. . . . .	234
<b>12 Risoluzione di problemi con Excel</b>	<b>243</b>
<b>13 Teoria dei Grafi</b>	<b>251</b>
13.1 Introduzione . . . . .	251
13.1.1 Applicazioni . . . . .	253
13.1.2 Taglio di un grafo . . . . .	254
13.1.3 Cammini, circuiti e cicli . . . . .	255
13.1.4 Grafi parziali, sottografi e componenti . . . . .	258
13.1.5 Alberi . . . . .	259
13.1.6 Rappresentazione dei grafi . . . . .	260
13.2 NEW: (Cammini minimi) . . . . .	264
13.2.1 Formulazione matematica . . . . .	265
13.2.2 Assunzioni . . . . .	265
13.2.3 Distance label . . . . .	266
13.2.4 Condizioni di ottimalità . . . . .	267
13.2.5 Algoritmi Label Correcting . . . . .	268

13.2.6 Algoritmo di Dijkstra . . . . .	269
13.2.7 Algoritmo di Floyd-Warshall . . . . .	274
13.3 NEW: (Alberi di copertura) . . . . .	279
13.3.1 Applicazioni . . . . .	279
13.3.2 Formulazione matematica . . . . .	280
13.3.3 Condizioni di ottimalità . . . . .	280
13.3.4 Algoritmo di Prim-Dijkstra . . . . .	283
13.3.5 Algoritmo di Kruskal . . . . .	286
13.4 NEW: (Flusso massimo) . . . . .	290
13.4.1 Formulazione matematica . . . . .	293
13.4.2 Assunzioni e definizioni . . . . .	294
13.4.3 Condizioni di ottimalità . . . . .	295
13.4.4 Algoritmo di Ford-Fulkerson . . . . .	297
13.5 NEW: (Flusso di costo minimo) . . . . .	302
13.5.1 Assunzioni e definizioni . . . . .	304
13.5.2 Condizioni di ottimalità . . . . .	305
13.5.3 Cycle Cancelling Algorithm . . . . .	305
13.5.4 Successive Shortest Path Algorithm . . . . .	306
13.5.5 Primal-Dual Algorithm . . . . .	308
13.5.6 Out-of-Kilter Algorithm . . . . .	310
13.5.7 Algoritmi polinomiali . . . . .	313
13.5.8 Algoritmi Fortemente Polinomiali . . . . .	314



# Capitolo 1

## Introduzione alla Ricerca Operativa

### 1.1 La Ricerca Operativa

La Ricerca Operativa è una disciplina relativamente recente che mette a disposizione metodi scientifici *quantitativi*, basati sulla definizione di strumenti modellistici ed algoritmici, per la formulazione e la risoluzione di *problemi decisionali* per sistemi organizzati complessi. Il termine italiano usato per la denominazione della disciplina non chiarisce molto il suo ambito di applicazione, mentre il termine anglosassone Operations, che significa “attività”, specifica con maggiore chiarezza che la disciplina si occupa della ricerca sulla pianificazione delle attività. Per questo motivo la ricerca operativa è anche denominata in vari ambiti Scienza delle Decisioni (Decision Science) o Scienza della Gestione (Management Science). Più recentemente la ricerca operativa è considerata una disciplina fondamentale dell’Intelligenza Artificiale e più in particolare dell’analytics ed è nota come *prescriptive analytics* proprio per la capacità di fornire metodi in grado di prescrivere come ottenere da un sistema il comportamento desiderato.

Le origini della ricerca operativa sono generalmente fatte risalire alla seconda guerra mondiale<sup>1</sup>, quando alcuni gruppi di lavoro interdisciplinari, denominati Research on Military Operations (da cui discende il nome anglosassone), furono costituiti per risolvere problemi pratici quali la localizzazione delle stazioni radar o la pianificazione delle attività logistiche. In realtà molti dei metodi della ricerca operativa, quali ad esempio i metodi per la risoluzione dei sistemi lineari, hanno una origine molto precedente ma la loro sistematizzazione e generalizzazione negli strumenti che discuteremo in questo testo è più recente ed è dovuta al

---

<sup>1</sup>Per un approfondimento sulle origini di veda <https://www.informs.org/Explore/History-of-O.R.-Excellence/Bibliographies/The-Origins-of-OR#:~:text=The%20term%20operational%20research%22%20was,into%20Royal%20Air%20Force%20tactics>.

lavoro di alcuni studiosi, tra cui George B. Dantzig (1914-2005), che nel secondo dopoguerra hanno gettato le basi della disciplina come la conosciamo.

## 1.2 Sistemi organizzati e problemi decisionali

Un *sistema organizzato complesso* può essere definito come un insieme di elementi tra loro legati da forme di interazione. Un tipico esempio di tali sistemi è costituito da un reparto produttivo, in cui gli elementi sono le macchine, le stazioni di lavoro o i sistemi di movimentazione e stoccaggio, che interagiscono tra loro attraverso i flussi di materiali e di pezzi nelle varie fasi di lavorazione (vedi figura 1.1).

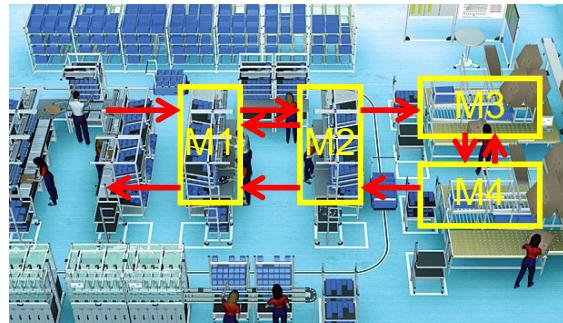


Figura 1.1: Esempio di sistema organizzato complesso: un reparto di produzione.

Un altro esempio tipico di sistema organizzato è rappresentato da un sistema logistico di distribuzione delle merci in cui gli elementi sono le fabbriche i depositi ed i clienti e l'interazione è rappresentata dal flusso delle merci.

Un *problema decisionale* consiste nella scelta, tra diverse alternative, della combinazione di un insieme di decisioni, ossia una soluzione, che consente di ottenere dal sistema le prestazioni desiderate.

Facendo riferimento al reparto di produzione di figura 1.1, gli elementi del sistema sono rappresentati dalle macchine e dai sistemi di movimentazione e stoccaggio che interagiscono per la produzione di pezzi attraverso il flusso di questi. In tale contesto possono essere immaginati diversi tipi di decisioni:

- Decisioni di tipo *strategico*, ossia aventi un orizzonte temporale di applicazione lungo (molti anni): ad esempio, la struttura dell'impianto, la tipologia ed il numero di macchinari da impiegare.
- Decisioni di tipo *tattico*, ossia aventi un orizzonte temporale di applicazione medio (da alcuni mesi ad uno o due anni): ad esempio, la politica di approvvigionamento delle materie prime, la definizione dei periodi di manutenzione, l'assegnazione del personale ai diversi reparti.
- Decisioni di tipo *operativo*, ossia aventi un orizzonte temporale di applicazione breve (da alcuni giorni a pochi mesi): ad esempio, la sequenza di

lavorazione dei pezzi sulle macchine, la formazione dei turni di lavoro, la gestione delle scorte per la produzione.

I tipi di decisione sopraelencati richiedono informazioni su cui basare le stesse che hanno natura e qualità molto diverse. Ad esempio, in un problema strategico quale quello della progettazione di un impianto occorre tenere presente della valutazione della domanda potenziale dei beni da produrre in un orizzonte temporale molto lungo affinché il dimensionamento sia adeguato alla domanda futura da soddisfare o sul costo futuro delle materie prime. È evidente che tali informazioni sono frutto di previsioni che per loro natura non hanno carattere deterministico e quindi sono affette da possibili errori anche consistenti. In un problema operativo invece le informazioni sono normalmente disponibili con maggiore precisione ed affidabilità. Ad esempio i quantitativi dei pezzi da produrre sono noti dagli ordinativi dei clienti, le scorte di materie prime ed il personale disponibile sono anch'essi noti con precisione.

Le decisioni da prendere devono considerare un criterio di valutazione che permetta di valutarne l'effetto rispetto ad una misura di prestazione. Le misure di prestazione maggiormente utilizzate riguardano:

- la minimizzazione dei costi (di investimento, di produzione, ecc...);
- la massimizzazione del margine economico;
- la massimizzazione dell'efficienza, ad esempio misurata attraverso il volume di produzione, il tempo di lavorazione, il tempo di soddisfacimento delle richieste dei clienti, ecc...

Spesso i criteri, o *obiettivi*, con cui valutare le decisioni sono molteplici ed a volte sono contrastanti tra loro. Ad esempio nella scelta di un macchinario non è certamente il costo o la velocità di lavorazione il solo criterio che si adotta per la decisione e spesso macchine più veloci sono anche più costose. In tali casi le decisioni da prendere devono trovare un compromesso tra i diversi criteri di scelta adottati. Sebbene esistano metodologie che permettono di trattare problemi che richiedano di tenere conto di diversi obiettivi, noti come *problem multi-obiettivo*, in pratica spesso questi vengono combinati in una funzione di prestazione complessiva in cui i diversi obiettivi vengono pesati mediante coefficienti di peso che ne riflettono l'importanza per il decisore.

Nel seguito faremo principalmente riferimento a problemi decisionali di tipo operativo, che per la loro definizione possono contare su informazioni di tipo deterministico note con precisione, e caratterizzate dall'ottimizzazione di un singolo obiettivo (problem *mono-obiettivo*), eventualmente ottenuto combinando tra loro obiettivi contrastanti.

### 1.3 Problemi, Modelli e Algoritmi

Per sviluppare una soluzione per il supporto alle decisioni per ottenere da un sistema le prestazioni desiderate, sono necessari i seguenti passi:

- modellare matematicamente il problema che si vuole risolvere;
- identificare la sua complessità;
- progettare l'algoritmo più adatto, usando le tecniche di soluzione in relazione al modello e alla complessità del problema.

Nei prossimi capitoli mostreremo come problemi molto semplici da formulare possono risultare molto difficili da risolvere e come problemi molto complessi da descrivere possono essere in realtà molto facili da risolvere.

Prima di proseguire è necessario definire che cos'è un *problema*, un *modello* e un *algoritmo*.

**Problema, [pro-blè-ma] s.m. (pl. -mi)**

**Garzanti:** “*quesito con cui si chiede di trovare, con un procedimento di calcolo, uno o più dati sconosciuti, partendo dai dati noti contenuti nell'enunciato del quesito stesso.*”

**Coletti:** “*in matematica e in altre scienze, domanda con cui si chiede di trovare, sulla base di dati noti ed enunciati, dati non noti, logicamente deducibili dai primi.*”

Figura 1.2: Definizione di problema

In figura 1.2 sono riportate due definizioni fornite da due noti dizionari della lingua italiana. Come si può notare, aldilà delle diverse sfumature, le due definizioni concordano sul fatto che un problema chiede di trovare attraverso un procedimento di calcolo dei *dati non noti* a partire dai *dati noti* e dagli enunciati che lo definiscono. Chiaramente i dati noti corrispondono all'input del problema, i dati non noti all'output atteso, mentre il procedimento di calcolo altro non è che un algoritmo in grado di risolvere il problema.

E' importante sottolineare che i dati noti rappresentano una *istanza* del problema. Per esempio, calcolare l'area di un triangolo è il problema, mentre il triangolo di altezza 20 cm e base 30 cm è una istanza.

Nel nostro contesto per risolvere un problema è necessario costruire un modello e definire un algoritmo di soluzione.

Il concetto di algoritmo è ormai usato quotidianamente, ma vale la pena ricordare la sua definizione. In figura 1.3 è riportata la definizione di algoritmo fornita da Treccani, che ci ricorda che un algoritmo è una procedura definita in modo preciso per risolvere una classe di problemi.

Per determinare il problema e un algoritmo per risolverlo dobbiamo definire un modello. I modelli sono una rappresentazione, spesso semplificata, di concetti, fenomeni, relazioni, strutture, processi, sistemi, etc. In generale, i modelli possono riguardare sia aspetti puramente teorici che del mondo reale e possono essere di tipo verbale, grafico, fisico, matematico, etc. I modelli consentono di perseguire i seguenti obiettivi:

- facilitare la comprensione, identificando le componenti fondamentali e i meccanismi di funzionamento;

**Algoritmo (da Treccani):**

"*termine, derivato dall'appellativo al-Khuwārizmī ("originario della Corasmia") del matematico Muhammad ibn Mūsā del 9° sec., che designa qualunque schema o procedimento sistematico di calcolo (per es. l'a. euclideo, delle divisioni successive, l'a. algebrico, insieme delle regole del calcolo algebrico ecc.). Con un algoritmo si tende a esprimere in termini matematicamente precisi il concetto di procedura generale, di metodo sistematico valido per la soluzione di una certa classe di problemi.*"

Figura 1.3: Definizione di algoritmo

- spiegare, controllare e predire eventi, anche sulla base delle osservazioni passate;
- supportare i processi decisionali.

Molto spesso ciò che deve essere rappresentato nel modello può essere composto da molte componenti, le cui interrelazioni possono essere anche molto complesse. In questi casi è necessario applicare delle semplificazioni al modello, che però non devono "interferire" con l'utilizzo che se ne vuole fare. Il processo di semplificazione deve identificare le componenti di interesse e definire eventuali approssimazioni.

Lo strumento di modellazione a cui siamo interessati è il *modello matematico*, che consente di rappresentare la realtà di interesse con un alto grado di astrazione per mezzo di simboli, equazioni, etc. Per poter scrivere i modelli matematici e sapere come analizzarli e risolverli è necessario conoscere il linguaggio e i metodi della matematica.

I modelli matematici che utilizzeremo prevedono l'uso di parametri che rappresentano l'input (cioè i dati noti) e di variabili che rappresentano l'output (cioè i dati non noti). Nel caso in cui i parametri possono essere definiti con esattezza a priori parleremo di *modelli deterministici*, mentre nel caso in cui i parametri non sono certi a priori parleremo di *modelli stocastici*. In molte applicazioni, come in economia e finanza, molto spesso i modelli sono di tipo stocastico. I modelli di tipo stocastico sono solitamente difficili da risolvere e spesso ci si riporta a modelli deterministici, ad esempio enumerando sottoinsiemi di scenari rappresentativi della realtà di interesse.

Nella definizione e nell'uso dei modelli non bisogna dimenticarsi dell'aspetto numerico, perché l'approssimazione dei parametri e gli errori nel calcolo possono inficiare la validità dell'output. La matematica permette di prevedere anche la precisione con cui sarà fornito l'output. Di questo argomento se ne occupa l'analisi numerica.

## 1.4 Approssimazione di un modello

Quando si costruisce un modello è importante considerare dove possono e devono essere applicate delle approssimazioni. In particolare, si devono considerare le seguenti approssimazioni:

- Prima del calcolo:
  - modello;
  - misure empiriche;
  - precedenti calcoli.
- Durante il calcolo:
  - troncamento o discretizzazione;
  - arrotondamento.

L'accuratezza del risultato finale dipenderà da tutte le approssimazioni che sono state applicate e l'errore complessivo sarà dovuto alla combinazione degli effetti delle diverse approssimazioni. L'approssimazione durante il calcolo spesso è inevitabile, ma può essere gestita e mitigata, anche scegliendo opportunamente gli algoritmi e le modalità di implementazione.

*L'errore di troncamento* è dato dalla differenza tra il risultato “esatto” e quello prodotto dall'algoritmo. Per esempio, può essere dovuto alla terminazione di un algoritmo di soluzione di tipo iterativo prima del raggiungimento del valore “finale”.

*L'errore di arrotondamento* è dato dalla differenza tra il risultato prodotto da un dato algoritmo usando un'aritmetica esatta e il risultato prodotto dallo stesso algoritmo usando un'aritmetica approssimata. L'aritmetica del “computer” è approssimata a causa dell'inesatta rappresentazione dei numeri reali e dall'applicazione degli operatori aritmetici ad essi.

**Esempio 1** Il calcolo della superficie della terra usando la formula  $A = 4\pi r^2$  implica le seguenti approssimazioni: la terra viene modellata come una sfera; il valore del raggio  $r$  è dato da misure empiriche e precedenti calcoli (come?); il valore di  $\pi$  richiede un troncamento; i valori dei dati in input e i risultati delle operazioni aritmetiche sono arrotondati dal computer.

Nella figura 1.4 è illustrato come Eratostene (Cirene, 276 A.C. – Alessandria d'Egitto, 194 A.C.) riuscì a stimare con buona precisione il raggio terrestre. Nel modello di Eratostene per stimare il raggio della terra furono applicate le seguenti approssimazioni: la terra era una sfera, i raggi del sole erano considerati paralleli, le località di Aswan e Alessandria furono considerate allineate lungo lo stesso meridiano, la distanza tra le due località era una stima fatta con i mezzi del tempo. Nonostante questo la stima fornita da Eratostene era molto prossima alla stima che siamo in grado di calcolare oggi.

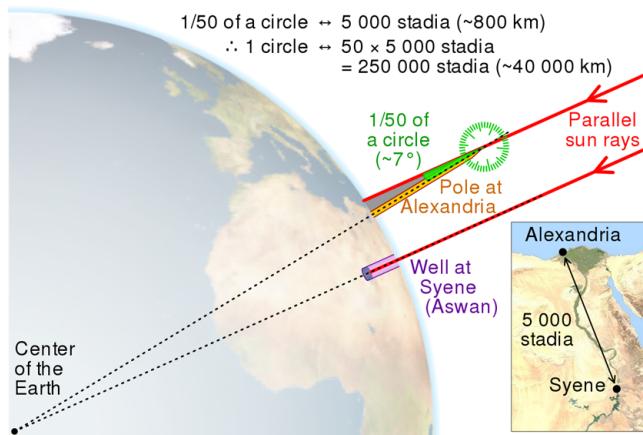


Figura 1.4: Il calcolo del raggio terrestre effettuato da Eratostene (Fonte: <https://en.wikipedia.org/wiki/Eratosthenes>).

**Esempio 2** Gli errori di calcolo possono “propagarsi” in modo inaspettato e a volte disastroso, fino al punto di rendere il risultato inutilizzabile. Un esempio molto semplice può essere ottenuto utilizzando un “foglio elettronico”, uno strumento di lavoro molto usato in ambito scientifico/economico. Si può provare a calcolare con Excel (o una applicazione analoga) la seguente “formula ricorsiva”:

- Passo base:  $x_0 = 0.2$
- Passo ricorsivo:  $x_{i+1} = 11x_i - 10x_0$

Per induzione possiamo dedurre analiticamente che  $x_i = x_0$  per ogni  $i$ :

- Passo base:  $x_1 = 11x_0 - 10x_0 = x_0$
- Passo ricorsivo: se è vero fino a  $i$ , allora  $x_{i+1} = 11x_0 - 10x_0 = x_0$

Ma cosa accade se la formula ricorsiva è calcolata usando Excel? Si riesce ad ottenere che  $x_i = x_0 = 0.2$  per ogni  $i$ ?

Per rispondere a queste domande basta implementare la formula ricorsiva in Excel come mostrato in figura 1.5. Inizialmente non sembrano esserci errori, ma se continuiamo a scorrere i risultati, come illustrato in figura 1.6, scopriamo che la situazione peggiora piuttosto rapidamente fino a raggiungere dei risultati inaccettabili ben prima delle 20 iterazioni.<sup>2</sup>

---

<sup>2</sup>questo esempio viene proposto da molti anni e tutti gli anni prima di proporlo viene verificato se il comportamento di Excel rimane lo stesso. Con sorpresa, nonostante sia un caso noto, tutti gli anni si conferma il comportamento illustrato.

A	B	C	D	E	F	G
1	Iterazioni	X(i+1)				
2	1	0,2 <= Basso base: valore costante X(0)=0,2				
3	2	0,2 <= Passo ricorsivo: X(i+1) = 11 X(i) - 10 X(0).				
4	3	0,2				
5	4	0,2				
6	5	0,2				
7	6	0,2				
8	7	0,2				
9	8	0,2				

Figura 1.5: Esempio di implementazione della formula ricorsiva in Excel.

A	B	C	D	E	F	G	H	I	J	K	L	M
1	Iterazioni	X(i+1)										
2	1	0,2 <= Basso base: valore costante X(0)=0,2										
3	2	0,2 <= Passo ricorsivo: X(i+1) = 11 X(i) - 10 X(0).										
4	3	0,2										
5	4	0,2										
6	5	0,2										
7	6	0,2										
8	7	0,2										
9	8	0,2										
10	9	0,200000003										
11	10	0,200000038										
12	11	0,200000419										
13	12	0,200004607										
14	13	0,2000050682										
15	14	0,200057497										
16	15	0,206132466										
17	16	0,267457121										
18	17	0,942028336										
19	18	8,362311691										
20	19	89,9854286										
21	20	987,8397146										
22	21	10864,23686										
23	22	119504,6055										
24	23	1314548,66										
25	24	14460033,26										

Figura 1.6: Errore generato da Excel nel calcolo della formula ricorsiva.

**Esempio 3** Consideriamo il problema della soluzione di un sistema di equazioni lineari:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m \end{cases}$$

Una particolare istanza può essere:

$$\begin{cases} x_1 + x_2 = 3 \\ x_1 - x_2 = 1 \end{cases}$$

Questo sistema lineare ha un'unica soluzione che rappresenta l'intersezione delle rette rappresentate dalle due equazioni. Questo problema è semplice, ma per alcune istanze può essere difficile ottenere una soluzione sufficientemente accurata.

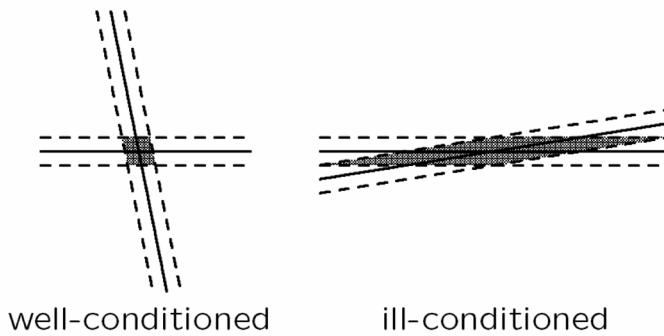


Figura 1.7: Esempio di istanze ben-condizionate e mal-condizionate.

In figura 1.7 è illustrato graficamente come più le rette tendono a essere parallele e più vengono amplificati gli errori. In questo caso, in gergo si dice che peggiora il “condizionamento” dell’istanza del problema.

Il malcondizionamento che si riscontra in alcune istanze nella soluzione dei sistemi lineari potrebbe emergere anche nella soluzione di problemi di programmazione lineare che saranno presentati nei prossimi capitoli.

## 1.5 Complessità computazionale

Finora si è parlato genericamente di problemi di supporto alle decisioni, ma in realtà è necessario distinguere tra le seguenti tipologie di problemi:

- Problema di “Decisione” (o problema decisionale): data un’istanza si vuole determinare se esiste o meno una certa soluzione;
- Problema di “Ricerca”: data un’istanza si vuole determinare una possibile soluzione;
- Problema di “Enumerazione”: data un’istanza si vuole determinare tutte le possibili soluzioni;

- Problema di “Ottimizzazione”: data un’istanza si vuole determinare la “migliore soluzione” possibile rispetto ad una misura fissata.

**Esempio** Un fattorino che svolge le consegne per un corriere nel rispetto di alcuni vincoli ha bisogno di risolvere:

- un problema di decisione, se deve determinare se può riuscire a fare tutte le consegne assegnate entro la giornata (cioè ha bisogno di sapere se esiste una soluzione);
- un problema di ricerca, se deve determinare un modo per svolgere tutte le consegne entro la giornata senza considerare la qualità del giro di consegne (cioè ha bisogno solo di una soluzione ammissibile);
- un problema di enumerazione, se vuole determinare tutti i modi possibili con cui è possibile svolgere tutte le consegne (cioè ha bisogno di tutte le soluzioni ammissibili);
- un problema di ottimizzazione, se deve determinare il modo migliore per svolgere tutte le consegne (cioè ha bisogno della soluzione migliore).

La teoria della complessità computazionale è un ambito di ricerca della matematica, oggi spesso denotato come “informatica teorica”, che ha l’obiettivo di classificare i problemi secondo la loro intrinseca complessità. Banalizzando, la teoria della complessità consente di determinare se un problema è “facile” o “difficile”.

La difficoltà di un problema è una sua caratteristica generale che non è associata a una particolare istanza. Infatti, si ricordi che un problema è un’entità astratta, mentre un’istanza è un caso particolare. Quindi, un algoritmo risolve un problema se è in grado di generare una soluzione per ogni possibile istanza. Se un problema è semplice si può garantire la sua soluzione in un tempo e una quantità di risorse (e.g., memoria) “ragionevole”. Se un problema è difficile non si può garantire la sua soluzione.

Semplificando un argomento in verità molto complesso, si potrebbe dire che:

- I problemi semplici sono quelli polinomiali (insieme **P**).
- I problemi difficili sono quelli “non-polynomiali”, tra questi vi sono i problemi che si trovano nell’insieme **NP** che a sua volta comprende i problemi **NP-Completi**.

**NP** significa Nondeterministic Polynomial time, in quanto **NP** è l’insieme dei problemi decisionali in cui se la risposta è “Si” lo si può verificare in tempo polinomiale utilizzando una macchina di Turing deterministica, o in modo equivalente possiamo dire che il problema può essere risolto in tempo polinomiale da una macchina di Turing non-deterministica. Al momento possiamo facilmente dimostrare che  $\mathbf{P} \subseteq \mathbf{NP}$ , ma non sappiamo se  $\mathbf{P} \neq \mathbf{NP}$ .

Un problema decisionale  $f : I \leftarrow \{0, 1\}$  è *riducibile polinomialmente* a  $g : I \leftarrow \{0, 1\}$ , i.e.  $f \propto g$ , se esiste una funzione  $h$ , calcolabile in tempo polinomiale, tale che per ogni  $x \in I$ :

$$f(x) = g(h(x))$$

Un problema decisionale  $p : I \leftarrow \{0, 1\}$  è definito **NP**-Completo se e solo se:

- $p \in \mathbf{NP}$ ;
- per ogni  $f \in \mathbf{NP}$  si ha  $f \propto p$

Fino ad oggi non è stato dimostrato che l'insieme dei problemi **NP** e l'insieme dei problemi **NP**-Completi sono disgiunti. Se così non fosse, allora  $\mathbf{P} = \mathbf{NP}$ . Il Clay Mathematics Institute<sup>3</sup> lo ha incluso tra i 7 problemi matematici del millennio per i quali è previsto un premio di 1,000,000 di dollari per chi fornirà la dimostrazione corretta per primo.

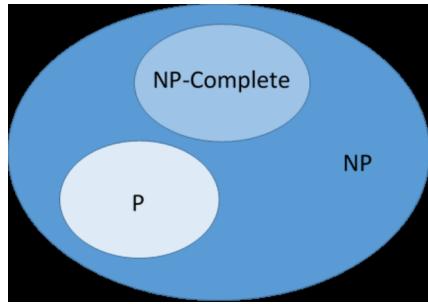


Figura 1.8: Relazione tra problemi **P**, **NP** e **NP**-Completi.

La relazione nota al momento tra problemi **P**, **NP** e **NP**-Completi è rappresentata in figura 1.8.

Un problema di ottimizzazione si definisce **NP-Hard** se esiste un problema **NP**-Completo che può essere ridotto ad esso con trasformazione polinomiale.

## 1.6 Problemi vs Algoritmi

In letteratura può capitare che per un dato problema siano proposti più algoritmi. Questo è dovuto alla naturale evoluzione delle soluzioni proposte dalla ricerca scientifica, ma anche al fatto che l'algoritmo più adatto dipende dalle istanze e dagli specifici obiettivi dell'applicazione finale in termini di prestazioni.

Le prestazioni di un algoritmo possono essere misurate con:

- il *tempo di calcolo* richiesto per ottenere la soluzione;

---

<sup>3</sup><https://www.claymath.org/millennium-problems/>

- la *qualità della soluzione* (cioè quanto il risultato ottenuto approssima il risultato corretto).

Per ogni istanza di un problema esistono algoritmi che forniscono prestazioni migliori di altri. Inoltre, per i problemi difficili c'è un ulteriore opzione:

- *Algoritmi euristici*, che trovano soluzioni di “buona qualità”;
- *Algoritmi esatti*, che trovano le soluzioni “ottime”.

Per la scelta dell'algoritmo è determinante definire la complessità computazionale del problema. Infatti, per i problemi polinomiali la scelta naturale dovrebbero essere quella degli algoritmi esatti, a meno che non ci sia l'esigenza di risolvere istanze (per esempio, di grandi dimensioni) in tempi molto ridotti non sufficienti per ottenere una soluzione esatta, anche se il problema è polinomiale. Mentre, nel caso dei problemi non polinomiali non si può garantire l'ottenimento della soluzione migliore per qualsiasi istanza anche se si decide di utilizzare un algoritmo esatto.

Anche il modello scelto per un dato problema ha un impatto determinante sulla scelta dell'algoritmo più adatto per risolverlo. Infatti, si potrebbe scegliere un modello che approssima maggiormente il problema, ma che può essere risolto in modo molto efficiente/efficace. Mentre, modelli che descrivono in modo molto preciso il problema potrebbero essere troppo difficili da risolvere.

La scelta del modello e dell'algoritmo più adatto è condizionata anche dalla qualità della soluzione che viene richiesta dall'applicazione e dal tempo e dalle risorse di calcolo disponibili. Per esempio, un algoritmo di ottimizzazione utilizzato per pianificare le modalità di taglio di una lastra di metallo da parte di un macchinario ha a disposizione il tempo richiesto per eseguire il taglio pianificato precedentemente, perché tempi di calcolo più lunghi implicherebbero un fermo macchina.

La stragrande maggioranza degli algoritmi di soluzione nell'ambito dell'Intelligenza Artificiale (AI) sono classificabili come algoritmi euristici. Nell'ambito dell'ottimizzazione molti modelli possono essere risolti con algoritmi esatti.

## 1.7 Ottimizzazione non vincolata

Per ottimizzazione non vincolata si intende la ricerca del minimo o massimo di una funzione.

### 1.7.1 Esempio: Domanda e Ricavo

Una casa editrice prevede che l'equazione della domanda relativa alle vendite del suo ultimo romanzo sia:

$$q = -2000p + 150000 \quad (1.1)$$

dove  $q$  è il numero di libri che si prevede di vendere ogni anno se il prezzo di vendita è di  $p$  Euro. L'equazione della domanda rappresenta il modello matematico che descrive l'andamento delle vendite in funzione del prezzo scelto.

Se la casa editrice vuole sapere quale prezzo deve applicare per ottenere il massimo del ricavo annuo, deve definire il modello matematico che descrive il ricavo in funzione del prezzo. Siccome il ricavo è dato dal prodotto tra il prezzo e la quantità venduta:

$$R = qp \quad (1.2)$$

se si sostituisce  $q$  con l'equazione relativa alla domanda diventa:

$$R = qp = (-2000p + 150000)p = -2000p^2 + 150000p \quad (1.3)$$

Il ricavo è in funzione del prezzo ed è dato da:

$$R(p) = -2000p^2 + 150000p \quad (1.4)$$

Per cui, si deve trovare il prezzo  $p$  che massimizza il ricavo  $R(p)$ .

Si noti che (1.3) corrisponde all'equazione di una parabola e (1.4) è una funzione quadratica. Il coefficiente del termine quadratico è  $-2000$ , ossia è negativo, quindi la funzione è concava (vedi figura 1.9). Per trovare l'ottimo possiamo determinare il vertice della parabola o usare il calcolo differenziale applicando le condizioni di ottimalità del primo ordine.

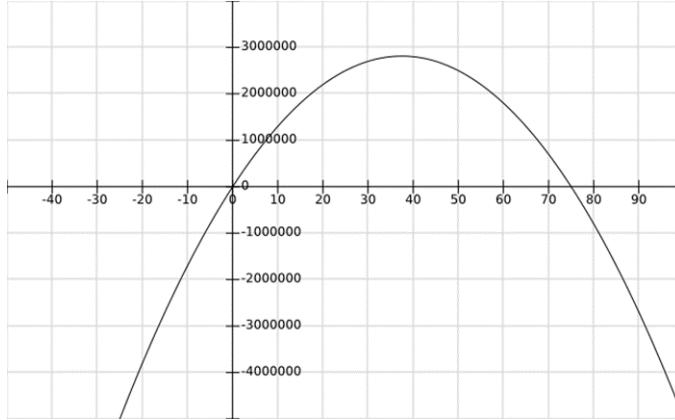


Figura 1.9: Funzione del ricavo.

Applicando le condizioni di ottimalità del primo ordine otteniamo:

$$R'(p) = -4000p + 150000 = 0 \implies p = \frac{150000}{4000} = 37.50 \quad (1.5)$$

Per cui, se si vuole massimizzare i ricavi si deve fissare un prezzo di 37.50 Euro per ogni copia del libro, ottenendo un ricavo pari a:

$$R(37.50) = -2000(37.50)^2 + 150000(37.50) = 2812500 \quad (1.6)$$

## 1.8 Ottimizzazione vincolata

Per ottimizzazione vincolata si intende la ricerca della soluzione di un problema in cui sono previsti dei vincoli che limitano l'insieme delle soluzioni ammissibili.

### 1.8.1 Esempio: Miscelazione

Un'azienda di bibite produce tre miscele di succo le cui ricette prevedono che per produrre ogni litro sia richiesta la seguente composizione:

- PineOrange: 2 parti di succo d'ananas e 2 parti di succo d'arancia;
- PineKiwi: 3 parti di succo d'ananas e 1 parte di succo di kiwi;
- OrangeKiwi: 3 parti di succo d'arancia e 1 parte di succo di kiwi.

L'azienda dispone ogni giorno di 800 parti di succo d'ananas, 650 parti di succo d'arancia e 350 parti di succo di kiwi. Il responsabile della produzione vuole determinare quanti litri di ciascuna miscela deve produrre per utilizzare tutte le materie prime (per esempio, perché è necessario svuotare tutte le cisterne al termine della giornata lavorativa per poter procedere al loro lavaggio).

Per modellare il problema e risolverlo, il primo passaggio è l'identificazione delle variabili decisionali:

- $x_1$ : numero di litri di PineOrange prodotti in un giorno;
- $x_2$ : numero di litri di PineKiwi prodotti in un giorno;
- $x_3$ : numero di litri di OrangeKiwi prodotti in un giorno.

Le informazioni che abbiamo a disposizione sono le seguenti:

	$x_1$	$x_2$	$x_3$	Totale disponibile
Succo Ananas (parti)	2	3	0	800
Succo Arancia (parti)	2	0	3	650
Succo Kiwi (parti)	0	1	1	350

L'insieme dei vincoli determinato dalle tre ricette e dalla quantità di materia prima disponibile origina il seguente sistema lineare:

$$\begin{cases} 2x_1 + 3x_2 = 800 \\ 2x_1 + 3x_3 = 650 \\ \quad x_2 + x_3 = 350 \end{cases}$$

Si può risolvere il sistema lineare con il metodo di eliminazione di Gauss-Jordan oppure uno dei tanti algoritmi alternativi. La soluzione del sistema lineare è:

$$(x_1, x_2, x_3) = (100, 200, 150)$$

che suggerisce di produrre ogni giorno 100 litri di PineOrange, 200 litri di PineKiwi e 150 litri di OrangeKiwi per riuscire a consumare tutte le materie

disponibili. In questo caso la soluzione era unica, quindi non è necessario trovare la soluzione più “conveniente” in base a un qualche criterio.

Nel caso in cui venisse rilassato il vincolo di consumare tutte le materie prime, le equazioni sarebbero sostituite da disequazioni:

$$\begin{cases} 2x_1 + 3x_2 \leq 800 \\ 2x_1 + 3x_3 \leq 650 \\ +x_2 + x_3 \leq 350 \end{cases}$$

Inoltre, bisogna anche garantire che le quantità di prodotto da produrre non siano negative, per cui è necessario aggiungere i cosiddetti vincoli di non negatività delle variabili decisionali, cioè  $x_1, x_2, x_3 \geq 0$ .

Per risolvere il sistema di disequazioni non si possono applicare direttamente le operazioni elementari previste, per esempio, nel metodo come Gauss-Jordan. Se lo si volesse fare, bisogna prima trasformare le disequazioni in equazioni aggiungendo delle *variabili di scarto*  $s_i$  non negative per ogni disequazione  $i \in 1, 2, 3$  (cioè  $s_i \geq 0$ ). Nel caso di esempio potremo avere:

$$\begin{cases} 2x_1 + 3x_2 + s_1 = 800 \\ 2x_1 + 3x_3 + s_2 = 650 \\ +x_2 + x_3 + s_3 = 350 \end{cases}$$

Come si vedrà nel capitolo 3, una volta trasformate le disequazioni in equazioni si potranno usare le operazioni elementari, analoghe a quelle impiegate in metodi come Gauss-Jordan, per trovare le soluzioni del sistema. Si vedrà anche che i vincoli di non negatività applicati alle variabili non complicano la soluzione.

### 1.8.2 Esempio: Flussi di Traffico

Il traffico che attraversa un quartiere del centro di una cittadina rispetta il sistema di sensi unici descritti nella figura 1.10.

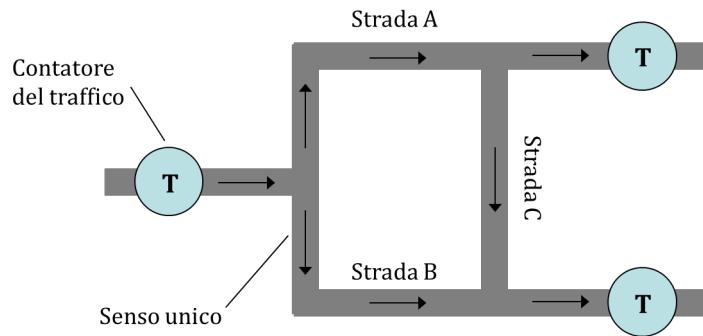


Figura 1.10: Flussi di traffico nel quartiere del centro di una cittadina.

I tre contatori del traffico rilevano i seguenti dati:

- Contatore a Ovest: 200 automobili in entrata ogni ora;
- Contatori a Est: 100 automobili in uscita ogni ora ciascuno.

Si può notare che il numero di automobili entrante nel quartiere (contatore a ovest) è pari al numero di automobili uscenti (somma dei contatori a est). Quindi, si può ipotizzare che il numero di auto che entrano nel quartiere e parcheggiano è compensato dal numero di auto che lasciano i parcheggi ed escono dal quartiere. Inoltre, in una rete stradale si applica il *principio della conservazione del flusso*: il numero di auto entranti in un incrocio è pari al numero delle auto uscenti.

L'amministrazione pubblica della città si chiede se con le informazioni a disposizione è possibile stabilire quante auto transitano ogni ora nelle strade A, B e C rappresentate nella figura 1.10.

Per rispondere al quesito, il primo passaggio è l'identificazione delle variabili decisionali:

- $x_A$ : numero di auto che transitano ogni ora nella Strada A;
- $x_B$ : numero di auto che transitano ogni ora nella Strada B;
- $x_C$ : numero di auto che transitano ogni ora nella Strada C.

Applicando il principio di conservazione del flusso possiamo scrivere le seguenti equazioni che devono essere tutte soddisfatte:

$$\begin{cases} x_A + x_B = 200 \\ x_A - x_C = 100 \\ +x_B + x_C = 100 \end{cases}$$

Se risolviamo il sistema lineare (e.g., con il metodo di eliminazione di Gauss-Jordan) scopriremo che una delle tre equazioni è ridondante (cioè è combinazione lineare delle altre 2). La soluzione del sistema lineare è:

$$\begin{cases} x_A = x_C + 100 \\ x_B = -x_C + 100 \\ x_C = \text{arbitrario} \end{cases}$$

Quindi, i tre contatori non ci permettono di conoscere il flusso esatto sulle tre strade, perché servirebbe conoscere il valore della variabile  $x_C$  (cioè il flusso nella strada C).

### 1.8.3 Esempio: Ridistribuzione auto a noleggio

Un'azienda di noleggio di automobili alla fine di ogni giornata deve bilanciare le auto disponibili nelle sue sedi per essere in grado di rispondere alle richieste dei clienti il giorno dopo.

L'azienda ha quattro sedi in città (vedi figura 1.11) con la seguente situazione:

- Sede A: che ha 20 auto più del necessario;

- Sede B: che necessita di 10 auto in più di quelle di cui dispone;
- Sede C: che ha 15 auto più del necessario;
- Sede D: che necessita di 25 auto in più di quelle di cui dispone.

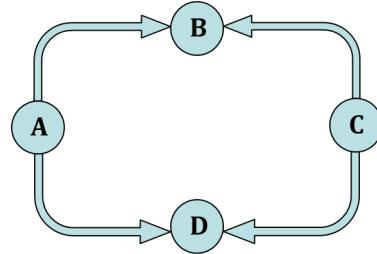


Figura 1.11: Rete per il trasporto di auto tra le diverse sedi.

Per spostare le auto da una sede all'altra si incorre in un costo dovuto al tempo speso da un dipendente alla guida e i costi di carburante e manutenzione per ogni chilometro percorso. I costi per spostare un'auto tra le diverse sedi sono i seguenti:

- Dalla Sede A alla Sede B costa complessivamente 10 Euro;
- Dalla Sede C alla Sede B costa complessivamente 5 Euro;
- Dalla Sede A alla Sede D costa complessivamente 20 Euro;
- Dalla Sede C alla Sede D costa complessivamente 10 Euro.

L'azienda di noleggio ha un budget per gli spostamenti di 475 Euro e vuole sapere se spendendo tutto il budget è possibile spostare le auto tra le 4 sedi come è richiesto.

Per modellare il problema possiamo definire le variabili decisionali:

- $x_{AB}$ : numero di auto spostate dalla Sede A alla sede B;
- $x_{AD}$ : numero di auto spostate dalla Sede A alla sede D;
- $x_{CB}$ : numero di auto spostate dalla Sede C alla sede B;
- $x_{CD}$ : numero di auto spostate dalla Sede C alla sede D;

e la rete dei trasporti può essere rappresentata come in figura 1.12.

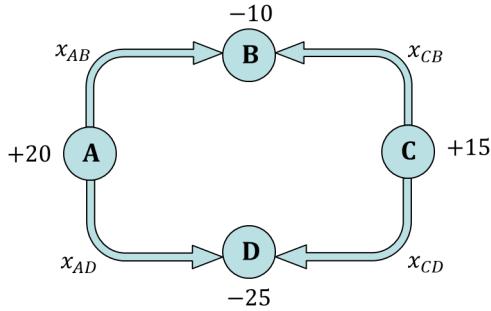


Figura 1.12: Modello della rete per il trasporto di auto tra le diverse sedi.

Le informazioni a disposizione consentono di definire i vincoli del problema che determinano il seguente sistema lineare:

$$\left\{ \begin{array}{rcl} x_{AB} & +x_{AD} & = 20 \\ x_{AB} & +x_{CB} & = 10 \\ & +x_{CB} & +x_{CD} = 15 \\ x_{AB} & +x_{AD} & +x_{CD} = 25 \\ x_{AB} & +x_{AD} & +x_{CB} & +x_{CD} = 475 \end{array} \right.$$

Nonostante il sistema lineare abbia 5 equazioni e 4 variabili, ha una soluzione unica perché uno dei vincoli è ridondante e compatibile con le altre equazioni. La soluzione è  $(x_{AB}, x_{AD}, x_{CB}, x_{CD}) = (5, 15, 5, 10)$ .

La soluzione ottenuta è intera, ma non era scontato e se il budget fosse stato diverso, la soluzione sarebbe cambiata e qualche variabile sarebbe potuta diventare frazionaria o negativa. Nel caso la soluzione fosse stata frazionaria, le auto non possono essere frazionate, per cui la soluzione non sarebbe applicabile. In questo caso sarebbe necessario aggiungere esplicitamente il vincolo di interezza. Nel capitolo 7 vedremo come è possibile farlo.

Se invece di spendere tutto il budget si volesse spostare tutte le auto minimizzando il budget necessario, si dovrebbe eliminare il vincolo di budget (cioè l'ultima equazione) e risolvere il sistema di equazioni lineari così ottenuto. La nuova soluzione sarebbe la seguente:

$$\left\{ \begin{array}{rcl} x_{AB} & = & x_{CD} - 5 \\ x_{AD} & = & 25 - x_{CD} \\ x_{CB} & = & 15 - x_{CD} \\ x_{CD} & = & \text{arbitrario} \end{array} \right.$$

Chiaramente il valore della variabile  $x_{CD}$  non può essere scelto in modo completamente arbitrario, in quanto deve essere intero e non negativo. Inoltre, il valore di  $x_{CD}$  incide sul valore delle altre variabili, quindi è necessario scegliere solo quei valori per cui le altre variabili non siano negative. In questo caso,  $x_{CD}$  deve essere un intero compreso tra 5 e 15.

Si potrebbe dimostrare che il sistema lineare senza il vincolo di budget, se i termini noti sono interi, ha sempre una soluzione intera. Questa proprietà dipende dalla particolare struttura della matrice dei coefficienti (cioè la matrice è *totalmente unimodulare*). Infatti, scopriremo nel capitolo 13 che il modello usato in questo esempio è un caso particolare del *problema dei cammini di costo minimo*.

Rimane da determinare la soluzione che minimizza il budget speso. Per farlo è necessario scrivere la funzione di costo:

$$c(\mathbf{x}) = 10x_{AB} + 20x_{AD} + 5x_{CB} + 10x_{CD}$$

che può essere scritta in funzione della sola variabile  $x_{CD}$  sostituendo la soluzione  $\mathbf{x} = (x_{CD} - 5, 25 - x_{CD}, 15 - x_{CD}, x_{CD})$  come segue:

$$\begin{aligned} c(\mathbf{x}) &= 10x_{AB} + 20x_{AD} + 5x_{CB} + 10x_{CD} \\ c(\mathbf{x}) &= 10(x_{CD} - 5) + 20(25 - x_{CD}) + 5(15 - x_{CD}) + 10x_{CD} \\ c(\mathbf{x}) &= (10 - 20 - 5 + 10)x_{CD} + (-50 + 500 + 75) \\ c(\mathbf{x}) &= -5x_{CD} + 525 \end{aligned}$$

Siccome si ha un risparmio di 5 Euro per ogni auto trasportata dalla sede C alla sede D, cercheremo di inviare il massimo numero di auto possibile, che è pari a 15. Se fissiamo  $x_{CD} = 15$  la soluzione sarà  $\mathbf{x} = (10, 10, 0, 15)$  e il costo complessivo pari a  $c(\mathbf{x}) = 450$  Euro.



## Capitolo 2

# Programmazione Matematica

### 2.1 Problemi di ottimizzazione matematica

L'ottimizzazione matematica è uno dei principali strumenti messi a disposizione dalla Ricerca Operativa per la formulazione e soluzione di problemi decisionali. In tale contesto il problema decisionale viene formulato attraverso la definizione di un opportuno *problema di ottimizzazione matematica*, o di *programmazione matematica*.

Un problema di ottimizzazione matematica è definito da tre componenti principali.

1. La prima componente è il vettore  $x = (x_1, \dots, x_n) \in \mathbb{R}^n$  di *variabili decisionali*, ossia incognite il cui valore rappresenta una decisione da prendere nel problema corrente. Ad esempio le variabili possono rappresentare il numero di prodotti da realizzare, gli istanti in cui produrli, i quantitativi di merci o materie prime da immagazzinare o l'istante del loro riordino, il luogo in cui realizzare una infrastruttura o i tratti di strada da scegliere in un percorso. A seconda della natura delle decisioni che rappresentano, le variabili decisionali possono assumere valori continui (ad esempio, se rappresentano i quantitativi di materie prime da inserire in una miscela o una percentuale) o valori discreti o binari (quando, ad esempio, rappresentano il numero di unità da produrre o decisioni del tipo si/no). Si noti che, come sarà ampiamente discusso nel seguito, la natura delle variabili decisionali ha un forte impatto sulla difficoltà di risoluzione del problema di ottimizzazione, essendo i problemi con variabili discrete molto più difficili da risolvere rispetto ai problemi continui.
2. La seconda componente è rappresentata dalla *regione ammissibile*  $F \subseteq \mathbb{R}^n$ , ossia l'insieme delle soluzioni ammissibili del problema. La regione ammissibile  $F$  può essere definita in modo esplicito, specificando le proprietà

delle soluzioni ammissibili oppure elencando i suoi elementi. Ad esempio  $F = [0, 1]^2$  oppure  $F = \{x \text{ intere nell'ipercubo di lato 1}\}$ . Più frequentemente però la regione ammissibile viene definita in modo implicito, servendosi di equazioni e disequazioni, generalmente chiamati *vincoli* del problema. In tal caso la regione ammissibile è il luogo dei punti (cioè vettori dello spazio  $\mathbb{R}^n$ ) che soddisfano *tutte* le relazioni utilizzate per definirla. In generale, la regione ammissibile può essere descritta nel seguente modo

$$F = \{x \in \mathbb{R}^n : g_i(x) \leq 0, i = 1, \dots, m; h_j(x) = 0, j = 1, \dots, p\}, \quad (2.1)$$

dove  $m$  e  $p$  rappresentano il numero di disequazioni e di equazioni, rispettivamente.

In Figura 2.1 è rappresentato un esempio di regione ammissibile definita implicitamente dalle tre disequazioni lineari:

$$F = \{x \in \mathbb{R}^2 : 5x_1 + 3x_2 \leq 15; 5x_1 - 3x_2 \geq 0; x_2 \geq 1/2\}. \quad (2.2)$$

Si noti che, come sarà più evidente in seguito, alcune diseguaglianze sono espresse in forma di  $\geq$  mentre altre sono scritte nella forma di  $\leq$ .

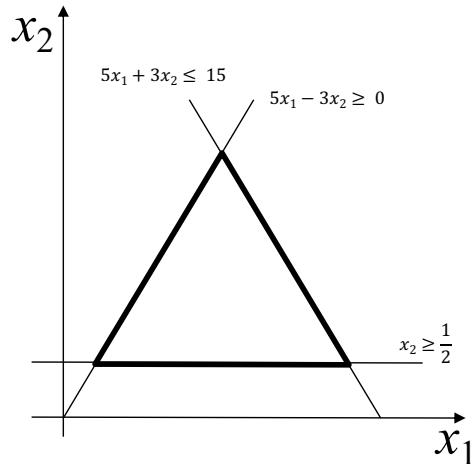


Figura 2.1: Esempio di regione ammissibile definita implicitamente

3. La terza componente è, infine, la funzione obiettivo (o funzione costo), ovvero una funzione  $\varphi : F \rightarrow R$  che associa ad ogni soluzione ammissibile un valore che ne rappresenta il costo o il profitto.

Un problema di ottimizzazione, in forma di minimizzazione, può dunque essere espresso come:

$$(P) \quad \min_{x \in F} \varphi(x). \quad (2.3)$$

Risolvere il problema  $(P)$  consiste nel determinare il minimo della funzione  $\varphi$  sugli elementi dell'insieme  $F$ , ossia identificare un ottimo globale  $x^* \in F$  tale che:

$$\varphi(x^*) \leq \varphi(x) \forall x \in F. \quad (2.4)$$

Nel caso generale, la funzione obiettivo e la regione ammissibile sono qualsiasi. Per questo motivo, talvolta la determinazione di un ottimo globale può essere computazionalmente onerosa. In tal caso, spesso ci si accontenta di determinare un ottimo locale.

**Definizione 2.1.** Un punto  $y \in F$  è un ottimo locale se esiste un intorno  $N \subseteq F$  tale che  $\varphi(y) \leq \varphi(x), \forall x \in N$ .

È noto che, nel caso di funzioni continue, al fine della definizione di ottimo locale, possa essere utilizzato l'*intorno euclideo* di dimensione  $\epsilon > 0$ , ossia  $N_\epsilon = \{x \in F : \|x - y\| \leq \epsilon, \epsilon > 0\}$ .

In Figura 2.2 è mostrato un esempio di problema con una variabile decisionale definita in una regione compatta e caratterizzata da una funzione obiettivo nonlineare continua. Per tale problema l'ottimo globale è il punto B, mentre gli altri punti di minimo (A e C) lo è solo localmente e sono pertanto ottimi locali.

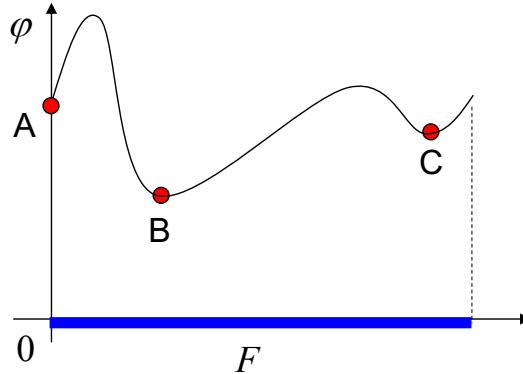


Figura 2.2: Esempio di problema di ottimizzazione

Più precisamente un ottimo globale può essere definito come segue.

**Definizione 2.2.** Un ottimo globale è un ottimo locale di valore minimo.

**Definizione 2.3.** Un intorno si dice esatto se un ottimo locale rispetto a tale intorno è anche un ottimo globale.

Si noti che non è detto che l'ottimo globale esista. Questo può accadere in diversi casi; i più comuni sono i seguenti

- la regione ammissibile è vuota ( $F = \emptyset$ ); in questo caso il problema è detto *impossibile* e convenzionalmente si pone  $\min\{\varphi(x) : x \in F\} = +\infty$ ;
- la funzione  $\varphi$  non è limitata inferiormente sull'insieme  $F$ . In tal caso si dice che il problema è detto *illimitato* e convenzionalmente si pone  $\min\{\varphi(x) : x \in F\} = -\infty$ . Questo è quello che accade, ad esempio, quando  $F = \{x \in \mathbb{R} : x \leq 10\}$  e la funzione obiettivo è  $\min \varphi(x) = x$ .
- non esiste un ottimo globale finito poichè la regione ammissibile non è un insieme compatto. Ad esempio, si consideri il caso in cui  $F = \{x \in \mathbb{R} : 1 < x \leq 10\}$  e la funzione obiettivo è  $\min \varphi(x) = x$ . In questo caso esiste un limite inferiore al valore assunto dalla funzione, ma non è possibile raggiungere il punto nel quale questo valore viene assunto.

Inoltre, l'ottimo globale può non essere unico, come nel caso di una funzione obiettivo periodica quale una sinusoide. Infine va osservato che possiamo sempre assumere che il problema da trattare richieda la minimizzazione della funzione obiettivo, come nella (2.3). Infatti, nel caso in cui la funzione obiettivo sia da massimizzare, è facile trasformare il problema in un problema di minimizzazione equivalente. A tal fine, basta ricordare che

$$\max_{x \in F} \varphi(x) = -\min_{x \in F} -\varphi(x).$$

Per questo motivo, in seguito faremo generalmente riferimento al solo caso di problemi espressi in forma di minimizzazione.

I problemi di ottimizzazione matematica appena descritti sono spesso chiamati problemi di programmazione matematica. Si noti che tale termine deriva dal fatto che tali problemi sono stati storicamente utilizzati per descrivere problemi di programmazione o pianificazione economica. Nel caso in cui la regione ammissibile sia un insieme discreto e finito o numerabile, tali problemi sono chiamati di ottimizzazione combinatoria.

Vediamo ora alcuni esempi di problemi di programmazione ed esaminiamone le proprietà principali.

## 2.2 Classificazione dei problemi di ottimizzazione

I problemi di ottimizzazione possono essere raggruppati in classi, a seconda delle caratteristiche delle variabili, dei vincoli, e quindi della regione ammissibile  $F$  e della funzione obiettivo  $\varphi$ . In ciascuna classe valgono determinate proprietà che, ad esempio, caratterizzano le soluzioni ottime, e che possono essere sfruttate da algoritmi di soluzione specifici per i problemi di quella classe. Pertanto, conoscere a quale classe appartiene un determinato problema di ottimizzazione

non è solamente un esercizio teorico; infatti, tale conoscenza ci permette di prevedere con buona accuratezza se il problema sia risolubile in modo efficiente e, eventualmente, quale sia l'algoritmo più idoneo per risolvere tale problema.

### 2.2.1 Problemi di Programmazione Non Lineare (PNL)

In tale classe non è presente nessuna restrizione sulla forma di  $\varphi$  e  $F$ . È quindi la classe più generale di problemi di ottimizzazione che, come ci si può facilmente aspettare, comprende problemi “difficili” da risolvere. In tali casi è tipicamente possibile soltanto calcolare ottimi locali per il problema.

### 2.2.2 Problemi di Programmazione Convessa (PC)

I problemi convessi rappresentano un caso speciale dei problemi PNL in cui funzione obiettivo e regione ammissibile hanno proprietà particolari che ne favoriscono la soluzione. Ricordiamo le seguenti definizioni:

**Definizione 2.4.** *Dati due punti  $x, y \in \mathbb{R}^n$  un punto  $z \in \mathbb{R}^n$  è combinazione convessa di  $x$  e  $y$  se esiste un  $\lambda \in [0, 1]$  tale che  $z = \lambda x + (1 - \lambda)y$ .*

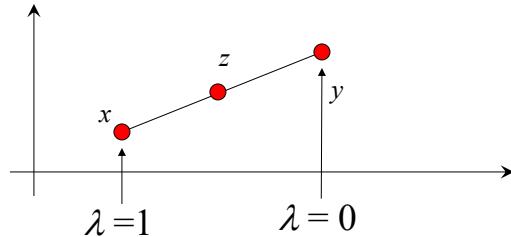


Figura 2.3: Esempio di combinazione convessa di due punti  $x, y \in \mathbb{R}^2$ .

**Definizione 2.5.** *Un insieme  $F \subseteq \mathbb{R}^n$  è convesso se contiene tutte le combinazioni convesse dei suoi punti, ossia se per ogni  $x, y \in F$  e per ogni  $\lambda \in [0, 1]$ , il punto  $z = \lambda x + (1 - \lambda)y$  appartiene all'insieme  $F$*

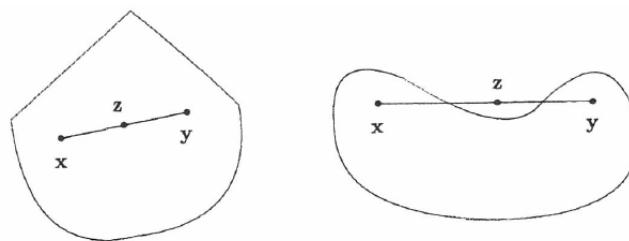


Figura 2.4: Esempio di insieme convesso (sinistra) e di insieme non convesso (destra).

Si noti che  $\mathbb{R}^n$  è ovviamente un insieme convesso e inoltre vale la seguente proprietà:

**Teorema 2.1.** *Data una collezione di insiemi convessi  $F_i, i = 1, \dots, k$ , l'insieme  $F$  intersezione di tali insiemi è un insieme convesso.*

**Dim.: 2.1.** Consideriamo due punti  $x, y \in F$  appartenenti all'insieme intersezione. Per definizione  $x, y \in F_i \forall i$ . Poiché gli insiemi  $F_i$  sono convessi le combinazioni convesse  $z = \lambda x + (1 - \lambda)y \in F_i \forall i$  e quindi  $z \in F$  da cui discende che  $F$  è convesso.

Si noti che, come illustrato dalla figura 2.5 l'unione di insiemi convessi non è invece necessariamente un insieme convesso.

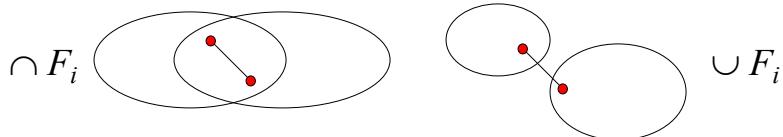


Figura 2.5: Intersezione ed unione di insiemi convessi.

**Definizione 2.6.** Sia  $F \subseteq \mathbb{R}^n$  un insieme convesso. Una funzione  $f : F \rightarrow \mathbb{R}$  è convessa se, per ogni  $x, y \in F$  e  $\lambda \in [0, 1]$ , si ha

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y)$$

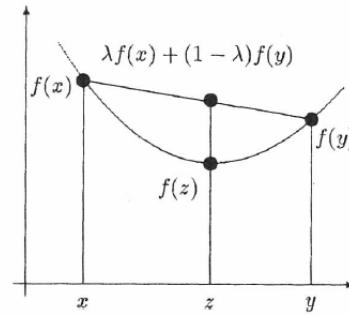


Figura 2.6: Esempio di funzione convessa.

Un problema è di programmazione convessa (PC) se la funzione obiettivo  $\varphi$  è una funzione convessa e la regione ammissibile  $F$  è un insieme convesso.

Per i problemi PC vale la seguente proprietà fondamentale:

**Teorema 2.2.** Sia  $F \subseteq \mathbb{R}^n$  un insieme convesso e sia  $\varphi : F \rightarrow \mathbb{R}$  una funzione convessa. Sia  $x \in F$  un ottimo locale, ossia tale che

$$\varphi(x) \leq \varphi(y) \quad \forall y \in N_\varepsilon(x) = \{y \in F : \|x - y\| \leq \varepsilon\} \quad \text{per qualche } \varepsilon > 0.$$

Allora  $x$  è un ottimo globale per  $\varphi$  su  $F$ .

In altre parole il teorema implica che l'intorno euclideo è un intorno esatto per il problemi PC. Il teorema 2.2 può essere facilmente dimostrato per contraddizione:

**Dim.: 2.2.** *Supponiamo che la tesi del teorema sia falsa. Dato un punto  $x \in F$  ottimo locale rispetto ad un intorno euclideo  $N_\varepsilon(x)$ , esiste un punto  $y \in F \setminus N_\varepsilon(x)$  non appartenente a tale intorno che è in realtà ottimo globale, ossia  $\varphi(y) \leq \varphi(z) \forall z \in F$ .*

*Scegliamo, come illustrato in figura 2.7, un punto  $z \in N_\varepsilon(x)$  combinazione convessa di  $x$  ed  $y$ , ossia  $z = \lambda x + (1 - \lambda)y$  con  $\lambda$  prossimo ad 1. Per la convessità della funzione  $\varphi$  si ha che  $\varphi(z) = \varphi(\lambda x + (1 - \lambda)y) \leq \lambda\varphi(x) + (1 - \lambda)\varphi(y)$  da cui si ricava che.*

$$\varphi(y) \geq (\varphi(z) - \lambda\varphi(x))/(1 - \lambda).$$

*Poiché  $z \in N_\varepsilon(x)$  si ha che  $\varphi(z) \geq \varphi(x)$ , per cui sostituendo nella disequazione precedente  $\varphi(x)$  a  $\varphi(z)$  questa rimane soddisfatta. Da questo discende che*

$$\varphi(y) \geq (\varphi(x) - \lambda\varphi(x))/(1 - \lambda) = \varphi(x)$$

*che implica che  $y$  non possa essere ottimo globale essendo il suo valore non migliore di quello dell'ottimo locale  $x$ .*

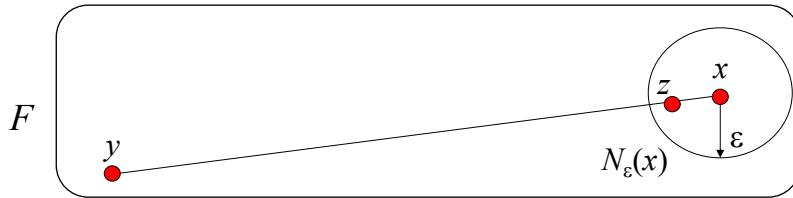


Figura 2.7: Illustrazione del teorema 2.2.

Quindi in un PC un ottimo locale è anche un ottimo globale. I PC sono risolvibili in modo efficiente, ossia esistono algoritmi che convergono ad un ottimo globale in modo efficiente.

### 2.2.3 Problemi di Programmazione Lineare (PL)

Se  $\varphi$  è una funzione lineare e  $F$  è definito da vincoli lineari allora il problema è di Programmazione Lineare (PL). Spesso per distinguere i problemi PL da quelli in cui le variabili sono tutte o in parte vincolate ad assumere valori discreti i problemi PL sono anche detti di Programmazione Lineare Continua (PLC). Nel caso di un problema PL si ha dunque:

- $f(x) = f(x_1, x_2, \dots, x_n) = \sum_{j=1}^n c_j x_j$

- $X = \{x \in \Re_+^n : \sum_{j=1}^n a_{ij}x_j \geq b_i \quad (i = 1, \dots, m)\}$

Si noti che poiché le funzioni lineari sono anche un caso particolare di funzioni convesse i problemi PL sono un caso particolare dei problemi PC e quindi anche per essi un algoritmo che converge ad un ottimo locale determina l'ottimo globale per il problema.

Spesso i problemi PL sono espressi in forma matriciale nel seguente modo:

$$(PL) \quad z(PL) = \min c^T x \\ s.t. \quad Ax \geq b \\ x \geq 0$$

Si consideri il seguente esempio di problema PL con  $n = 2$  e  $m = 3$ :

$$\begin{array}{rcl} \min & 4x_1 - 7x_2 \\ & x_1 - 2x_2 \geq 10 \\ - & 3x_1 + 4x_2 \geq 2 \\ & 5x_1 \geq 8 \end{array} \quad \text{La rappresentazione matriciale del problema è la seguente:}$$

- vettore  $c$  di dimensione  $n$  (*costi*):  $c_1 = 4, c_2 = -7$
- matrice  $A$  di dimensione  $m \times n$  (*matrice dei vincoli*):

$$\begin{aligned} a_{11} &= 1, & a_{12} &= -2, \\ a_{21} &= -3, & a_{22} &= 4, \\ a_{31} &= 5, & a_{32} &= 0 \end{aligned}$$

- vettore  $b$  di dimensione  $m$  (*termini noti*):  $b_1 = 10, b_2 = 2, b_3 = 8$

#### 2.2.4 Problemi di Programmazione Lineare Intera (PLI)

In molte applicazioni, le variabili decisionali rappresentano delle grandezze che non possono essere frazionate a piacere, ma che devono assumere un valore intero. Questo accade, ad esempio, quando le variabili rappresentano il numero di pezzi da produrre, il numero di persone da assumere, il numero di impianti da costruire. Inoltre esistono delle situazioni nelle quali le variabili decisionali modellano delle scelte logiche da fare, ad esempio “devo fare questa scelta oppure no?”, “devo costruire un magazzino in questa posizione oppure no?”, “devo servire un certo cliente oppure no?”. Per gestire tutte queste situazioni, bisogna aggiungere al modello matematico il vincolo che impone che le variabili decisionali assumano valori interi. Se il vincolo di interezza viene aggiunto ad una formulazione PL, il modello derivante è un modello di PLI, che può essere descritto (in forma matriciale) nel seguente modo

$$(PL) \quad z(PL) = \min c^T x$$

$$\text{s.t.} \quad Ax \geq b$$

$$x \geq 0 \quad \text{intere}$$

### 2.2.5 Problemi di Programmazione Lineare Mista (PLM)

Se il vincolo di interezza coinvolge solo *alcune* variabili ed il resto del problema è un PL, si ottiene un problema lineare misto, che può essere descritto nel seguente modo

$$(PL) \quad z(PL) = \min c^T x$$

$$\text{s.t.} \quad Ax \geq b$$

$$x_j \geq 0 \quad \text{intere} \quad j \in J$$

$$x \geq 0,$$

dove  $J$  indica l'insieme di variabili che devono assumere un valore intero.

### 2.2.6 Esempio 1: Problema lineare continuo

Consideriamo il caso illustrato nella Figura 2.8, in cui  $F = [0, 3/2]^2 = \{(x_1, x_2) : 0 \leq x_1 \leq 3/2; 0 \leq x_2 \leq 3/2\}$  e con funzione obiettivo  $\max \varphi(x) = x_1 + x_2$ . La regione ammissibile del problema è mostrata nella Figura 2.8(a) ed è costituita da un quadrato di lato  $3/2$  con un vertice nell'origine. Si noti che tale regione ammissibile è un insieme convesso, più precisamente un poliedro, che contiene infinite soluzioni ammissibili. La funzione obiettivo del problema è lineare e considerando le curve  $x_1 + x_2 = c$ , dove  $c$  è un valore costante arbitrario è possibile individuare la soluzione ottima del problema. Nella Figura 2.8(b) sono mostrati alcuni esempi di tali curve corrispondenti a diversi valori della costante. Queste curve sono dette *curve di livello* e descrivono gli insiemi di punti  $(x_1, x_2)$  per i quali la funzione  $\varphi$  assume un valore costante pari a  $c$ . Data la linearità della funzione obiettivo è possibile verificare che le curve  $x_1 + x_2 = c$  definiscono un fascio di rette parallele. La soluzione ottima del problema considerato è facilmente determinabile come la soluzione ammissibile che giace sulla retta del fascio corrispondente al valore massimo della costante  $c$  ossia il punto di coordinate  $(3/2, 3/2)$  per il quale il valore della funzione obiettivo è 3. Si noti che in questo caso la soluzione ottima si trova in corrispondenza di uno dei vertici del poliedro e, come vedremo in seguito, tale proprietà è una caratteristica dei problemi di programmazione lineare.

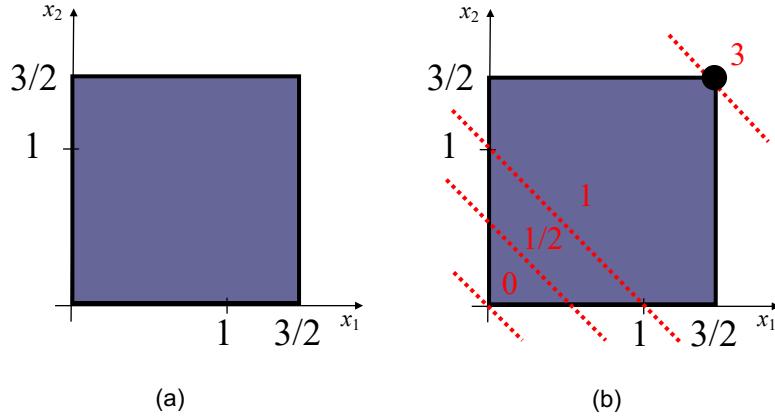


Figura 2.8: Esempio di problema di ottimizzazione lineare continuo.

### 2.2.7 Esempio 2: Problema lineare intero

Consideriamo il caso illustrato nella Figura 2.9, in cui  $F = \{0, 3/2\}^2 = \{(x_1, x_2) : 0 \leq x_1 \leq 3/2; 0 \leq x_2 \leq 3/2, \text{ intere}\}$  e con funzione obiettivo  $\max \varphi(x) = x_1 + x_2$ . La regione ammissibile del problema è mostrata nella Figura 2.9(a) ed è costituita dai quattro punti  $\{(0,0), (0,1), (1,0), (1,1)\}$ . Si noti che in questo caso la regione ammissibile non è un insieme convesso ed è costituita da un insieme finito di soluzioni ammissibili. La funzione obiettivo del problema è lineare e considerando le curve  $x_1 + x_2 = c$ , dove  $c$  è un valore costante arbitrario è possibile individuare la soluzione ottima del problema come nell'esempio precedente. Nella Figura 2.9(b) sono mostrati alcuni esempi di tali curve corrispondenti a diversi valori della costante. Data la linearità della funzione obiettivo è possibile verificare che le curve  $x_1 + x_2 = c$  definiscono un fascio di rette parallele. La soluzione ottima del problema considerato è facilmente determinabile come la soluzione ammissibile che giace sulla retta del fascio corrispondente al valore massimo della costante  $c$  ossia il punto di coordinate  $(1,1)$  per il quale il valore della funzione obiettivo è 2. Si noti che in questo caso la soluzione ottima si potrebbe determinare più semplicemente per enumerazione, ossia valutando la funzione obiettivo in corrispondenza di ciascuna soluzione intera. Per l'esempio illustrato sarebbero quindi sufficienti quattro valutazioni per determinare la soluzione ottima. Tale metodo però risulta rapidamente inadeguato qualora il numero di variabili decisionali cresca. Infatti in tal caso il numero di punti da valutare cresce in modo estremamente rapido. A tal proposito si consideri il caso più generale in cui  $F = \{0, 1\}^n$  per cui le soluzioni ammissibili sono i vertici di un ipercubo di lato unitario, le cui coordinate sono tutti i possibili vettori binari di dimensione  $n$ , che sono  $2^n$ . E facile osservare che già per valori relativamente piccoli di  $n$  il numero di soluzioni da valutare diventa troppo elevato per essere effettuato in tempi ragionevoli. Ad esempio per  $n = 100$  il numero di soluzioni è pari a  $2^{100} = 1.27 \dots \cdot 10^{30}$ .

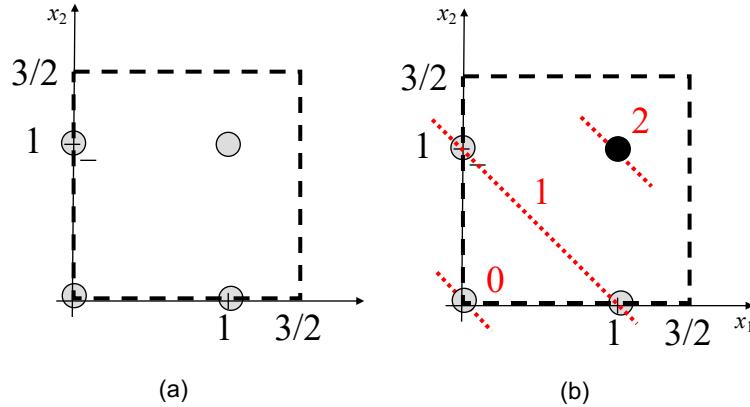


Figura 2.9: Esempio di problema di ottimizzazione lineare intero.

### 2.2.8 Esempio 3: Problema nonlineare continuo

Consideriamo il caso illustrato nella Figura 2.10, in cui  $F = [0, 1]^2 = \{(x_1, x_2) : 0 \leq x_1 \leq 1; 0 \leq x_2 \leq 1\}$  e con funzione obiettivo  $\min \varphi(x) = (x_1 - 1/2)^2 + (x_2 - 1/2)^2$ . La regione ammissibile del problema è mostrata nella Figura 2.10(a) ed è costituita da un quadrato di lato unitario con un vertice nell'origine. Si noti che tale regione ammissibile è un insieme convesso, più precisamente un poliedro, che contiene infinite soluzioni ammissibili. La funzione obiettivo del problema è nonlineare e considerando le curve  $x_1 + x_2 = c$ , dove  $c$  è un valore costante arbitrario è possibile individuare la soluzione ottima del problema. Nella Figura 2.10(b) sono mostrati alcuni esempi di tali curve corrispondenti a diversi valori della costante ed è possibile verificare che le curve  $\varphi(x) = c$  definiscono un insieme di circonferenze concentriche tutte con centro nel punto  $(1/2, 1/2)$ . La soluzione ottima del problema considerato è facilmente determinabile come la soluzione ammissibile che giace sulla circonferenza corrispondente al valore minimo della costante  $c$  ossia il punto di coordinate  $(1/2, 1/2)$  per il quale il valore della funzione obiettivo è 0. Si noti che in questo caso in conseguenza della nonlinearità di una delle componenti del problema, la soluzione ottima si trova in corrispondenza di uno degli infiniti punti all'interno del poliedro e non più di uno dei suoi vertici come nell'esempio della sezione 2.2.6.

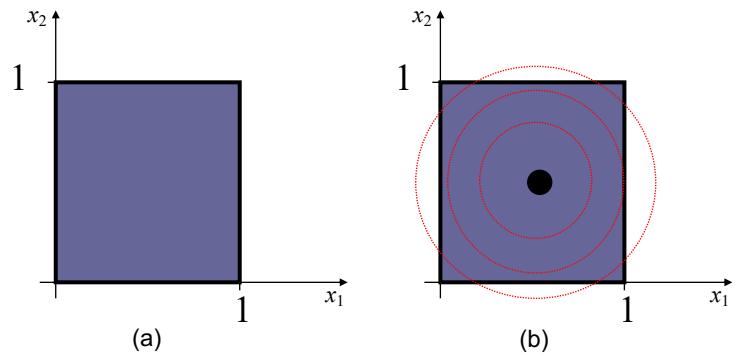


Figura 2.10: Esempio di problema di ottimizzazione non lineare continuo.

## Capitolo 3

# Programmazione Lineare

In questo capitolo vengono introdotte le proprietà fondamentali dei problemi di programmazione lineare così definiti

**Definizione 3.1 (Problema PL).**  $(F, \varphi)$  è un problema di programmazione lineare (PL, o linear programming, LP) se la funzione obiettivo  $\varphi$  è lineare

$$\varphi(x) = c^T x = c_1 x_1 + c_2 x_2 + \cdots + c_n x_n$$

e la regione  $F \subseteq \mathbb{R}^n$  è definita da  $g_i(x) \leq 0$  e  $h_j(x) = 0$  ( $i = 1, \dots, m$ ;  $j = 1, \dots, p$ ) con  $g_i, h_j : \mathbb{R}^n \rightarrow \mathbb{R}$  lineare  $\forall i, j$

$$g_i(x) : a_{i1} x_1 + a_{i2} x_2 + \dots + a_{in} x_n \geq d_i$$

$$h_j(x) : a_{j1} x_1 + a_{j2} x_2 + \dots + a_{jn} x_n = d_j$$

Generalmente i problemi PL si esprimono in forma matriciale

$$(PL) \quad \min c^T x \tag{3.1}$$

$$\text{s.t.} \quad Ax \geq d, \tag{3.2}$$

$$x \geq 0 \tag{3.3}$$

**Esempio 3.1.** Si consideri il problema PL

$$\begin{array}{llll} \min & 3x_1 & -2x_2 & +x_3 \\ \text{s.t.} & 2x_1 & +x_2 & -x_3 & \geq 2 \\ & x_1 & & +2x_3 & \geq 1 \\ & x_1, & x_2, & x_3 & \geq 0 \end{array}$$

Tale problema può essere rappresentato nella forma matriciale (7.1)–(7.3) definendo  $n = 3$  quale numero delle variabili pari al numero delle colonne della matrice  $A$ ,  $m = 2$  quale numero dei vincoli pari al numero delle righe della matrice  $A$ , da cui si ottiene

$$c^T = [3 \ -2 \ 1], \quad A = \begin{bmatrix} 2 & 1 & -1 \\ 1 & 0 & 1 \end{bmatrix}, \quad d = \begin{bmatrix} 2 \\ 1 \end{bmatrix}.$$

Si noti che per le proprietà degli insiemi convessi (si veda l'appendice) è facile verificare che la regione ammissibile di un problema PL è un insieme convesso, noto come *poliedro*. Una caratteristica importante dei poliedri è di essere delimitati da iperpiani (associati alle equazioni e disequazioni che lo definiscono) che si intersecano in un numero finito di vertici. Vedremo nel seguito che i vertici della regione ammissibile rivestono un ruolo fondamentale nella risoluzione dei problemi PL.

### 3.1 Formulazione di problemi PL

La formulazione di un problema PL deve seguire diversi passi.

1. Raccogliere la descrizione del problema, l'obiettivo da perseguire e le informazioni caratteristiche.
2. Individuare le variabili decisionali.
3. Definire la funzione obiettivo come combinazione delle variabili decisionali.
4. Definire i vincoli come combinazione delle variabili decisionali.
5. Identificare eventuali limiti superiori o inferiori (upper o lower bound) per i valori delle variabili decisionali.

Nel seguito forniremo alcuni esempi di problemi formulabili con modelli PL seguendo i passi sopra elencati.

#### 3.1.1 Problema di produzione di sedie

Come primo esempio di problema di programmazione lineare si considera la pianificazione della produzione di un turno di lavoro di una fabbrica di sedie. La fabbrica produce due tipi di sedie: una con telaio in legno (SL) e la seconda con telaio in alluminio (SA). Il flusso di produzione è rappresentato nella Figura 3.1. Per ciascun tipo di sedia l'assemblaggio del telaio viene realizzata in un apposito distinto reparto di lavorazione: RL per le sedie in legno ed RA per quelle in alluminio. Entrambe le sedie sono poi rifinite nello stesso reparto per la lavorazione delle parti in tessuto, RT.

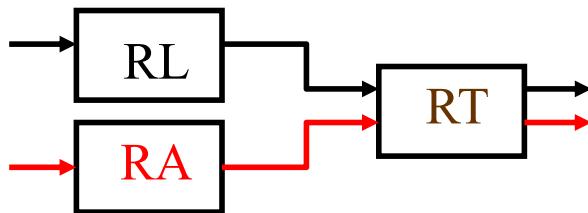


Figura 3.1: Flusso di lavorazione delle sedie attraverso i reparti lavorazione legno (RL), alluminio (RA) e tessuto (RT).

La lavorazione di una sedia in legno richiede 10 minuti nel reparto RL e 30 minuti in RT, mentre una in alluminio richiede 20 minuti in RA e 20 in RT. In ogni turno di lavoro sono disponibili per la lavorazione delle sedie 40 minuti nel reparto RL, 120 minuti in RA e 180 minuti in RT. Infine la vendita di una sedia in legno permette di ottenere un profitto di 30 € mentre quella in alluminio permette un profitto di 50 €. I dati del problema sono quindi

- tempi di lavorazione in minuti per pezzo,
- disponibilità dei reparti in minuti per turno,
- Profitto dalla vendita in € per pezzo,

e sono riassunti nella seguente tabella:

Tipo	tempi (min)			Profitto (€)
	RL	RA	RT	
SL	10	–	30	30
SA	–	20	20	50
Disponibilità	40	120	180	

Supponendo di poter vendere tutta la produzione, il problema consiste nel definire il mix di produzione che massimizza il profitto per le sedie in un generico turno, nel rispetto dei tempi disponibili per la lavorazione nei diversi reparti.

Le variabili decisionali del problema sono quindi associate alla produzione di ciascun tipo di sedia in un turno e sono espresse in unità di sedie prodotte. Utilizzeremo quindi le due variabili decisionali

- $x_1$ : numero di sedie in legno prodotte in un turno,
- $x_2$ : numero di sedie in alluminio prodotte in un turno.

Si noti che le variabili decisionali possono assumere valori frazionari dato che la produzione è relativa ad un singolo turno e si protrarrà per più turni. Un valore frazionario di una variabile decisionale in questo caso significa semplicemente che la frazione di sedia iniziata in un turno sarà completata nel turno successivo. Ad esempio  $x_1 = 3.5$  significa che ogni due turni saranno prodotte complessivamente 7 sedie in legno.

Una volta individuate le variabili decisionali è facile definire la funzione obiettivo ed i vincoli del problema. In particolare la funzione obiettivo rappresenta il profitto complessivo della produzione che si desidera massimizzare. Tale profitto è ottenibile sommando il profitto ottenibile dalla produzione di ciascun tipo di sedia. I vincoli del problema impongono il rispetto dei limiti del tempo disponibile in ciascun reparto in un turno e saranno quindi vincoli di tipo  $\leq$  il cui termine noto è la disponibilità del reparto in questione. Per ciascun reparto il tempo complessivo utilizzato è rappresentato dal tempo utilizzato dalla produzione di ciascun tipo di sedia. Infine, per ciascuna variabile decisionale è necessario impostare un limite inferiore pari a 0 al suo valore dato che produzioni negative non hanno senso. Il modello completo è quindi

$$\max \quad 30x_1 + 50x_2 \quad (3.4)$$

$$\text{s.t.} \quad 10x_1 \leq 40 \quad (3.5)$$

$$+20x_2 \leq 120 \quad (3.6)$$

$$30x_1 + 20x_2 \leq 180 \quad (3.7)$$

$$x_1, x_2 \geq 0 \quad (3.8)$$

Ai fini della risoluzione è preferibile che i coefficienti del modello siano interi di piccolo valore assoluto per cui il modello può essere riscritto in maniera equivalente dividendo i coefficienti della funzione obiettivo e dei vincoli per 10 e dividendo ulteriormente per 2 il secondo vincolo ottenendo il modello

$$\begin{aligned} \max \quad & 3x_1 + 5x_2 \\ \text{s.t.} \quad & x_1 \leq 40 \\ & x_2 \leq 120 \\ & 3x_1 + 2x_2 \leq 180 \\ & x_1, x_2 \geq 0 \end{aligned}$$

### 3.1.2 Problema di produzione di vasche per idromassaggio

Si consideri ora un altro esempio di problema di produzione, questa volta di vasche per idromassaggio. L'azienda HotBubbles produce due tipi di vasca, denominate Blue Tornado e Hot Spring, la cui produzione rende 350 e 300 € per pezzo, rispettivamente. La produzione di ciascuna vasca richiede un motore ed un certo numero di ore di lavoro e di metri di tubazione riassunti nella seguente tabella:

Tipo	Blue Tornado	Hot Spring
Motore	1	1
lavoro (ore)	9	6
tubi (metri)	12	16
Profitto	350€	300€

In ciascun turno di lavoro l'azienda dispone per la produzione di 200 motori, 1566 ore di lavoro e 2880 metri di tubo. Supponendo di poter vendere tutta la produzione, il problema consiste nel definire il mix di produzione che massimizza il profitto per le sedie in un generico turno, nel rispetto della disponibilità delle risorse.

Le variabili decisionali del problema sono quindi associate alla produzione di ciascun tipo di vasca in un turno e sono espresse in unità prodotte. Utilizzeremo quindi le due variabili decisionali

- $x_1$ : numero di vasche Blue Tornado prodotte in un turno,

- $x_2$ : numero di vasche Hot Spring prodotte in un turno.

Si noti che le variabili decisionali possono assumere valori frazionari dato che la produzione è relativa ad un singolo turno e si protrarrà per più turni. Un valore frazionario di una variabile decisionale in questo caso significa semplicemente che la frazione di vasca iniziata in un turno sarà completata nel turno successivo. Ad esempio  $x_1 = 3.5$  significa che ogni due turni saranno prodotte complessivamente 7 vasche di tipo Blue Tornado.

Una volta individuate le variabili decisionali, procedendo come nell'esempio precedente è facile definire la funzione obiettivo ed i vincoli del problema. In particolare la funzione obiettivo rappresenta il profitto complessivo della produzione che si desidera massimizzare. Tale profitto è ottenibile sommando il profitto ottenibile dalla produzione di ciascun tipo di sedia. I vincoli del problema impongono il rispetto dei limiti del tempo disponibile in ciascun reparto in un turno e saranno quindi vincoli di tipo  $\leq$  il cui termine noto è la disponibilità del reparto in questione. Per ciascun reparto il tempo complessivo utilizzato è rappresentato dal tempo utilizzato dalla produzione di ciascun tipo di sedia. Infine, per ciascuna variabile decisionale è necessario impostare un limite inferiore pari a 0 al suo valore dato che produzioni negative non hanno senso. Il modello completo è quindi

$$\max \quad 350x_1 + 300x_2 \quad (3.9)$$

$$\text{s.t.} \quad x_1 + x_2 \leq 200 \quad (3.10)$$

$$9x_1 + 6x_2 \leq 1566 \quad (3.11)$$

$$12x_1 + 16x_2 \leq 2880 \quad (3.12)$$

$$x_1, x_2 \geq 0 \quad (3.13)$$

### 3.1.3 Problemi di mix di produzione

I due esempi precedenti sono rappresentativi della famiglia di problemi di mix di produzione. In generale tali problemi sono caratterizzati dalla presenza dei seguenti dati:

$n$  prodotti da produrre (in quantità anche frazionaria)

$m$  risorse (macchine, materie prime, ...)

$r_j$  ricavo ottenibile dalla produzione di una unità del prodotto  $j$ ,  $j = 1, \dots, n$

$a_{ij}$  unità della risorsa  $i$  necessarie per la produzione di una unità del prodotto  $j$ ,  $i = 1, \dots, m$ ,  $j = 1, \dots, n$

$d_i$  unità di risorsa  $i$  disponibili,  $i = 1, \dots, m$ .

Le variabili decisionali (frazionarie) del problema sono:

$x_j$  unità del prodotto  $j$  da produrre,  $j = 1, \dots, n$

Il modello PL del problema, in forma matriciale, risulta essere:

$$(MixProd) \quad \max r^T x \quad (3.14)$$

$$\text{s.t.} \quad Ax \leq d, \quad (3.15)$$

$$x \geq 0 \quad (3.16)$$

### 3.1.4 Problema della dieta

Un altro esempio rilevante di problema di ottimizzazione formulabile con un modello PL è rappresentato dal cosiddetto problema della dieta. In questo caso bisogna definire la dieta ottimale da somministrare agli animali di un allevamento miscelando in modo opportuno dei mangimi. La dieta deve contenere un fabbisogno minimo di un certo numero di elementi nutritivi e per ciascun mangime sono noti il contenuto di ciascun elemento per Kg ed il relativo costo. La seguente tabella riporta un esempio dei dati di un problema con due mangimi e tre elementi nutritivi (Proteine, Carboidrati e Vitamine):

Elemento	Mangime A	Mangime B	Fabbisogno
Proteine (g/Kg)	300	300	900g
Carboidrati (g/Kg)	600	200	1200g
Vitamine (mg/Kg)	100	500	500mg
Costo (€/Kg)	1	2	

Si desidera determinare la miscela di costo minimo che garantisca il fabbisogno minimo di elementi agli animali. Le variabili decisionali del problema sono quindi associate al quantitativo in Kg di ciascun mangime nella miscela e possono assumere valori frazionari. Utilizzeremo quindi le due variabili decisionali

- $x_1$ : Kg di mangime A nella miscela,
- $x_2$ : Kg di mangime B nella miscela.

Una volta individuate le variabili decisionali, procedendo come nell'esempio precedente è facile definire la funzione obiettivo ed i vincoli del problema. In particolare la funzione obiettivo rappresenta il costo complessivo della miscela, ottenibile sommando il costo di ciascun tipo di mangime moltiplicato per il relativo costo unitario, che si desidera minimizzare. I vincoli del problema impongono il rispetto dei limiti minimi del fabbisogno di elementi nutritivi nella dieta e saranno quindi vincoli di tipo  $\geq$  il cui termine noto è il relativo fabbisogno. Infine, per ciascuna variabile decisionale è necessario imporre un limite inferiore pari a 0 al suo valore dato che quantità negative non hanno senso. Il modello completo, in cui i coefficienti dei vincoli sono stati divisi per 100, è quindi

$$\min \quad x_1 + 2x_2 \quad (3.17)$$

$$\text{s.t.} \quad 3x_1 + 3x_2 \geq 9 \quad (3.18)$$

$$6x_1 + 2x_2 \geq 12 \quad (3.19)$$

$$x_1 + 5x_2 \geq 5 \quad (3.20)$$

$$x_1, x_2 \geq 0 \quad (3.21)$$

In generale i problemi della dieta sono caratterizzati dalla presenza dei seguenti dati:

$n$  alimenti o mangimi da utilizzare (in quantità anche frazionaria)

$m$  sostanze nutritive necessarie

$c_j$  costo di una unità dell'alimento  $j$ ,  $j = 1, \dots, n$

$a_{ij}$  unità della sostanza  $i$  contenute in una unità dell'alimento  $j$ ,  $i = 1, \dots, m$ ,  $j = 1, \dots, n$

$d_i$  fabbisogno della sostanza  $i$ ,  $i = 1, \dots, m$ .

Le variabili decisionali (frazionarie) del problema sono:

$x_j$  unità dell'alimento  $j$  da acquistare,  $j = 1, \dots, n$

Il modello PL del problema, in forma matriciale, risulta essere:

$$(Diet) \quad \min c^T x \quad (3.22)$$

$$\text{s.t.} \quad Ax \geq d, \quad (3.23)$$

$$x \geq 0 \quad (3.24)$$

## 3.2 Forme dei Problemi PL

Un problema PL può essere espresso in diverse forme a seconda delle tipologie di vincoli che in esso compaiono. La cosiddetta *forma generale* è evidentemente il caso generico di un problema PL in cui compaiono sia disequazioni sia equazioni e le variabili sono sia vincolate in segno sia libere:

$$\text{Forma Generale} \quad \min \text{ or } \max c^T x \quad (3.25)$$

$$\text{s.t.} \quad a_i^T x = d_i, \quad i \in M \quad (3.26)$$

$$a_i^T x \geq d_i, \quad i \in M' \quad (3.27)$$

$$x_j \geq 0 \quad j \in N \quad (3.28)$$

$$x_j \text{ libera} \quad j \in N' \quad (3.29)$$

Esistono però altre due forme particolari dei problemi PL che assumono un ruolo rilevante nella definizione degli algoritmi risolutivi. La prima è la *forma canonica* in cui il problema è formulato utilizzando unicamente disequazioni e variabili non-negative:

$$\text{Forma Canonica} \quad \min c^T x \quad (3.30)$$

$$\text{s.t.} \quad Ax \geq d, \quad (3.31)$$

$$x \geq 0 \quad (3.32)$$

La seconda è denominata *forma standard* ed in questo caso il problema è formulato utilizzando solo equazioni e variabili non-negative.

$$\text{Forma Standard} \quad \min c^T x \quad (3.33)$$

$$\text{s.t.} \quad Ax = d, \quad (3.34)$$

$$x \geq 0 \quad (3.35)$$

La particolare forma in cui un problema PL è inizialmente formulato non è però rilevante perché è facile trasformare la formulazione definendo un problema equivalente che assume la forma desiderata. Nel seguito esaminiamo le trasformazioni equivalenti di un problema PL.

### 3.2.1 Trasformazione della funzione obiettivo da max a min

È molto frequente che nei problemi PL la funzione obiettivo sia da massimizzare mentre come si è visto nella forma canonica e standard il problema è posto in forma di minimizzazione. La trasformazione in questo caso è molto semplice sfruttando la seguente proprietà illustrata nella figura 3.2:

$$\max c^T x = -\min -c^T x. \quad (3.36)$$

È quindi sufficiente cambiare il segno di tutti i coefficienti della funzione obiettivo. Si noti che risolvendo il problema così trasformato il valore ottenuto come soluzione ottima sarà uguale in valore assoluto a quello del problema originale ma il suo segno sarà cambiato come illustrato nella figura 3.2.

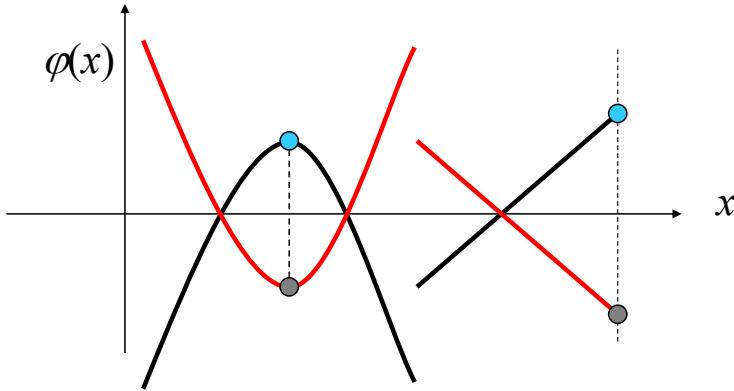


Figura 3.2: Equivalenza tra massimizzazione e minimizzazione di una funzione nel caso non lineare (sinistra) e lineare (destra).

### 3.2.2 Trasformazione di disequazioni in equazioni

La forma standard richiede che tutti i vincoli propri del problema PL siano espressi come equazioni. Qualora il problema contenga delle disequazioni queste possono facilmente essere trasformate in equazioni equivalenti aggiungendo ad esse una variabile non-negativa che rappresenta la differenza tra il primo membro ed il termine noto. Vanno considerati due casi:

- Disequazioni di tipo  $\leq$ : si aggiunge una variabile *slack* al primo membro che rappresenta il difetto del valore di  $x$  rispetto al termine noto  $d_i$

$$a_i^T x \leq d_i \quad \Rightarrow \quad a_i^T x + x_s = d_i \quad (3.37)$$

- Disequazioni di tipo  $\geq$ : si sottrae una variabile *surplus* al primo membro che rappresenta l'eccesso del valore di  $x$  rispetto al termine noto  $d_i$

$$a_i^T x \geq d_i \quad \Rightarrow \quad a_i^T x - x_s = d_i \quad (3.38)$$

Si noti che tutti i valori di  $x$  che soddisfano le disequazioni soddisfano anche le corrispondenti equazioni e che le trasformazioni aumentano il numero delle variabili decisionali del problema dovendo aggiungere una variabile slack/surplus per ogni disequazione.

### 3.2.3 Trasformazione di equazioni in disequazioni

La forma canonica richiede che tutti i vincoli propri del problema PL siano espressi come disequazioni di tipo  $\geq$ . Ovviamente le disequazioni di tipo  $\leq$  possono essere trasformate in disequazioni di tipo  $\geq$  moltiplicandole per  $-1$ .

Qualora il problema contenga delle equazioni queste possono facilmente essere trasformate in una coppia di disequazioni equivalenti come segue:

$$a_i^T x = d_i \Rightarrow \begin{cases} a_i^T x \geq d_i \\ -a_i^T x \geq -d_i \end{cases} \quad (3.39)$$

Si noti che i valori di  $x$  che soddisfano l'equazione sono l'intersezione delle soluzioni delle due disequazioni soddisfano e che le trasformazioni aumentano il numero dei vincoli del problema, dovendo aggiungere una disequazione per ogni equazione da trasformare.

### 3.2.4 Trasformazione di variabili libere

Le forme canonica e standard prevedono che tutte le variabili decisionali del problema siano non negative. Qualora il problema preveda delle variabili non positive ( $x_i \leq 0$ ) queste possono essere sostituite nella formulazione da variabili non negative e cambiando tutti i segni dei coefficienti ad esse relative nel problema.

Nel caso in cui il problema preveda variabili libere queste possono invece essere sostituite dalla differenza di due variabili non negative

$$x_i \text{ libera} \Rightarrow x_i = x_i^+ - x_i^- . \quad (3.40)$$

Si noti che tale trasformazione aumenta il numero di variabili decisionali dovendo aggiungerne una per ogni variabile libera.

### 3.2.5 Esempio di trasformazione di forma

Per illustrare le trasformazioni descritte si consideri il seguente esempio in forma generale:

$$\max z = -2x_1 + 3x_2 \quad (3.41)$$

$$\text{s.t.} \quad x_1 + 2x_2 \leq 4 \quad (3.42)$$

$$2x_1 - x_2 = 2 \quad (3.43)$$

$$x_1 \geq 0 \quad (3.44)$$

$$x_2 \text{ libera} \quad (3.45)$$

$$(3.46)$$

Consideriamo dapprima la trasformazione del problema in forma canonica. Trasformiamo innanzitutto la variabile libera  $x_2 = x_2^+ - x_2^-$  e sostituiamola in tutto il problema

$$\max z = -2x_1 + 3x_2^+ - 3x_2^- \quad (3.47)$$

$$\text{s.t.} \quad x_1 + 2x_2^+ - 2x_2^- \leq 4 \quad (3.48)$$

$$2x_1 - x_2^+ + x_2^- = 2 \quad (3.49)$$

$$x_1, x_2^+, x_2^- \geq 0 \quad (3.50)$$

A questo punto possiamo modificare la prima disequazione moltiplicandola per  $-1$  e convertire l'equazione in disequazioni come mostrato nel paragrafo 3.2.3. Infine modifichiamo la funzione obiettivo in  $\min$  come illustrato nel paragrafo 3.2.1. Il problema risultante in forma canonica è:

$$-\min -z = 2x_1 - 3x_2^+ + 3x_2^- \quad (3.51)$$

$$\text{s.t.} \quad -x_1 - 2x_2^+ + 2x_2^- \geq -4 \quad (3.52)$$

$$2x_1 - x_2^+ + x_2^- \geq 2 \quad (3.53)$$

$$-2x_1 + x_2^+ - x_2^- \geq -2 \quad (3.54)$$

$$x_1, x_2^+, x_2^- \geq 0 \quad (3.55)$$

Operiamo ora la trasformazione in forma standard. Dapprima sostituiamo la variabile libera come mostrato per la forma canonica. Poi sostituiamo la disequazione con una equazione aggiungendo la variabile surplus  $x_3$ . Il problema in forma standard è:

$$-\min -z = 2x_1 - 3x_2^+ + 3x_2^- \quad (3.56)$$

$$\text{s.t.} \quad -x_1 - 2x_2^+ + 2x_2^- + x_3 = -4 \quad (3.57)$$

$$2x_1 - x_2^+ + x_2^- = 2 \quad (3.58)$$

$$x_1, x_2^+, x_2^-, x_3 \geq 0 \quad (3.59)$$

Si noti che rispetto al problema originale i problemi in forma canonica e standard, pur essendo a questo e tra loro equivalenti, contengono variabili aggiuntive e hanno una funzione obiettivo in forma di minimo. Risolvendo uno di tali problemi per ottenere la soluzione ottima del problema originale in forma generale sarà necessario:

- a) definire il valore della variabile libera da quelli delle due variabili ausiliarie, ovvero  $x_2 = x_2^+ - x_2^-$ ,
- b) cambiare il segno del valore della funzione obiettivo  $-z$  determinato.

I valori delle variabili slack e surplus potranno essere ignorati.

### 3.2.6 Equivalenza tra le forme di PL

## 3.3 Interpretazione Geometrica dei Problemi PL

In questo paragrafo esamineremo le proprietà geometriche dei problemi PL e illustreremo la risoluzione per via grafica di problemi PL contenti due o tre variabili decisionali. Nei capitoli successivi saranno invece discussi metodi più generali per la risoluzione di problemi PL di dimensione qualsiasi.

I problemi PL illustrati nei paragrafi precedenti hanno una struttura molto semplice e la loro soluzione potrebbe essere ottenuta anche sulla base di semplici considerazioni intuitive. Consideriamo ad esempio il problema di produzione di sedie del paragrafo 3.1.1. In tale problema le sedie in alluminio hanno un profitto più elevato. Inoltre, nel reparto lavorazione tessuto in cui le sedie sono in concorrenza richiedono un tempo di lavorazione inferiore. È dunque ragionevole pensare di produrre il maggior numero possibile di tali sedie ed impiegare le risorse eventualmente disponibili per la produzione di sedie in legno. Il massimo numero di sedie in alluminio producibili è facilmente ottenibile dalle (3.6)–(3.7) imponendo in queste  $x_1 = 0$  e calcolando il valore massimo di  $x_2$  che rispetta tutti i vincoli. Tale valore è  $x_2 = 6$  corrispondente al vincolo (3.6) che risulta il più stringente. Successivamente si può ottenere il valore massimo di  $x_1$  compatibile con i vincoli sostituendo in questi il valore  $x_2 = 6$  trovato ed ottenendo dal vincolo (3.7)  $x_1 = 2$ . In tal modo si è ottenuta la soluzione  $(2, 6)$  cui corrisponde un valore complessivo di 360€ che, come si vedrà in seguito, corrisponde alla soluzione ottima del problema.

Tali semplici considerazioni permettono in generale di determinare soluzioni ammissibili dei problemi PL ma in generale non permettono di determinare la soluzione ottima o di provarne l'ottimalità. Se consideriamo infatti il problema della produzione di vasche del paragrafo 3.1.2, procedendo in modo analogo a quanto descritto per la produzione di sedie potremmo preferire la produzione di vasche di tipo Blue Tornado che hanno un profitto maggiore. Imponendo  $x_2 = 0$  nei vincoli (3.10)–(3.12) si ottiene  $x_1 = 174$  in corrispondenza del vincolo (3.11) che risulta essere il più stringente. Si noti che fissato  $x_1$  a tale valore non risultano disponibili risorse sufficienti a produrre vasche di tipo Hot Spring in quanto la produzione determinata utilizza completamente le ore di lavoro disponibili nel turno. La soluzione così determinata risulta essere  $(174, 0)$  a cui corrisponde un profitto di 60900€. Tale soluzione, pur essendo ammissibile non è la soluzione ottima del problema che corrisponde invece alla soluzione  $(122, 78)$  a cui corrisponde un profitto di 66100€.

Vediamo ora, sempre utilizzando gli esempi precedenti, come rappresentarli geometricamente e determinarne la soluzione per via grafica.

### 3.3.1 Disegno della Regione Ammissibile di un Problema PL

Prendiamo dapprima in considerazione la regione ammissibile dei problemi PL osservando che questa risulta essere il luogo dei punti che soddisfano tutti i

vincoli del problema definiti da equazioni e disequazioni

$$F = \{x \in \Re^n : g_i(x) \leq 0, i = 1, \dots, m; h_j(x) \leq 0, j = 1, \dots, p\} \quad (3.60)$$

ossia l'insieme dei punti che costituisce l'intersezione degli insiemi che soddisfano singolarmente ciascun vincolo.

È noto che nello spazio  $\Re^n$  una equazione lineare definisce un insieme di punti detto *iperpiano*. Nel caso  $\Re^2$  questo risulta essere una retta come illustrato in figura 3.3, mentre in  $\Re^3$  questo risulta essere un piano (si veda la figura 3.4). Una disequazione lineare definisce invece un semispazio delimitato dall'equazione lineare di supporto ottenuta sostituendo la diseguaglianza con “=” come illustrato in figura 3.5.

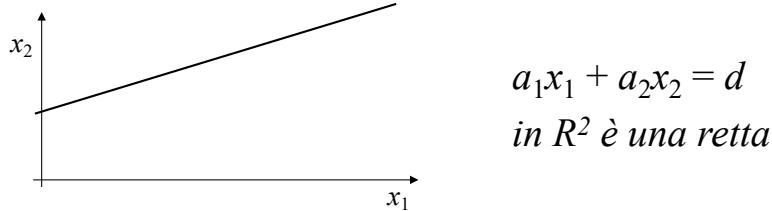


Figura 3.3: Luogo dei punti corrispondenti ad una equazione lineare in  $\Re^2$ .

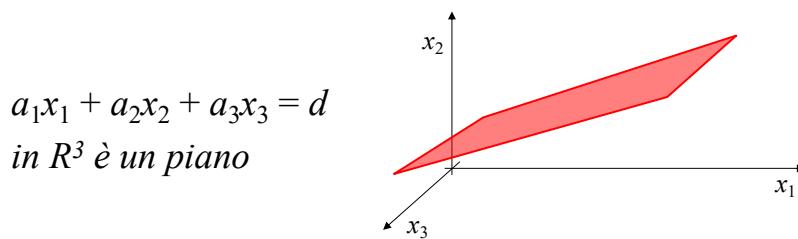


Figura 3.4: Luogo dei punti corrispondenti ad una equazione lineare in  $\Re^3$ .

$S = \{ x \in R^n : a^T x \leq d \}$  è un **semispazio**

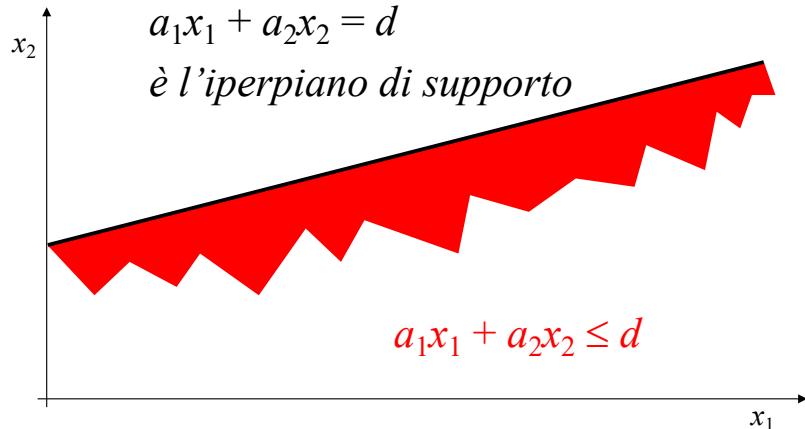


Figura 3.5: Luogo dei punti corrispondenti ad una disequazione lineare in  $\Re^2$ .

La regione ammissibile di un problema PL risulta quindi essere l'intersezione di un numero finito di iperpiani e di semispazi, ossia quello che viene definito un *poliedro convesso*.

Consideriamo l'esempio della produzione di sedie descritto nel paragrafo 3.1.1 che è definito mediante due variabili decisionali e quindi si presta ad essere rappresentato graficamente nello spazio  $\Re^2$ . La regione ammissibile di tale problema si trova come intersezione dei 5 vincoli del problema ossia dei 3 vincoli propri associati ai limiti di produzione dei diversi reparti ed i 2 vincoli di non negatività delle variabili. Questi ultimi due restringono la regione ammissibile al primo quadrante del piano cartesiano mentre i tre vincoli rimanenti sono associati a tre disequazioni che complessivamente definiscono la regione ammissibile rappresentata in figura 3.6. Più precisamente, al fine di determinare la regione ammissibile di un problema PL devono essere considerati tutti i vincoli del problema, comprendendo anche gli eventuali vincoli sul segno delle variabili decisionali. Per ciascun vincolo deve essere determinata la corrispondente regione ammissibile. Ad esempio, come già discusso, i vincoli di non negatività delle variabili restringono la regione ammissibile al primo quadrante mentre il vincolo  $x_1 \leq 4$  definisce il semispazio alla sinistra della retta  $x_1 = 4$ . Intersecando le regioni ammissibili di tutti i vincoli si determina la regione ammissibile del problema.

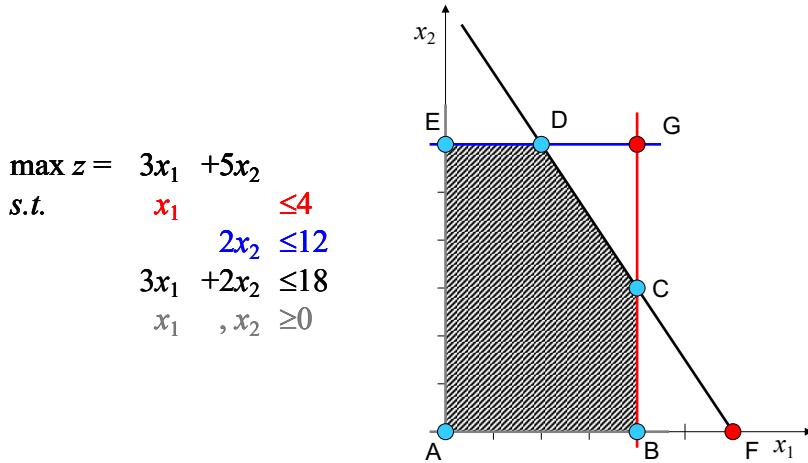


Figura 3.6: Regione ammissibile del problema di produzione di sedie.

### 3.3.2 Poliedri e vertici

Come già discusso la regione ammissibile definita da equazioni e disequazioni lineari definisce un poliedro convesso. Le facce di tale poliedro sono porzioni di iperpiani che si intersecano in spigoli che a loro volta si intersecano definendone i vertici, come mostrato nella figura 3.6. Si noti che la presenza di equazioni tra i vincoli "riduce" la dimensione della regione ammissibile rispetto a quella dello spazio che lo contiene. Ad esempio, se consideriamo il problema rappresentato in figura 3.6 la sua regione ammissibile è definita solo da disequazioni in  $\mathbb{R}^2$  ed è un poliedro di dimensione 2. Se in tale problema il primo vincolo fosse invece l'equazione  $x_1 = 4$  allora la regione ammissibile si ridurrebbe al segmento CB ossia ad un insieme di dimensione 1.

I vertici sono punti particolarmente importanti di un poliedro ed assumeranno un ruolo fondamentale anche nella soluzione dei problemi PL. Informalmente i vertici di un poliedro sono punti ottenibili intersecando tra loro un numero di equazioni di supporto dei vincoli del problema pari alla dimensione dello spazio che contiene la regione ammissibile che, in generale, coincide con il numero delle variabili decisionali. Ad esempio per il problema illustrato in figura 3.6 che ha due variabili decisionali, i vertici si ottengono intersecando due equazioni di supporto dei vincoli. Si noti però che intersecando coppie di equazioni associate ai vincoli si ottengono anche punti esterni alla regione ammissibile come, ad esempio, il punto G della figura 3.6 che si ottiene come intersezione delle equazioni  $x_1 = 4$  e  $x_2 = 6$ . Tali vertici "impropri" del poliedro si definiscono *vertici non ammissibili*. Riassumendo, i vertici A, B, C, D, ed E di figura 3.6 sono vertici ammissibili, mentre F e G sono vertici non ammissibili.

Vediamo ora meglio il ruolo dei vertici nella soluzione dei problemi PL enunciando una proprietà fondamentale di tali problemi.

**Definizione 3.2.** Un punto  $x \in \mathbb{R}^n$  è vertice di un insieme convesso  $S \subset \mathbb{R}^n$  se e solo se le sue coordinate non sono esprimibili come combinazione convessa di altri punti di  $S$ .

I vertici sono punti fondamentali di un poliedro convesso. Vale infatti il seguente teorema la cui definizione è omessa per brevità.

**Teorema 3.1.** Dati i vertici  $\{p_1, \dots, p_k\}$  di un poliedro limitato  $P$ , detto politopo, si ha il seguente teorema: le coordinate di un punto generico  $x \in P$  possono essere ottenute come combinazione convessa delle coordinate dei vertici, ossia esiste un vettore  $\lambda \geq 0$  di coefficienti non tutti nulli e tali che  $\sum_{i=1}^k \lambda_i = 1$  da cui si ottiene

$$x = \sum_{i=1}^k \lambda_i p_i \quad (3.61)$$

Possiamo a questo punto introdurre il teorema fondamentale della PL.

**Teorema 3.2.** In un problema PL con regione ammissibile  $F$  non vuota e limitata esiste sempre almeno un vertice ottimo.

Per dimostrare il teorema occorre ragionare per assurdo. Consideriamo un problema di minimo e siano  $\{x^1, \dots, x^k\}$  i vertici della regione ammissibile  $F$  e  $c$  il vettore dei coefficienti di costo della funzione obiettivo. Supponiamo che la soluzione ottima del problema sia il punto  $x^0 \in F$  e che questa non coincida con uno dei vertici di  $F$ . Per il teorema 3.1 possiamo esprimere  $x^0$  come

$$x^0 = \sum_{i=1}^k \lambda_i x^i, \quad \text{con } \lambda_i \geq 0 \text{ e } \sum_{i=1}^k \lambda_i = 1, \quad (3.62)$$

Sia ora  $x^j$  il vertice di costo minimo, ossia  $c^T x^j = \min_{1 \leq i \leq k} \{c^T x^i\}$ . Il valore della funzione obiettivo nel punto ottimo,  $c^T x^0$  può essere espresso come combinazione convessa del valore della funzione obiettivo dei vertici, ossia  $c^T x^0 = \sum_{i=1}^k \lambda_i c^T x^i$ . Se in tale relazione sostituiamo a ciascun vertice le coordinate del vertice migliore è facile verificare che la quantità al secondo membro diventa in generale più piccola dato che nella combinazione convessa sostituiamo al valore della funzione obiettivo calcolata in ciascun vertice quello della funzione obiettivo nel vertice migliore. Possiamo ottenere le relazioni

$$c^T x^0 = \sum_{i=1}^k \lambda_i c^T x^i \geq \sum_{i=1}^k \lambda_i c^T x^j = c^T x^j \sum_{i=1}^k \lambda_i = c^T x^j \quad (3.63)$$

da cui discende che  $c^T x^0 \geq c^T x^j$ , ossia esiste un vertice cui corrisponde un valore della funzione obiettivo non peggiore di quella calcolata nel punto  $x^0$ . Pertanto si arriva alla contraddizione che o il punto  $x^0$  non è la soluzione ottima o esiste un vertice che rappresenta una soluzione ottima dello stesso valore.

La conseguenza del teorema 3.2 è che per determinare la soluzione ottima di un problema PL è sufficiente considerare i vertici della regione ammissibile.

Considerando che il numero di vertici di un politopo è finito anche se il politopo contiene infiniti punti, dal punto di vista computazionale la proprietà enunciata permette di determinare la soluzione ottima di un problema che ha infinite soluzioni ammissibili limitandosi ad esaminare un numero finito di punti.

### 3.3.3 Vincoli Ridondanti

La definizione implicita della regione ammissibile attraverso le equazioni e le disequazioni può portare a formulazioni inefficienti del problema ossia che contengono vincoli “inutili”. Considerando, ad esempio, il problema della produzione delle sedie se fosse necessario considerare un ulteriore reparto di lavorazione con una disponibilità di 70 minuti di lavoro e per il quale le sedie in legno richiedono 10 minuti di lavorazione e quelle in alluminio 7 minuti, sarebbe necessario aggiungere al problema il vincolo

$$10x_1 + 7x_2 \leq 70. \quad (3.64)$$

Aggiungendo tale vincolo al problema si può però osservare dalla figura 3.7 che tale vincolo non altera in alcun modo la regione ammissibile definita dagli altri vincoli e può essere quindi rimosso dal problema. Lo stesso può dirsi del vincolo  $3x_1 + x_2 \leq 15$  indicato in blu nella medesima figura. Si noti che, sebbene la rimozione di vincoli ridondanti sia sicuramente vantaggiosa per la risoluzione di un problema PL, la loro individuazione in modo analitico non è facile e la loro presenza nelle formulazioni di problemi di ottimizzazione è molto frequente.

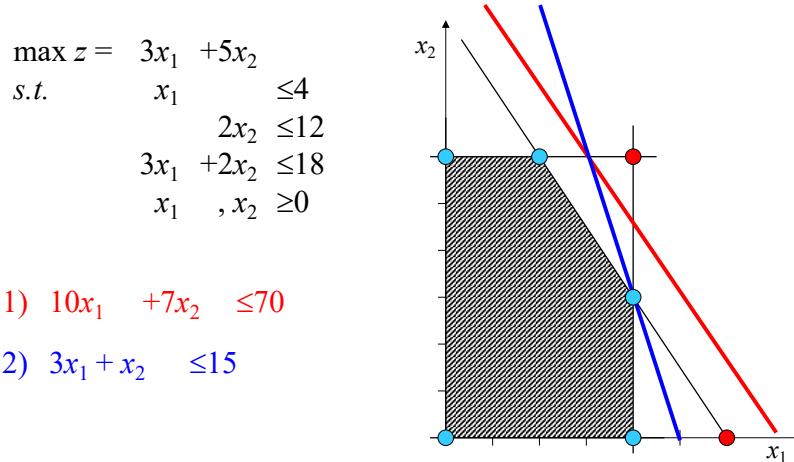


Figura 3.7: Esempi di vincoli ridondanti per un problema PL.

### 3.3.4 Risoluzione Grafica di un problema PL

Una volta disegnata la regione ammissibile del problema PL per individuare in questa la soluzione ottima occorre prendere in considerazione la funzione

obiettivo. A tal fine si osservi che ponendo la funzione obiettivo uguale ad un valore costante,  $\varphi(x) = \text{costante}$ , si ottiene il luogo dei punti che corrispondono ad un valore costante di questa. Osservando che la funzione obiettivo è lineare tali espressioni definiscono un fascio di iperpiani in  $\mathbb{R}^n$  paralleli. Chiaramente i punti della regione ammissibile che giacciono su un iperpiano definito da un particolare valore costante sono soluzioni ammissibili per il problema cui corrisponde tale valore della funzione obiettivo. Per determinare la soluzione ottima del problema è dunque sufficiente individuare la soluzione ammissibile che giace sull'iperpiano di valore minimo (o massimo, a seconda del tipo di problema).

Nel caso del problema delle sedie gli iperpiani  $\varphi(x) = \text{costante}$  sono delle rette in  $\mathbb{R}^2$ . In figura 3.10 sono mostrate le rette equi-costo che intersecano la regione ammissibile corrispondenti ai valori  $3x_1 + 5x_2 = 15, 20, 25, \dots$  è quindi chiaro che trattandosi di un problema di massimizzazione la soluzione ottima corrisponderà ad una retta del fascio che interseca la regione ammissibile ad un valore più elevato della funzione obiettivo. Tale punto è costituito dal vertice di coordinate  $(2, 6)$  cui corrisponde il valore 36 della funzione obiettivo.

Si noti che dato che la regione ammissibile è convessa iperpiani di valore più elevato della funzione ammissibile non avranno intersezione con la regione ammissibile pertanto il punto così individuato è la soluzione ottima del problema e corrisponde ad una produzione di 2 sedie in legno e 6 in alluminio garantendo un profitto di 360 € per turno. Si ricordi infatti che i coefficienti della funzione obiettivo erano stati divisi per 10 per cui il valore ottimo determinato con la funzione obiettivo  $\max z = 3x_1 + 5x_2$  è espresso in decine di €.

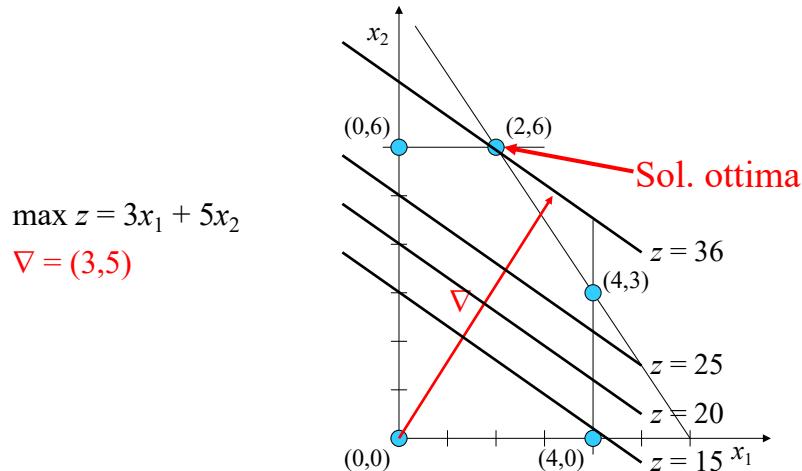


Figura 3.8: Risoluzione per via grafica del problema PL di produzione delle sedie.

In alternativa al calcolo delle rette equi-costo della funzione obiettivo può essere determinato il gradiente  $\nabla$  della funzione obiettivo, ossia il vettore che

congiunge l'origine con il punto le cui coordinate sono le derivate parziali prime della funzione obiettivo rispetto alle variabili decisionali. Nel caso della funzione obiettivo  $\max z = 3x_1 + 5x_2$  si ha che il gradiente, illustrato in figura 3.10 si ottiene congiungendo l'origine con il punto di coordinate (3, 5). È noto che il gradiente di una funzione di più variabili indica per ogni punto la direzione di variazione di tale funzione e nel caso di funzioni lineari tale direzione sia costante e per tale motivo la direzione del gradiente risulta perpendicolare alle rette equi-costo definite dalla funzione obiettivo. Inoltre il verso del gradiente indica la direzione di aumento della funzione e questo è ben illustrato dalla figura 3.10 che mostra le rette equi-costo corrispondenti a valori crescenti della funzione obiettivo. Pertanto una volta determinato il gradiente è sufficiente, con l'aiuto di un righello muoversi perpendicolarmente alla sua direzione ed in verso concorde per problemi di massimizzazione e nel verso opposto per la minimizzazione, finché non si interseca l'ultimo punto della regione ammissibile che corrisponderà alla soluzione ottima cercata.

### 3.3.5 Considerazioni sulla soluzione ottima di un PL

Si sarà notato dalla soluzione dell'esempio precedente che la soluzione ottima del problema è stata individuata in corrispondenza di un vertice della regione ammissibile del problema. In realtà questo non è un caso fortuito ma una proprietà fondamentale dei problemi PL che, come meglio vedremo in seguito, garantisce che se la soluzione ottima esiste (ossia se il problema PL non risulta impossibile) essa coincide sempre con almeno un vertice della regione ammissibile. Tale proprietà discende direttamente dalla linearità dei vincoli e della funzione obiettivo. Infatti, la prima ha come conseguenza che la regione ammissibile sia un poliedro convesso e la seconda che le superfici equi-costo della funzione obiettivo definiscono un fascio di iperpiani paralleli. Considerando esempi come quello della produzione di sedie che sono rappresentabili graficamente in  $\mathbb{R}^2$  non è difficile osservare che questo si verifichi mentre in un paragrafo successivo proveremo che tale proprietà vale in generale. Si osservi inoltre che se viene a mancare la linearità di una delle componenti del problema tale proprietà viene a cadere come illustrato in figura 3.9 in cui a sinistra è rappresentato il caso in cui non sia lineare la funzione obiettivo mentre a destra quello in cui ad essere non lineare è uno dei vincoli del problema. In entrambi i casi la soluzione ottima non è necessariamente in un vertice ma è lungo uno spigolo della regione ammissibile. Si ricordi anche l'esempio illustrato nel paragrafo 2.2.8 in cui la funzione obiettivo è non lineare e la soluzione ottima è un punto all'interno della regione ammissibile.

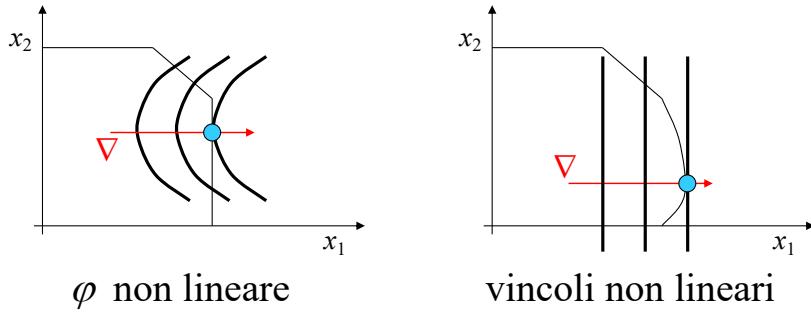


Figura 3.9: Esempio di problemi non lineari e relativa soluzione ottima.

In generale la soluzione ottima è unica ed ha valore finito come nel caso appena considerato, ma si possono verificare anche alcuni casi particolari. Se, ad esempio, nel problema della produzione di sedie i profitti fossero diversi e la funzione obiettivo corrispondente fosse  $\max z = 3x_1 + 2x_2$ , risolvendo il problema per via grafica come mostrato precedentemente si otterrebbe che la soluzione ottima non è più unica. Infatti, muovendosi perpendicolarmente al gradiente ed in verso concorde si può vedere che l'ultima intersezione con la regione ammissibile coincide con l'intero spigolo tra il punto di coordinate  $(2, 6)$  e quello di coordinate  $(4, 3)$ . In questo caso le soluzioni ottime del problema sono infinite e tutte corrispondenti al valore 18 della funzione obiettivo. Si osservi però che tra queste infinite soluzioni ottime sono presenti anche due vertici della regione ammissibile ossia i punti di coordinate  $(2, 6)$  e  $(4, 3)$ .

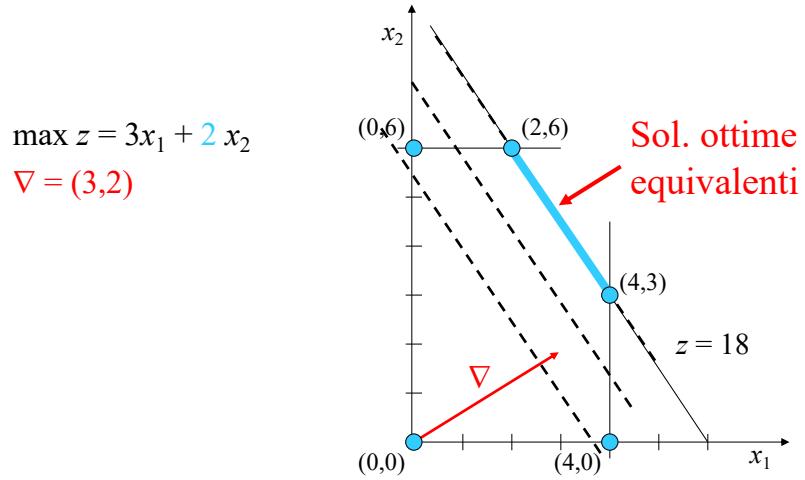


Figura 3.10: Esempio di soluzioni ottime equivalenti.

Nel problema esaminato fin qui la regione ammissibile è un poliedro limitato

ma tale situazione non è generalmente sempre verificata nei problemi PL. Sono infatti frequenti i casi di problemi di ottimizzazione in cui la regione ammissibile pur definita da vincoli lineari sia in realtà illimitata. Un esempio di tale situazione è rappresentato dal problema della dieta descritto nel paragrafo 3.1.4 cui i vincoli sono di tipo  $\geq$ . La regione ammissibile del problema è rappresentata in figura 3.11 ed è costituita dalla parte sopra le rette associate ai vincoli. Procedendo come per gli esempi precedenti si determina la soluzione ottima muovendosi in direzione opposta rispetto al gradiente dato che si tratta di un problema di minimizzazione. La soluzione ottima si trova nel vertice definito dai vincoli  $6x_1 + 2x_2 \geq 12$  e  $x_1 + 5x_2 \geq 5$ . Intersecando le due rette associate ai vincoli si trovano le coordinate del vertice che corrisponde al punto  $(\frac{5}{2}, \frac{9}{14})$  ed ha un costo pari a 5.642... €. Si noti che in questo caso pur essendo la regione ammissibile illimitata la soluzione ottima del problema ha valore finito.

Nel caso in cui per lo stesso problema la funzione obiettivo fosse stata da massimizzare risolvendo graficamente il problema si noterebbe che la soluzione ottima del problema è illimitata. Pur essendo questa una soluzione ammissibile per un problema PL, normalmente verificare che il problema è illimitato è indice di un errore nella modellazione del problema dato che questo corrisponde ad un valore infinito di almeno una delle variabili decisionali.

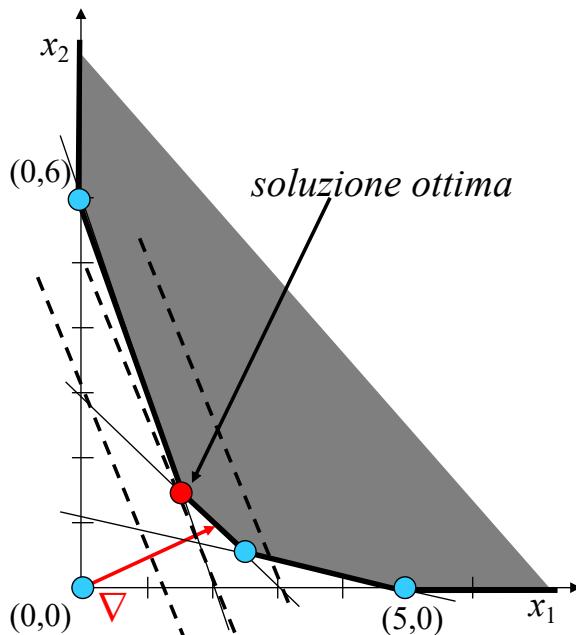


Figura 3.11: Risoluzione grafica del problema della dieta.

### 3.4 Assunzioni implicite della PL NEW: (Ver 3.0)

La costruzione di un modello PL implica diverse assunzioni importanti sulle caratteristiche del problema da modellare che esamineremo brevemente nel seguito.

#### 3.4.1 Proporzionalità

L'uso di funzioni lineari per modellare funzione obiettivo e vincoli del problema implica che l'impatto della decisione associata ad una variabile decisionale  $x_h$  :

- a) sia *proporzionale*, attraverso un coefficiente costante ( $c_h$  nella funzione obiettivo o  $a_{ih}$  nell' $i$ -simo vincolo), al valore di tale decisione;
- b) tale proporzionalità si mantenga per tutto l'intervallo di variazione della variabile.

Il contributo della variabile  $x_h$  alla funzione obiettivo di un problema PL è illustrato nella figura 3.12.

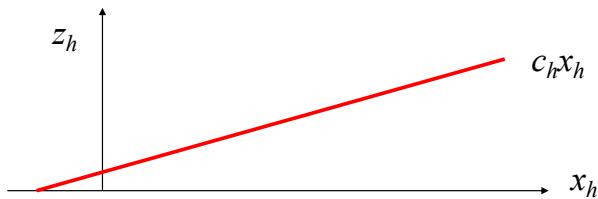


Figura 3.12: Contributo della variabile  $x_h$  alla funzione obiettivo in un problema PL.

Tale assunzione in molte situazioni pratiche è difficilmente verificata. Si pensi, ad esempio, alla produzione di beni in grandi quantità per i quali il profitto unitario tende a decrescere al crescere della produzione a causa dei maggiori investimenti in produzione o per la promozione delle vendite, come illustrato in figura 3.13. In maniera analoga tali assunzioni non sono facilmente verificate quando il contributo di una variabile mostra delle discontinuità, ad esempio motivate dagli investimenti aggiuntivi necessari a raggiungere un determinato livello di produzione. Tali fenomeni sono in generale approssimati molto male da una singola funzione lineare che in molti tratti sottostima o sovrastima l'effettivo contributo della variabile. Un possibile modo per mitigare tali errori di approssimazione è rappresentato dall'impiego di funzioni lineari a tratti come mostrato in figura 3.14 che però come vedremo in seguito richiedono l'impiego di variabili intere rendendo il modello di PLM.

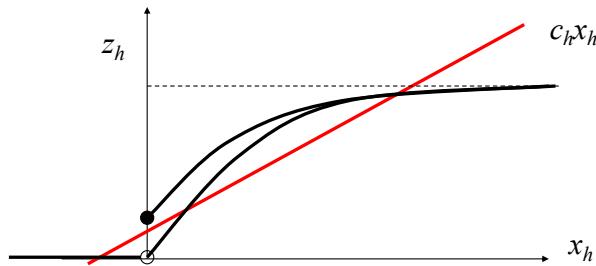


Figura 3.13: Fenomeni di saturazione o discontinuità nel contributo della variabile  $x_h$  alla funzione obiettivo.

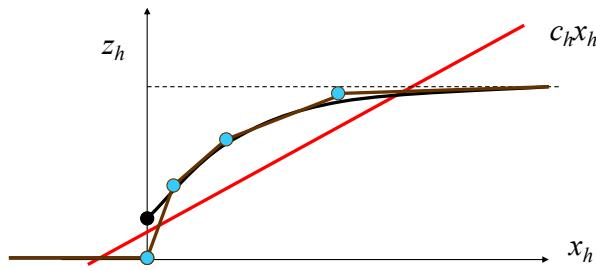


Figura 3.14: Approssimazione di fenomeni di saturazione o discontinuità mediante funzioni lineari a tratti.

### 3.4.2 Additività

L'uso di funzioni lineari per modellare funzione obiettivo e vincoli del problema implica che la funzione obiettivo e le espressioni al primo membro dei vincoli siano la somma di termini ciascuno dipendente unicamente dalla decisione associata ad una sola variabile decisionale  $x_h$ . In altre parole l'impatto su funzione obiettivo e vincoli di una decisione  $x_h$  dipende unicamente dal valore di tale decisione e non da quello di altre decisioni del modello.

Tale assunzione in molte situazioni pratiche è difficilmente verificata. Si pensi, ad esempio, alla produzione di beni in concorrenza tra di loro. In molti casi il profitto ottenibile dalla vendita di tali beni non dipende unicamente dalla quantità di questi prodotti ma anche dalla quantità di beni simili che l'azienda produce o sono disponibili sul mercato. Infatti, la disponibilità di prodotti alternativi concorrenti sul mercato potrebbe ridurre la facilità di vendita dei prodotti considerati costringendo a praticare sconti aggiuntivi e riducendone il profitto.

### 3.4.3 Divisibilità

L'uso di modelli PL implica che le variabili decisionali siano valori continui e quindi potenzialmente frazionari. In molti casi tale assunzione è perfettamente compatibile con il problema da modellare. Ad esempio quando le variabili rappresentano tassi di produzione in un periodo, percentuali sul valore totale o grandezze frazionarie quali i quantitativi di sostanze da acquistare o prevedere in una miscela.

In molti problemi però le decisioni rappresentate dalle variabili hanno senso solo se costituite da valori interi. Ad esempio, le decisioni potrebbero rappresentare un numero di addetti da impiegare in un lavoro o il numero di infrastrutture da costruire. In tali casi le variabili del modello devono necessariamente essere vincolate ad essere intere ed il modello diviene un modello PLM o PLI.

### 3.4.4 Certezza

L'ultimo aspetto da considerare nell'impiego di modelli PL è rappresentato dal fatto che i coefficienti utilizzati nel definire i dati di ingresso del modello sono costanti e di conseguenza assunti come dati deterministici, ossia noti con certezza. La conseguenza principale di questo è che la soluzione ottima determinata dipende strettamente da tali valori.

Nella maggior parte dei casi i dati del problema sono però il frutto di misure o stime e sono quindi inevitabilmente affetti da approssimazioni ed errori e quindi le soluzioni determinate potrebbero non essere necessariamente ottime se alcuni dati di ingresso variassero rispetto al valore utilizzato come input del modello. Si noti che tale inconveniente è del tutto generale nell'impiego di modelli in cui si impiegano dati deterministici e quindi anche per modelli PLM e PLI.

In generale, una volta determinata la soluzione ottima del problema è necessario analizzarne la sensibilità alla variazione dei parametri di ingresso e questo spesso richiede l'esecuzione ripetuta della soluzione con insiemi di dati di ingresso differenti che rappresentino le possibili variazioni di questi. Nel caso della PL, come vedremo meglio nel capitolo 5, l'analisi di sensitività alla variazione dei parametri di ingresso può essere condotta in modo piuttosto semplice permettendo di ottenere per ciascun dato di ingresso l'intervallo di variazione rispetto al valore attuale che mantenga ottima la soluzione determinata.

# Capitolo 4

## L’Algoritmo del Simplex

L’algoritmo del simplex<sup>1</sup> è uno strumento fondamentale per la risoluzione dei problemi di programmazione lineare (PL). L’algoritmo è dovuto a G.B. Dantzig che lo ha messo a punto nel 1947 ed è costituito da un procedimento iterativo che esplora i vertici del poliedro fino a determinare la soluzione ottima del problema. Nel seguito esporremo l’algoritmo in modo graduale dapprima esaminando una sua versione intuitiva basata sulle proprietà dei problemi PL da un punto di vista geometrico. Successivamente definiremo un algoritmo algebrico, analogo al metodo proposto da Dantzig basato sulla risoluzione di sistemi di equazioni in modo efficiente per ottenere la soluzione ottima dei problemi PL.

### 4.1 Versione intuitiva dell’algoritmo del simplex

Il metodo del simplex si basa su alcune proprietà dei problemi PL che abbiamo discusso nel capitolo precedente.

La prima di queste è la proprietà fondamentale dei problemi PL enunciata nel teorema 3.1 secondo la quale in un problema PL la soluzione ottima, se esiste, coincide con uno dei vertici del poliedro che costituisce la regione ammissibile. Considerando che in un poliedro limitato (politopo)<sup>2</sup> associato ad un problema PL ha un numero di vertici finito, un algoritmo enumerativo che li esamini tutti potrà quindi determinare la soluzione ottima del problema in un numero finito di iterazioni. Si noti che sebbene finito il numero di vertici del politopo può essere molto grande. Più precisamente in un problema PL in forma canonica con  $n$  variabili decisionali ed  $m$  disequazioni il numero di vertici è infatti limitato

---

<sup>1</sup>Un simplex è un politopo (poliedro limitato) con il minor numero possibile di vertici. In particolare un simplex di dimensione  $n$  è l’inviluppo di  $n + 1$  vertici.

<sup>2</sup>Per semplicità faremo principalmente riferimento a poliedri limitati ma quanto esposto si può facilmente estendere al caso di poliedri illimitati.

superiormente dal binomio di Newton

$$\binom{n}{m} = \frac{n!}{m!(n-m)!}. \quad (4.1)$$

La seconda proprietà su cui si basa l'algoritmo del simplesso è rilevante per la verifica dell'ottimalità della soluzione. In particolare si ricordi che i problemi PL sono un caso particolare dei problemi di ottimizzazione convessa per i quali si può dimostrare che l'intorno Euclideo è un intorno esatto. Come ricordato nella definizione 2.3, un intorno è esatto se un ottimo locale rispetto a tale intorno è in realtà un ottimo globale o assoluto per l'intero problema. In altre parole per verificare se il vertice che l'algoritmo sta esaminando sia o meno la soluzione ottima globale del problema è sufficiente verificarne l'ottimalità locale rispetto ad un intorno Euclideo di raggio  $\varepsilon$ . Nel caso dei problemi PL il problema di verificare computazionalmente l'ottimalità rispetto all'intorno Euclideo è ulteriormente semplificato da una semplice considerazione geometrica. In particolare per i problemi PL la verifica di ottimalità di un vertice può essere effettuata semplicemente confrontando il valore della funzione obiettivo nel vertice corrente con quella calcolata nei vertici adiacenti ad esso. Come mostrato nella figura 4.1 dato il vertice corrente ed i suoi vertici adiacenti questi definiscono un simplesso che contiene al suo interno la porzione ammissibile di un intorno Euclideo centrato nel vertice corrente. Dalla proprietà fondamentale dei problemi PL (Teorema 3.2) è noto che il valore della funzione obiettivo calcolata nel vertice migliore di un poliedro non è peggiore di quello calcolata in qualsiasi altro punto del poliedro. Quindi se il valore della funzione obiettivo calcolata nel vertice corrente risulta essere non peggiore (ossia nel caso di problemi di minimo sia  $\leq$ ) di quello calcolato nei vertici adiacenti allora il vertice corrente è ottimo locale rispetto all'intorno Euclideo essendo ottimo rispetto ad un poliedro che lo contiene e, di conseguenza, è l'ottimo globale del problema.

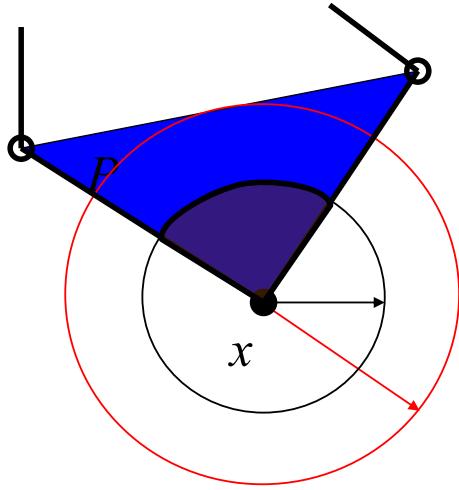


Figura 4.1: Illustrazione dell'intorno definito dai vertici adiacenti al vertice corrente.

Possiamo quindi definire una versione informale dell'algoritmo del simplex:

---

**Algorithm 1** Versione intuitiva dell'algoritmo del simplexo.

---

```

1: procedure SIMPLEX( $(c, A, d)$ )
2:   Inizializzazione: sia  $x$  un vertice ammissibile;
3:    $ottimo \leftarrow false$ 
4:   repeat
5:     if  $\exists$  un vertice  $x$  adiacente ad  $x$  migliore then
6:        $ottimo \leftarrow true$ 
7:     else
8:       Muoviti in un vertice  $x$  adiacente migliore
9:        $x \leftarrow x$ 
10:    end if
11:   until  $ottimo = true$ 
12:   return  $x$  (soluzione ottima)
13: end procedure

```

---

Consideriamo l'esempio della produzione di sedie che abbiamo introdotto nel paragrafo 3.1.1 e risolto graficamente nel paragrafo 3.3.4. Nella figura 4.2 è riportata una possibile applicazione dell'algoritmo 1 a tale problema in cui le frecce indicano l'evoluzione dell'algoritmo nelle diverse iterazioni. Supponiamo di partire dal vertice ammissibile  $E$ , corrispondente alla soluzione  $x = (0, 0)$  di valore 0 (riga 2). Entriamo nella fase iterativa dell'algoritmo e verifichiamo l'ottimalità del vertice corrente (riga 5). I vertici adiacenti a tale punto sono  $A = (0, 6)$  e  $D = (4, 0)$  e sono entrambi migliori di  $E$  dato che la funzione obiettivo in tali punti vale rispettivamente 30 e 12. Scegliamo tra tali punti il

vertice  $A$  e muoviamoci in tale punto definendo  $x = (0, 6)$  (riga 8). Ripetendo la verifica di ottimalità per il vertice corrente possiamo vedere che il vertice adiacente  $B = (3, 6)$  non è stato ancora esplorato e corrisponde ad un valore migliore della funzione obiettivo pari a 36. Si noti che il vertice  $E$ , adiacente ad  $A$  è già stato esplorato precedentemente e quindi ad esso non può corrispondere una soluzione migliore rispetto a quella del vertice corrente. Scegliamo quindi il vertice  $D$  e muoviamoci in tale punto definendo  $x = (3, 6)$  (riga 8). Verificando l'ottimalità del vertice corrente possiamo stabilire che questo è la soluzione ottima in quanto l'unico vertice adiacente  $C = (4, 3)$  corrisponde ad una soluzione peggiore di valore 27. Possiamo quindi terminare l'algoritmo restituendo come soluzione ottima il punto  $x = (3, 6)$  corrispondente al vertice  $B$ .

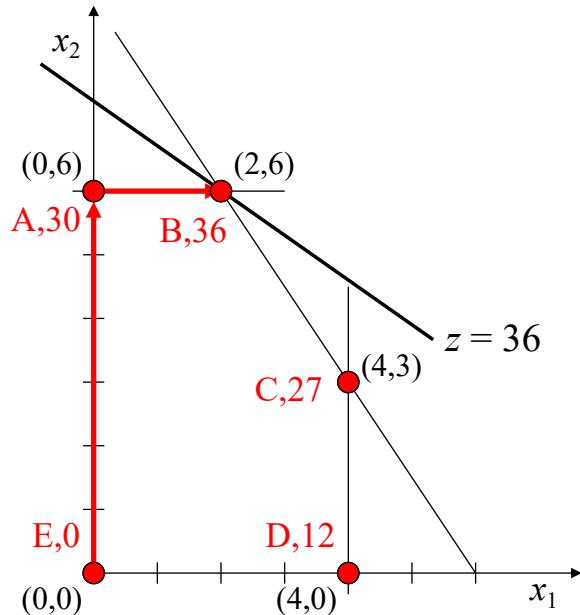


Figura 4.2: Applicazione dell'algoritmo del simplesso al problema della produzione di sedie.

## 4.2 Versione algebrica dell'algoritmo del simplesso

La versione dell'algoritmo del simplesso illustrata nell'algoritmo 1 è basata sui concetti geometrici di vertice ed intorno che sono sicuramente efficaci per una sua prima comprensione ma più difficilmente generalizzabili per la risoluzione di problemi di maggiore dimensione. A tal fine, sfruttando la disponibilità di metodi efficienti per la risoluzione di sistemi di equazioni lineari è possibile

riportarsi ad una versione algebrica dell'algoritmo del simplex che sarà più facile generalizzare e descrivere per la risoluzione di problemi PL di dimensione qualsiasi.

Per considerare la versione algebrica ci riferiamo quindi ad un problema in cui i vincoli siano tutti espressi sotto forma di equazione, ossia ad un problema espresso in forma standard (si veda il paragrafo 3.2). Considerando, ad esempio, il problema di produzione di sedie

$$\max \quad 3x_1 + 5x_2 \quad (4.2)$$

$$\text{s.t.} \quad x_1 \leq 4 \quad (4.3)$$

$$+x_2 \leq 6 \quad (4.4)$$

$$3x_1 + 2x_2 \leq 18 \quad (4.5)$$

$$x_1, x_2 \geq 0 \quad (4.6)$$

la sua espressione in forma standard è

$$-\min -3x_1 - 5x_2 \quad (4.7)$$

$$\text{s.t.} \quad x_1 + x_3 = 4 \quad (4.8)$$

$$+x_2 + x_4 = 6 \quad (4.9)$$

$$3x_1 + 2x_2 + x_5 = 18 \quad (4.10)$$

$$x_1, x_2, x_3, x_4, x_5 \geq 0 \quad (4.11)$$

Si noti che, dato un problema espresso in forma canonica con  $t$  variabili ed  $m$  vincoli espressi come disequazioni, il problema corrispondente in forma standard avrà  $n = t + m$  variabili. Nell'esempio, il problema originale ha infatti  $t = 2$  variabili ed  $m = 3$  vincoli mentre il problema in forma standard ha  $n = 2 + 3 = 5$  variabili e lo stesso numero di vincoli.

Lavorando sul problema in forma standard considereremo un problema con più variabili rispetto al problema originale e la regione ammissibile sarà rappresentata in uno spazio di dimensione superiore. Si noti però che la dimensione della regione ammissibile rimane la stessa di quella del problema originale. La presenza delle equazioni infatti proietta la regione ammissibile in uno spazio di dimensione inferiore. Sempre considerando l'esempio precedente, la regione ammissibile del problema originale è un sottoinsieme proprio di  $\mathbb{R}^2$  ed ha quindi dimensione 2. La regione del problema in forma standard pur essendo rappresentata nello spazio  $\mathbb{R}^5$  ha anch'essa dimensione 2 dato che la presenza di tre equazioni proietta tale regione sull'intersezione di tre iperpiani, ossia in un sottospazio di dimensione 2.

#### 4.2.1 Soluzioni del problema PL in forma standard

Per procedere alla definizione della versione algebrica dell'algoritmo del simplex è importante stabilire una relazione tra le soluzioni del problema originale e le soluzioni del problema in forma standard su cui lavoreremo.

In particolare, data una soluzione ammissibile del problema in forma standard questa è un vettore  $x \in \mathbb{R}^n$  e per ottenere la soluzione corrispondente a questa per il problema originale è sufficiente eliminare le  $m$  variabili aggiunte nella trasformazione. Considerando il problema (4.8)-(4.11) è facile verificare che il vettore  $x = (3, 2, 1, 4, 5)$  è una soluzione ammissibile delle equazioni (4.8)-(4.10). Eliminando da tale soluzione le ultime tre componenti, che sono state aggiunte per trasformarlo in forma standard, si ottiene il vettore  $x' = (3, 2)$  che è facile verificare sia una soluzione ammissibile del sistema di disequazioni (4.3)-(4.5). Si noti che in tali ragionamenti consideriamo solo i vincoli propri dei problemi tralasciando per il momento i vincoli di non negatività delle variabili decisionali che saranno considerati implicitamente in seguito.

### Soluzioni Aumentate

Dovendo lavorare sulla formulazione del problema in forma standard è importante caratterizzare le soluzioni di tale problema che corrispondono alle soluzioni del problema originale. Infatti, nel seguito vedremo come definire le soluzioni del problema in forma standard che corrispondono ai vertici del poliedro che come abbiamo visto assumono un ruolo fondamentale per la soluzione del problema PL. Per comodità faremo riferimento ad un problema originale espresso in forma canonica o comunque con solo disequazioni come vincoli, facendo quindi riferimento ad una formulazione originale del problema che utilizzi il minor numero di variabili).

**Definizione 4.1 (Soluzione Aumentata).** *Dato un problema  $P' = \min\{c'^T x, x' \in \mathbb{R}^t, A'x' \geq d, x' \geq 0\}$  espresso in forma canonica, con  $t$  variabili decisionali ed  $m$  vincoli, ed il suo problema equivalente  $P = \min\{c^T x, x \in \mathbb{R}^n, Ax = d, x \geq 0\}$  espresso in forma standard, con  $n = t + m$  variabili decisionali ed  $m$  vincoli, si dice soluzione aumentata la soluzione  $x$  di  $P$  corrispondente ad una soluzione  $x'$  di  $P'$ .*

Dalla definizione è facile ricavare tutte le componenti di  $x$  corrispondenti ad una generica soluzione ammissibile  $x'$  di  $P'$ . Si osservi che una soluzione  $x'$  è un vettore di  $t$  componenti per cui al fine di ricavare  $x$  è sufficiente ricavare le  $m$  componenti di tale vettore ottenute imponendo uguali a quelle di  $x'$  le componenti corrispondenti in  $x$ . Si noti che nella trasformazione tali componenti sono generalmente le prime  $t$  di  $x$  dato che le  $m$  variabili ausiliarie sono aggiunte in seguito alla trasformazione in forma standard.

Il sistema  $Ax = d$  è un sistema di  $m$  equazioni in  $n$  incognite con  $n > m$  ed è noto che tale sistema ammette in generale infinite soluzioni. Ciò non deve sorprendere dato che in effetti le soluzioni del problema originario sono anch'esse in generale infinite. Va però osservato che fissando  $t$  degli  $n$  valori delle incognite il sistema si riduce ad un sistema di  $m$  equazioni in  $m$  incognite la cui soluzione è in generale unica. In altre parole risolvendo tale sistema si determinano le rimanenti componenti dell'unico vettore  $x$  corrispondente alla soluzione  $x'$ .

Per illustrare la trasformazione e la costruzione della soluzione aumentata consideriamo sempre l'esempio del problema della produzione di sedie (4.3)-

(4.5)<sup>3</sup>, limitandoci come osservato ai soli vincoli propri del problema, e la sua formulazione equivalente in forma standard (4.8)-(4.10). Data una soluzione ammissibile per il problema originale  $x = (3, 2)$  possiamo sostituire tali valori alle variabili  $x_1$  e  $x_2$  nel sistema di equazioni (4.8)-(4.10), ottenendo il sistema

$$3 + x_3 = 4 \quad (4.12)$$

$$2 + x_4 = 6 \quad (4.13)$$

$$9 + 4 + x_5 = 18 \quad (4.14)$$

da cui si ricava

$$+x_3 = 4 - 3 = 1 \quad (4.15)$$

$$+x_4 = 6 - 2 = 4 \quad (4.16)$$

$$+x_5 = 18 - 13 = 5 \quad (4.17)$$

risolvendo il quale si ottiene  $x = (3, 2, 1, 4, 5)$ .

### Caratterizzazione dei vertici della regione ammissibile

Avendo definito le soluzioni aumentate vediamo ora come individuare le soluzioni aumentate del problema  $P = \min\{c^T x, x \in \Re^n, Ax = d, x \geq 0\}$ , che corrispondono ai vertici del poliedro del problema  $P' = \min\{c^T x', x' \in \Re^t, A'x' \geq d, x' \geq 0\}$  da risolvere. Si noti che le soluzioni del problema  $P$  sono i vettori  $x$  che soddisfano il sistema di equazioni  $Ax = d$  ed hanno componenti tutte non negative. Chiaramente considerando le sole equazioni le relative soluzioni possono avere anche valori negativi ma solo le soluzioni tutte non negative sono soluzioni ammissibili per  $P$ . Nel seguito considereremo in esplicito solo il sistema di equazioni e tratteremo i vincoli non negatività in modo implicito.

Esaminando il sistema di equazioni  $Ax = d$  sappiamo che la sua risolubilità dipende strettamente dalle proprietà della matrice  $A$ . Nel nostro caso, premetteremo le seguenti assunzioni.

**Definizione 4.2.** *Dato un problema PL,  $P = \min\{c^T x, x \in \Re^n, Ax = d\}$ , espresso in forma standard, assumiamo che la matrice  $A$  con  $m$  righe ed  $n$  colonne sia:*

- rettangolare con  $n > m$
- di rango  $m$ , ossia contenga almeno una sotto-matrice quadrata di dimensione  $m$  non singolare.

La prima di tali assunzioni deriva dal fatto che  $P$  essendo ottenuto da un problema  $P'$  con  $m$  disequazioni avrà necessariamente un numero di variabili superiore al numero di vincoli. Nel caso in cui  $P'$  fosse già espresso in forma

---

<sup>3</sup>Si osservi che la formulazione originale di tale problema non è strettamente in forma canonica ma che comunque tale formulazione contiene solo disequazioni di tipo  $\leq$ .

standard, e quindi  $P' = P$  riterremo comunque che tale assunzione sia verificata. Infatti, qualora  $n < m$  avremo a che fare con un sistema con più equazioni rispetto alle incognite che in generale non ammette soluzione e questo corrisponderebbe ad un problema sovra-vincolato. Nel caso in cui  $n = m$ , se la matrice  $A$  ha rango  $m$  il sistema ammette una singola soluzione e quindi avremmo a che fare con un problema banale con una sola soluzione. Il caso in cui  $n > m$ , corrisponde ad un sistema di equazioni che ammette infinite soluzioni e questo è ovviamente il caso che ci interessa considerare per il problemi PL.

La seconda assunzione prevede che il sistema di equazioni ammetta soluzioni. Se il rango di  $A$  non è pieno il sistema non ammette soluzioni ed almeno uno dei vincoli risulta essere combinazione lineare degli altri. Vedremo comunque come poter verificare, nel caso in cui il problema non ammetta soluzione, la violazione di tali assunzioni in un problema PL da risolvere.

Le infinite soluzioni del sistema  $Ax = d$  possono essere ottenute scegliendo  $t = n - m$  variabili ed assegnando a queste dei valori arbitrari. Di conseguenza il sistema si riduce ad un sistema di  $m$  equazioni in  $m$  incognite  $Bx_B = d$ , dove  $x_B$  è il vettore delle  $m$  incognite residue. Se  $B$  ha rango  $m$  il sistema può essere risolto fornendo il valore delle incognite residue che unite ai valori arbitrari di quelle eliminate restituisce la corrispondente soluzione  $x$ . Si osservi che il fatto che scelte le variabili da sostituire a queste assegniamo valori arbitrari e per ciascuna di queste otteniamo una soluzione ha come conseguenza che le soluzioni di  $Ax = d$  sono infinite (come del resto lo sono le soluzioni del problema PL originario).

Nel seguito i termini variabile decisionale e colonna del sistema di equazione saranno considerati sinonimi, così come i termini vincolo del problema e riga del sistema di equazioni.

A titolo di esempio consideriamo il sistema

$$x_1 + x_3 = 4 \quad (4.18)$$

$$x_2 + x_4 = 6 \quad (4.19)$$

$$3x_1 + 2x_2 + x_5 = 18 \quad (4.20)$$

Se sceglieremo le variabili  $x_4$  e  $x_5$  ed assegniamo a queste i valori 4 e 5, rispettivamente, otteniamo un sistema nelle incognite  $x_1$ ,  $x_2$  e  $x_3$  che, risolto, fornisce la soluzione  $(3, 2, 1)$  da cui ricaviamo la soluzione  $x = (3, 2, 1, 4, 5)$ . Se invece assegniamo alle incognite  $x_3$  e  $x_4$  il valore 0, la soluzione corrispondente sarà  $(4, 6, 0, 0, -6)$ . Si noti che questa soluzione viola le condizioni di non negatività delle incognite (infatti  $x_5 = -6 < 0$ ) e non è pertanto soluzione ammissibile di  $P$  pur essendo una soluzione del sistema  $Ax = d$ .

La figura 4.3 riporta alcuni punti del poliedro del problema di produzione di sedie e le relative soluzioni aumentate ottenute sostituendo le coordinate del punto alle variabili  $x_1$  e  $x_2$  in (4.18)–(4.20) risolvendo il sistema di equazioni risultante. Osservando la figura si può notare che:

- punti interni alla regione ammissibile corrispondono a soluzioni aumentate con tutte le componenti strettamente positive (es.  $(3, 2) \rightarrow (3, 2, 1, 4, 5)$ );

- punti esterni alla regione ammissibile corrispondono a soluzioni aumentate con almeno una componente strettamente negativa (es.  $(4, 6) \rightarrow (4, 6, 0, 0, -6)$ );
- punti appartenenti ad un lato della regione ammissibile corrispondono a soluzioni aumentate con una componente uguale a zero (es.  $(4, 2) \rightarrow (4, 2, 0, 4, 2)$ );
- punti corrispondenti a vertici della regione ammissibile corrispondono a soluzioni aumentate con  $t = n - m = 2$  componenti uguale a zero (es.  $(4, 3) \rightarrow (4, 3, 0, 3, 0)$ ).

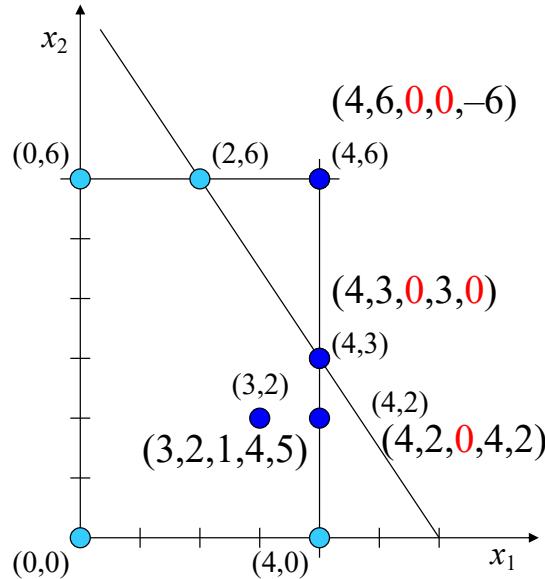


Figura 4.3: Soluzioni aumentate corrispondenti a punti del poliedro per il problema di produzione di sedie.

Le considerazioni precedenti discendono dal ruolo delle variabili nel sistema di vincoli del problema. Infatti ciascuna variabile del problema rappresenta lo slack (ossia la distanza) tra le coordinate di un punto e l'equazione associata al vincolo originario del problema. Ad esempio la variabile  $x_3$  è lo slack del vincolo  $x_1 \leq 4$  che in forma standard diventa  $x_1 + x_3 = 4$ . Dato un punto di coordinate  $(x_1, x_2)$  lo slack corrispondente è:

- strettamente positivo se  $x_1 < 4$ , ossia se il vincolo  $x_1 \leq 4$  è rispettato in modo stretto;
- nullo se  $x_1 = 4$ , ossia se il vincolo  $x_1 \leq 4$  è rispettato all'uguaglianza;

- strettamente negativo se  $x_1 > 4$ , ossia se il vincolo  $x_1 \leq 4$  è violato.

Si noti che anche le variabili  $x_1$  e  $x_2$  possono essere interpretate come slack dei rispettivi vincoli di non negatività. Da queste osservazioni consegue che se una soluzione aumentata contiene una variabile uguale a zero allora il punto corrispondente giace sulla equazione di supporto del vincolo di cui tale variabile è lo slack. Ad esempio, essendo  $x_3$  lo slack del vincolo  $x_1 \leq 4$  ogni soluzione aumentata con  $x_3 = 0$  corrisponde a punti che giacciono sulla retta  $x_1 = 4$ . Poiché un vertice è un punto della regione ammissibile che si trova all'intersezione di  $t$  vincoli di questa, le corrispondenti soluzioni aumentate devono avere  $t$  componenti uguali a zero per imporre che le coordinate della soluzione originale si trovino all'intersezione di  $t$  vincoli del problema.

### Soluzioni Base del problema PL in forma standard

Da quanto discusso precedentemente possiamo ricondurre la definizione delle soluzioni aumentate corrispondenti ai vertici (ammissibili e non ammissibili) della regione ammissibile all'enumerazione delle soluzioni ottenute imponendo a zero  $t$  variabili del problema in forma standard. Si può dimostrare che in questo modo si costruiscono tutte le soluzioni aumentate corrispondenti ai vertici di  $P$ . In modo del tutto equivalente si possono però scegliere le  $m = n - t$  variabili del problema che "sopravvivono" a tale eliminazione. Si noti che ponendo le altre  $t$  variabili uguali a zero queste "scompaiono" dal sistema  $Ax = d$  lasciando attive solo le  $m$  variabili scelte. Compattando il sistema eliminando le colonne corrispondenti alle variabili poste uguali a zero rimane il sistema  $Bx_B = d$  dove

- $B$  è la matrice quadrata  $m \times m$  ottenuta da  $A$  eliminando le colonne corrispondenti alle variabili poste a zero;
- $x_B$  è un vettore di  $m$  incognite corrispondente alle variabili rimaste nel problema.

È ben noto che il sistema  $Bx_B = d$  è risolubile se la matrice  $B$  è invertibile, ossia se le sue colonne sono tra loro linearmente indipendenti. In caso contrario il sistema  $Bx_B = d$  non è risolubile e questo è legato al fatto che almeno una delle variabili residue può essere ottenuta come combinazione lineare delle altre e può essere pertanto sostituita nel problema da tale combinazione ed eliminata. Se la matrice  $B$  è invertibile si ha infatti

$$B^{-1}Bx_B = B^{-1}d \Rightarrow x_B = B^{-1}d. \quad (4.21)$$

Possiamo quindi introdurre la seguente definizione:

**Definizione 4.3** (Base). *Dato un sistema di equazioni  $Ax = d$  con  $m$  equazioni in  $n$  incognite con  $n > m$ , una base è una collezione  $\mathcal{B} = \{A_{\beta(1)}, \dots, A_{\beta(m)}\}$  di colonne tra loro linearmente indipendenti, dove  $\beta(1), \dots, \beta(m)$  sono gli indici delle colonne scelte.*

Si noti che l'operazione di eliminazione delle colonne che non appartengono alla base scelta comporta in pratica una rinumerazione delle variabili rispetto al problema originario. Infatti delle  $n$  colonne della matrice  $A$  del problema in forma standard, che hanno quindi indici compresi tra 1 ed  $n$ , solo  $m$  sono presenti nel problema  $Bx_B = d$  e tali colonne nella matrice  $B$  hanno indici tra 1 ed  $m$ . Ad esempio se consideriamo il sistema (4.18)–(4.20) la corrispondente matrice  $A$  è

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 3 & 2 & 0 & 0 & 1 \end{bmatrix}. \quad (4.22)$$

Scegliendo la base  $\mathcal{B} = \{A_1, A_2, A_4\}$  e quindi definendo  $\beta(1) = 1, \beta(2) = 2$  e  $\beta(3) = 4$  ed imponendo  $x_3 = x_5 = 0$ , si ottiene la matrice

$$B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 3 & 2 & 0 \end{bmatrix}$$

di cui è facile verificare la lineare indipendenza delle colonne di  $B$  e la cui inversa è

$$B^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ -3/2 & 0 & 1/2 \\ 3/2 & 1 & -1/2 \end{bmatrix}$$

da cui si ricava

$$x_B = B^{-1}d = \begin{bmatrix} 1 & 0 & 0 \\ -3/2 & 0 & 1/2 \\ 3/2 & 1 & -1/2 \end{bmatrix} \cdot \begin{bmatrix} 4 \\ 6 \\ 18 \end{bmatrix} = \begin{bmatrix} 4 \\ 3 \\ 3 \end{bmatrix}.$$

La soluzione aumentata corrispondente alla base  $\mathcal{B} = \{A_1, A_2, A_4\}$  è quindi  $x = (4, 3, 0, 3, 0)$  e corrisponde al vertice  $(4, 3)$  del problema originario (si veda la figura 4.3).

Le soluzioni aumentate corrispondenti alle basi del sistema  $Ax = d$  sono dette *soluzioni base* e sono definite come segue.

**Definizione 4.4** (Soluzione base). *Dato un sistema di equazioni  $Ax = d$  con  $m$  equazioni in  $n$  incognite con  $n > m$  e una sua base  $\mathcal{B} = \{A_{\beta(1)}, \dots, A_{\beta(m)}\}$  la corrispondente soluzione base è*

$$x_j = \begin{cases} 0 & \text{se } A_j \notin \mathcal{B} \\ k\text{-sima componente di } B^{-1}d & \text{se } j = \beta(k) \end{cases}$$

Si osservi che una soluzione base è soluzione del sistema  $Ax = d$  ma non necessariamente rispetta le condizioni di non negatività  $x \geq 0$  che sono però parte integrante del problema PL. Possiamo quindi introdurre la seguente ulteriore definizione

**Definizione 4.5** (Soluzione base ammissibile). *Dato una soluzione base  $x$  questa è una soluzione base ammissibile (SBA) se e solo se  $x \geq 0$ .*

La relazione tra soluzioni base ammissibili e vertici del politopo sono enunciate nel seguente teorema di cui si omette la dimostrazione per brevità.

**Teorema 4.1.** *Una soluzione aumentata  $x$  è un vertice del politopo  $F = \{x \in \mathbb{R}^n, Ax = d, x \geq 0\}$  se e solo se  $x$  è una soluzione base ammissibile del sistema  $Ax = d$ .*

Vediamo ora con maggiore dettaglio il ruolo delle soluzioni base rispetto al sistema di equazioni  $Ax = d$  che definiscono i vincoli del problema PL. Per una maggiore semplicità di notazione, data una base  $\mathcal{B}$ , possiamo permutare le colonne del sistema in modo che le variabili appartenenti alla base siano le prime  $m$ . Tale permutazione in pratica è equivalente ad una rinumerazione delle colonne del problema definendo un problema equivalente che avrà una forma più comoda dal punto di vista della notazione. Avremo dunque

$$A = [\underbrace{A_1, \dots, A_m}_{\text{in base}} | \underbrace{A_{m+1}, \dots, A_n}_{\text{fuori base}}] = [B|F] \quad (4.23)$$

dove  $B$  è una matrice quadrata  $m \times m$  nonsingolare ed  $F$  è una matrice rettangolare  $m \times n - m$  corrispondente alle variabili fuori base. Il sistema di equazioni può quindi essere riscritto come

$$Ax = d \Rightarrow [B|F] \begin{bmatrix} x_B \\ x_F \end{bmatrix} = d \Rightarrow Bx_B + Fx_F = d, \quad (4.24)$$

in cui  $x_B$  ed  $x_F$  sono rispettivamente le variabili in base e fuori base. Da tale espressione è possibile riscrivere il sistema in forma canonica rispetto alla base corrente esprimendo le variabili base rispetto alle variabili fuori base. In particolare, portando nell'espressione (4.24) il termine con  $x_F$  al secondo membro e moltiplicando ambo i membri per  $B^{-1}$  si ottiene

$$x_B = B^{-1}d - B^{-1}Fx_F. \quad (4.25)$$

Si osservi che il sistema in forma canonica è una riscrittura del sistema di equazioni originario esplicitando alcune delle variabili (nel nostro caso quelle base) ed esprimendole in funzione del valore delle variabili rimanenti. Da tale espressione è possibile ricavare il valore delle  $x_B$  in funzione dei valori arbitrari che potremmo attribuire alle  $x_F$  per costruire una soluzione generica del sistema di equazioni come discusso precedentemente. Nel nostro caso siamo però interessati a costruire le soluzioni base del sistema che corrispondono alla scelta  $x_F = 0$  da cui si ottiene  $x_B = B^{-1}$ . L'espressione (4.25) sarà però utile nel seguito per caratterizzare punti anche diversi dai vertici della regione ammissibile osservando che in questo caso alcune delle variabili fuori base potrebbero assumere valori diversi da zero.

### 4.3 Versione algebrica dell'algoritmo

Abbiamo ora tutti gli elementi per definire la versione algebrica dell'algoritmo del simplex che si basa sull'enumerazione delle soluzioni base ammissibili del

sistema dei vincoli, che abbiamo visto essere equivalente all'enumerazione dei vertici della regione ammissibile.

---

**Algorithm 2** Versione algebrica dell'algoritmo del simplesso.

---

```

1: procedure SIMPLEX( $(c, A, d)$ )
2:   sia  $\mathcal{B} = \{A_{\beta(1)}, \dots, A_{\beta(m)}\}$  una base del sistema  $Ax = d$ 
   corrispondente ad un vertice ammissibile;                                 $\triangleright$  Inizializzazione
3:    $ottimo \leftarrow false$ ;
4:   repeat
5:     calcola la SBA,  $x$ , corrispondente a  $\mathcal{B}$ ;
6:     if  $x$  è ottimo then                                $\triangleright$  Test di ottimalità
7:        $ottimo \leftarrow true$ ;
8:     else
9:       Aggiorna  $\mathcal{B}$  in modo che corrisponda
      ad un vertice adiacente migliore;                            $\triangleright$  Spostamento
10:    end if
11:   until  $ottimo = true$ 
12:   return  $x$  (soluzione ottima)
13: end procedure

```

---

Si noti che la versione algebrica esposta è ancora definita in modo approssimativo dato che non viene dettagliato come realizzare i passi di Inizializzazione (linea 2), Test di ottimalità (linea 6) e di spostamento ad un'altra SBA (linea 9).

## 4.4 Test di Ottimalità

Data la base corrente  $\mathcal{B} = \{A_{\beta(1)}, \dots, A_{\beta(m)}\}$  e la corrispondente soluzione base ammissibile  $x$ , per verificarne l'ottimalità è sufficiente provare che tale soluzione è un ottimo locale rispetto ad un intorno euclideo, ossia verificare che non esistono in prossimità di tale punto soluzioni ammissibili migliori.

Data la base corrente l'espressione (4.25) ci permette di calcolare il valore delle variabili base in funzione di quelle fuori base. In un vertice le variabili  $x_F$  valgono 0 per imporre che la soluzione aumentata si trovi all'intersezione di un certo numero di vincoli che definiscono il vertice. Se una o più variabili  $x_F$  assumono valori positivi piccoli, la soluzione aumentata determinata utilizzando la (4.25) corrisponderà a punti ammissibili vicini al vertice. Calcolando il valore di tali soluzioni possiamo verificare se queste siano migliori o peggiori della soluzione associata al vertice e quindi determinare se questo sia la soluzione ottima o meno.

Esaminiamo dapprima l'esempio del problema della produzione di sedie e supponiamo di trovarci ad una iterazione generica dell'algoritmo in cui la base corrente sia  $\mathcal{B} = \{A_1, A_2, A_4\}$  corrispondente al vertice  $C = (4, 3)$  del problema originale come illustrato nella figura 4.4. Infatti essendo  $x_3 = x_5 = 0$  la soluzione base corrisponde all'intersezione dei vincoli di cui tali variabili sono gli slack,

ossia il primo ed il terzo vincolo del problema. Il sistema espresso in forma canonica rispetto alla base corrente è

$$x_1 = 4 - x_3 \quad (4.26)$$

$$x_2 = 3 + 3/2x_3 - 1/2x_5 \quad (4.27)$$

$$x_4 = 3 - 3/2x_3 + 1/2x_5 \quad (4.28)$$

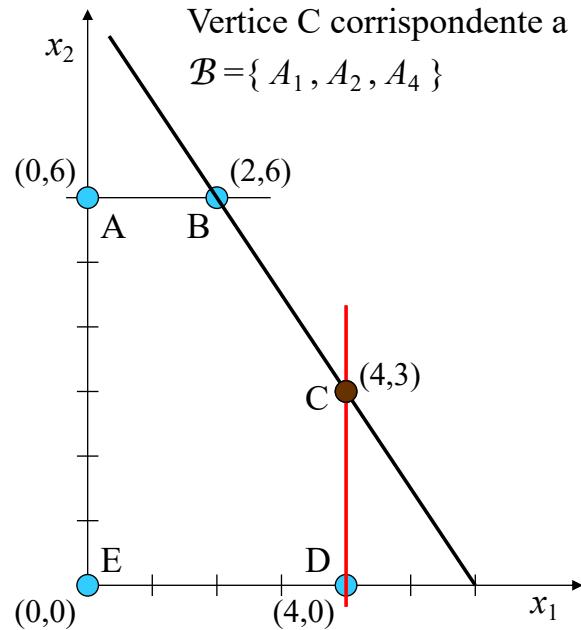


Figura 4.4: Verifica dell'ottimalità del vertice  $C$ .

La SBA corrispondente alla base è  $x = (4, 3, 0, 3, 0)$  il cui valore data la funzione obiettivo  $-\min z = -3x_1 - 5x_2$  è pari a  $-27$ . Se sostituiamo i valori delle variabili base nella funzione obiettivo esprimiamo anche quest'ultima in funzione della base corrente ottenendo

$$\begin{aligned} -z &= -3x_1 - 5x_2 = -3(4 - x_3) - 5(3 + 3/2x_3 - 1/2x_5) = \\ &= -27 - 9/2x_3 + 5/2x_5. \end{aligned} \quad (4.29)$$

L'espressione (4.29) esprime dunque la funzione obiettivo rispetto alla base corrente. Se poniamo  $x_F = (0, 0)$  sappiamo che stiamo imponendo che  $x_B$  definisca la soluzione aumentata corrispondente al vertice  $C$  e l'espressione (4.29) restituisce correttamente il valore  $z = 27$ . Se invece aumentiamo<sup>4</sup> una delle

<sup>4</sup>Si osservi che le variabili fuori base non possono assumere valori negativi perché questi corrisponderebbero sicuramente a soluzioni aumentate non ammissibili avendo componenti negative.

variabili fuori base la soluzione aumentata corrisponderà a punti prossimi al vertice e l'espressione (4.29) restituisce il nuovo valore della funzione obiettivo. Se, ad esempio, aumentassimo  $x_5$  mantenendo  $x_3 = 0$ , ci muoveremmo lungo il primo vincolo,  $x_1 + x_3 = 4$ , verso il vertice  $D$  e dalla (4.29) possiamo ricavare che tali soluzioni hanno valore  $z = -27 + 5/2x_5$  e sono quindi peggiori della soluzione corrente (vertice  $C$ ) qualsiasi sia il valore di  $x_5 > 0$ . Se viceversa aumentassimo  $x_3$  mantenendo  $x_5 = 0$ , ci muoveremmo lungo il terzo vincolo verso il vertice  $B$  e dalla (4.29) possiamo ricavare che tali soluzioni hanno valore  $z = -27 - 9/2x_3$  e sono quindi migliori della soluzione corrente qualsiasi sia il valore di  $x_3 > 0$ . Queste considerazioni ci portano a due conclusioni:

- il vertice  $C$  non è ottimo locale perché esistono punti ammissibili vicini cui corrisponde una soluzione migliore rispetto a quella calcolata nel vertice (ad esempio i punti lungo il lato  $\overline{CB}$  della regione ammissibile);
- muovendosi lungo il lato  $\overline{CB}$ , ossia aumentando  $x_3$  con  $x_5 = 0$ , incontriamo soluzioni ammissibili via via sempre migliori finché non raggiungiamo il vertice  $B$  oltre il quale non possiamo spostarci senza uscire dalla regione ammissibile.

Possiamo ora generalizzare le osservazioni intuitive fin qui fatte esaminando l'esempio. Data la base corrente  $\mathcal{B} = \{A_1, \dots, A_m\}$  possiamo esprimere il sistema di vincoli  $Ax = d$  in forma canonica rispetto alla base esplicitando le variabili base  $x_B$  in funzione delle variabili fuori base  $x_F$  come indicato nella (4.25). Per valutare l'ottimalità della base corrente dobbiamo prendere in considerazione la funzione obiettivo del problema ed in questa esplicitare il contributo dato dalle variabili base e fuori base. In particolare,

$$c^T x = [c_B^T \quad c_F^T] \begin{bmatrix} x_B \\ x_F \end{bmatrix} = c_B^T x_B + c_F^T x_F, \quad (4.30)$$

da cui, sostituendo ad  $x_B$  l'espressione (4.25), si ottiene

$$c^T x = c_B^T B^{-1}d - c_B^T B^{-1}F x_F + c_F^T x_F = \underbrace{c_B^T B^{-1}d}_{\text{costante}} + \underbrace{(c_F^T - c_B^T B^{-1}F)}_{\text{costo ridotto}} x_F. \quad (4.31)$$

Esaminando la (4.31) si può osservare che il primo termine non è altro che il valore della funzione obiettivo calcolato in corrispondenza del vertice associato alla base corrente. Infatti  $c_B^T B^{-1}d$  non è altro che il prodotto del costo delle variabili base moltiplicato per il valore di tali variabili nella soluzione base, ossia  $B^{-1}d$ . Inoltre, in corrispondenza del vertice le  $x_F = 0$  e quindi il secondo termine è nullo. L'espressione (4.31) però fornisce il valore della funzione obiettivo anche in corrispondenza di soluzioni non base, ossia in cui le  $x_F$  non sono tutte nulle. In questo caso il segno dei coefficienti del vettore  $\tilde{c} = c_F^T - c_B^T B^{-1}F$ , che è facile verificare abbia  $n-m$  elementi uno per ogni variabile  $x_F$ , definisce le conseguenze sul valore di tale funzione dell'aumento delle variabili fuori base. Il vettore  $\tilde{c}$  è noto come vettore dei *costi ridotti* o dei *costi residui*. Ricordando che stiamo considerando un problema di minimizzazione possiamo osservare che:

- se esiste una variabile fuori base  $j$  con  $A_j \notin \mathcal{B}$  il cui costo ridotto  $\tilde{c}_j$  è negativo allora il vertice corrispondente alla base corrente non è ottimo e facendo aumentare il valore della variabile  $j$  il costo della soluzione migliora (ossia diminuisce).
- se per tutte le variabili fuori base  $j$  con  $A_j \notin \mathcal{B}$  il costo ridotto  $\tilde{c}_j$  è positivo o nullo allora il vertice corrispondente alla base corrente è ottimo dato che facendo aumentare il valore della variabile  $j$  il costo della soluzione non migliora (ossia aumenta o rimane uguale).

I costi ridotti possono essere interpretati come derivate direzionali della funzione obiettivo rispetto alla variazione delle variabili fuori base. Considerando infatti la (4.31) e facendone la derivata rispetto ad una generica variabile fuori base  $x_j$  si ottiene:

$$\frac{\partial c^T x}{\partial x_j} = \frac{\partial(c_B^T B^{-1} d x_B + (c_F^T - c_B^T B^{-1} F) x_F)}{\partial x_j} = 0 + \tilde{c}_j. \quad (4.32)$$

Possiamo quindi enunciare il seguente

**Teorema 4.2** (Condizione di ottimalità). *Dato un problema PL,  $P = \min\{c^T x, x \in \Re^n, Ax = d\}$  ed una base  $\mathcal{B} = \{A_{\beta(1)}, \dots, A_{\beta(m)}\}$ , la corrispondente soluzione base  $(x_B, 0)$  con  $x_B = B^{-1}d \geq 0$  è ottima se e solo se i costi ridotti delle variabili fuori base  $\tilde{c} = c_F^T - c_B^T B^{-1} F$  sono tutti non negativi.*

Ovviamente se la base corrente non è ottima bisognerà determinare una nuova base ed i costi ridotti danno indicazioni utili al fine di individuare variabili fuori base che sia conveniente sostituire a quelle attualmente nella base. Infatti se una variabile fuori base  $j$  a costo ridotto negativo attualmente a 0 aumenta il suo valore fino  $x_j = \vartheta > 0$  la funzione obiettivo cambia di  $\tilde{c}_j \vartheta < 0$  ossia migliora.

Si noti che il costo ridotto delle variabili base è nullo. Infatti il vettore dei costi ridotti è

$$\tilde{c}^T = c^T - c_B B^{-1} A = \underbrace{[c_B^T - c_B B^{-1} B]}_{=0} | \underbrace{c_F^T - c_B B^{-1} F}_{=\tilde{c}_F^T}. \quad (4.33)$$

Osserviamo infine che se ad una variabile fuori base  $x_j$  corrisponde un costo ridotto nullo questo implica che aumentando il valore di tale variabile il valore della funzione obiettivo non cambia e di conseguenza il vertice che si trova all'estremo del lato della regione ammissibile associato a tale variabile ha lo stesso costo di quello da cui si proviene. Questa situazione è stata introdotta nel paragrafo 3.3.5 in cui sono state discusse le soluzioni ottime equivalenti.

## 4.5 Spostamento da SBA a SBA (Pivoting)

Nel caso in cui la SBA corrispondente alla base  $\mathcal{B} = \{A_{\beta(1)}, \dots, A_{\beta(m)}\}$  corrente non sia ottima esiste almeno una variabile  $x_j$  tale che  $A_j \notin \mathcal{B}$  tale che  $\tilde{c}_j < 0$ . A questo punto l'algoritmo del simplexso prevede di “spostarsi”

Supponiamo per semplicità che  $x_j$  sia la sola variabile fuori base in tali condizioni. Esamineremo infatti in seguito come scegliere tale variabile nel caso ci sia più di una variabile a costo ridotto negativo.

Dato che la variabile  $x_j$  ha costo ridotto negativo è conveniente farla assumere un valore positivo facendola quindi “entrare in base”. Ricordando quanto discusso nel paragrafo 4.2.1, questo renderebbe la soluzione aumentata non più coincidente con un vertice del poliedro dato che avrebbe meno di  $n - m$  componenti nulle. Occorre quindi definire una nuova base  $\mathcal{B}'$  ottenuta da  $\mathcal{B}$  inserendo la colonna  $A_j$  al posto di una delle colonne attualmente nella base.

**Definizione 4.6** (Basi adiacenti). *Due basi  $\mathcal{B}$  e  $\mathcal{B}'$  sono adiacenti se differiscono per una colonna.*

È evidente che dalla base corrente è possibile costruire  $m$  basi adiacenti ottenute sostituendo  $A_j$  a ciascuna delle  $m$  colonne attualmente nella base. Consideriamo, ad esempio il problema delle sedie esaminato nel paragrafo precedente. In questo caso la base corrente è  $\mathcal{B} = \{A_1, A_2, A_4\}$  corrispondente al vertice  $C = (4, 3)$  del problema originale. Il calcolo dei costi ridotti associati alle variabili fuori base restituisce  $\tilde{c}_3 = -9/2 < 0$  e  $\tilde{c}_5 = 5/2 > 0$  quindi il vertice corrente non è la soluzione ottima del problema e facendo entrare in base la variabile  $x_3$ , ossia muovendosi lungo il lato  $\overline{BD}$  si ottengono soluzioni migliori. Nella figura 4.5 sono indicate le tre basi adiacenti alla base corrente ossia:

- $\mathcal{B} = \{A_1, A_2, A_3\}$  corrispondente al vertice ammissibile  $B$ ,
- $\mathcal{B} = \{A_1, A_3, A_4\}$  corrispondente al vertice non ammissibile  $F$ ,
- $\mathcal{B} = \{A_3, A_2, A_4\}$  corrispondente al vertice ammissibile  $G$ .

Dall'esempio appare evidente che solo una delle tre basi che possono essere costruite a partire dalla base corrente corrisponde ad una SBA mentre le altre corrispondono a soluzioni base non ammissibili (SBnA).

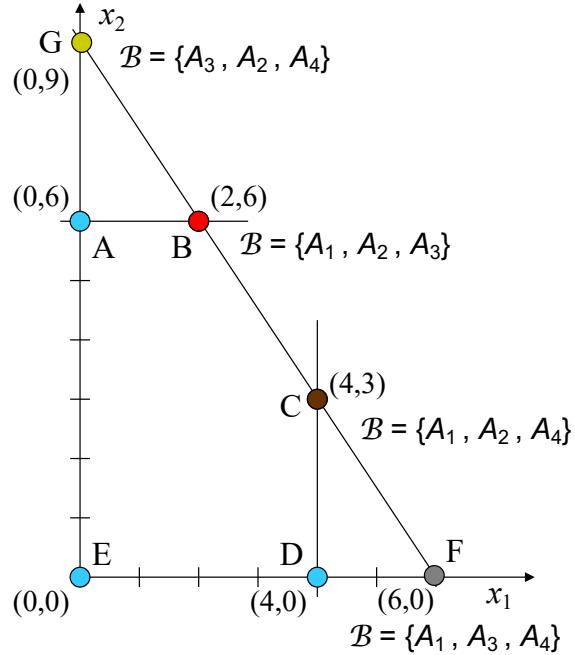


Figura 4.5: Basi Adiacenti.

Chiaramente la costruzione di una base adiacente corrispondente ad un SBnA non è compatibile con l'esecuzione dell'algoritmo del simplex precedentemente descritto per cui vedremo come, rinunciando alla libertà di scegliere quale variabile abbandona la base, definire una base adiacente alla corrente che corrisponda ad una SBA. Il procedimento che esamineremo è definito *pivoting*.

Data la base corrente  $\mathcal{B} = \{A_{\beta(1)}, \dots, A_{\beta(m)}\}$  e la variabile  $x_j$  che deve entrare nella base poiché ha  $\tilde{c}_j < 0$  la nuova base può essere ottenuta considerando la SBA associata e

- mantenendo uguali a zero tutte le variabili fuori base diverse da  $x_j$
- aumentando  $x_j$  il più possibile mantenendo la soluzione base ammissibile.

Per illustrare il procedimento con un esempio concreto riprendiamo la discussione svolta nel paragrafo precedente considerando il problema dell'ottimizzazione della produzione di sedie. A tal proposito ricordiamo che, data la base corrente  $\mathcal{B} = \{A_1, A_2, A_4\}$  corrispondente al vertice  $C = (4, 3)$ , il calcolo dei costi ridotti aveva permesso di stabilire che tale vertice non è la soluzione ottima del problema. Inoltre la variabile fuori base  $x_3$  ha costo ridotto negativo e quindi è vantaggioso far entrare tale variabile nella base. Il sistema di vincoli, in forma canonica rispetto alla base corrente, è dato dalle equazioni (4.26)–(4.28) dalle quali possiamo ricavare le condizioni di ammissibilità della base. Se in tali

equazioni poniamo  $x_5 = 0$ , otteniamo i valori delle variabili attualmente nella base in funzione del valore di  $x_3$ . Affinché la soluzione sia ammissibile tutti i valori delle variabili devono essere maggiori o uguali a zero ossia devono valere le seguenti relazioni:

$$x_1 = 4 \quad -x_3 \geq 0 \quad \Rightarrow \quad x_3 \leq 4 \quad (4.34)$$

$$x_2 = 3 \quad +3/2x_3 \geq 0 \quad \Rightarrow \quad x_3 \geq -2 \quad (4.35)$$

$$x_4 = 3 \quad -3/2x_3 \geq 0 \quad \Rightarrow \quad x_3 \leq 2 \quad (4.36)$$

Da tali relazioni si vede che se  $x_3$  aumenta i valori di  $x_1$  ed  $X_4$  diminuiscono mentre quello di  $x_4$  rimane positivo. Le relazioni (4.34) e (4.36) limitano quindi il massimo aumento di  $x_3$  ed il massimo valore di  $x_3$  compatibile con l'ammissibilità della soluzione è dato dalla relazione (4.36) che impone  $x_3 \leq 2$ . Infatti per valori maggiori di 2 la variabile  $x_4$  diventerebbe negativa e la soluzione non sarebbe più ammissibile. Si noti poi che imponendo  $x_3 = 2$  il valore di  $x_4$  diventa 0 e questo indica che tale variabile “esce” dalla base per effetto dell’ingresso di  $x_3$ . In tal caso la nuova base diventa  $\mathcal{B} = \{A_1, A_2, A_3\}$ . La nuova soluzione base può essere facilmente calcolata dalle relazioni (4.34)–(4.36) ed è  $x = (2, 6, 2, 0, 0)$  corrispondente al vertice  $B = (2, 6)$ . Il sistema può essere messo in forma canonica rispetto alla nuova base con semplici operazioni di sostituzione. In particolare dalla relazione (4.28) possiamo ricavare  $x_3$  in funzione delle variabili fuori base  $x_4$  e  $x_5$ . Possiamo poi sostituire tale relazione nelle (4.26) e (4.27) e nella (4.29) per ottenere anche  $x_1$  e  $x_2$  e la funzione obiettivo in funzione delle variabili fuori base, ottenendo il sistema

$$-z = -36 + 3/2x_4 + x_5 \quad (4.37)$$

$$x_1 = 2 + 1/3x_4 - 1/3x_5 \quad (4.38)$$

$$x_2 = 6 - 1/2x_4 \quad (4.39)$$

$$x_3 = 2 - 1/3x_4 + 1/3x_5 \quad (4.40)$$

Osservando i costi ridotti delle variabili fuori base nella (4.37) possiamo verificare che sono entrambi positivi e quindi il vertice corrente  $B = (2, 6)$  corrisponde alla soluzione ottima del problema il cui valore è  $z = 36$ .

Rivediamo ora l’operazione di pivoting con maggiore precisione. Consideriamo il sistema di equazioni in forma canonica rispetto alla base  $\mathcal{B} = \{A_1, \dots, A_m\}$  espresso dalla (4.25) e supponiamo che la variabile fuori base  $x_h$  sia a costo ridotto negativo e quindi sia selezionata per l’ingresso nella base. Ricordando che le altre variabili fuori base rimangono a valore 0, del prodotto  $B^{-1}F_{xF}$  rimane solo la colonna di  $B^{-1}F$  corrispondente alla variabile  $x_h$  moltiplicata per tale variabile, ossia

$$x_B = B^{-1}d - B^{-1}F_{xF} = B^{-1}d - B^{-1}A_h x_h = \hat{d} - \hat{A}_h x_h \quad (4.41)$$

in cui  $\hat{d}$  e  $\hat{A}$  indicano rispettivamente il vettore  $d$  e la matrice  $A$  moltiplicate per l’inversa della matrice di base corrente  $B^{-1}$ . Per imporre l’ammissibilità della soluzione corrente occorre quindi per ciascuna variabile base imporre che sia

$$x_{\beta(i)} = \hat{d}_i - \hat{A}_{ih} x_h \geq 0 \quad i = 1, \dots, m. \quad (4.42)$$

Nella (4.42) il termine  $\hat{d}_i$  è il valore corrente della variabile base  $x_{\beta(i)}$  ed è quindi maggiore o uguale a zero. Inoltre la variabile  $x_h$  che entra nella base è anch'essa non negativa per cui il soddisfacimento della (4.42) dipende unicamente dal termine  $\hat{A}_{ih}$  per il quale abbiamo due casi possibili:

- se  $\hat{A}_{ih} \leq 0$  la variabile  $x_{\beta(i)}$  rimane positiva qualunque sia il valore di  $x_h \geq 0$ , ossia la condizione non è restrittiva;
- se  $\hat{A}_{ih} > 0$  la condizione è restrittiva e la variabile  $x_{\beta(i)}$  rimane positiva solo se  $x_h \leq \hat{d}_i/\hat{A}_{ih}$ .

Poiché tutte le variabili base devono rimanere  $\geq 0$  bisogna considerare la condizione (4.42) più restrittiva che corrisponde al valore:

$$\vartheta = \min \left\{ \frac{\hat{d}_i}{\hat{A}_{ih}}, i = 1, \dots, m : \hat{A}_{ih} > 0 \right\} = \frac{\hat{d}_\ell}{\hat{A}_{\ell h}}. \quad (4.43)$$

Di conseguenza la variabile  $x_h$  entra in base a valore  $\vartheta$  ed esce dalla base la variabile  $x_{\beta(\ell)}$ . L'elemento della matrice  $\hat{A}$  che si trova in corrispondenza della riga  $\ell$  individuata con la formula (4.43) e della colonna  $h$  scelta per l'ingresso nella base si dice *elemento pivot* o *elemento cardine*.

Si noti che se nelle relazioni (4.42) i coefficienti  $\hat{A}_{ih}$  sono tutti non positivi, allora qualunque sia il valore di  $x_h > 0$  tutte le variabili base rimangono  $\geq 0$  e quindi  $x_h$  può crescere fino a  $+\infty$ . Tale situazione è quindi indicativa del verificare che il problema da risolvere sia illimitato. Nella figura 4.6 è illustrato un esempio di problema con soluzione ottima illimitata. In tale problema infatti la regione ammissibile  $F$  è illimitata ed il gradiente indicato nel caso di funzione obiettivo da massimizzare fa sì che la soluzione ottima sia illimitata. Si osservi che nel caso in cui la funzione obiettivo fosse da minimizzare, pur essendo la regione ammissibile illimitata la soluzione ottima sarebbe limitata ed in questo caso coinciderebbe con l'origine.

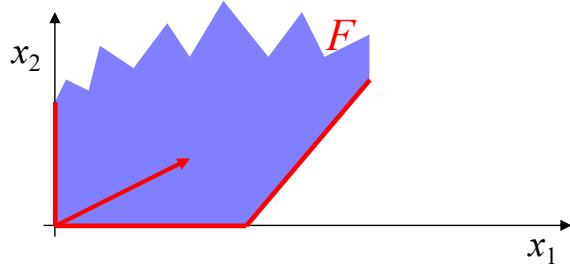


Figura 4.6: Esempio di problema illimitato.

## 4.6 Soluzioni base degeneri

Quanto discusso nella sezione 4.2.1 sulla costruzione delle soluzioni base implica che una base determina univocamente una soluzione base. Pertanto date due SBA diverse  $x_a$  e  $x_b$  allora queste sono corrispondenti a due basi  $\mathcal{B}^1 \neq \mathcal{B}^2$ :

$$x_a \neq x_b \Rightarrow \mathcal{B}_a \neq \mathcal{B}_b. \quad (4.44)$$

Tale implicazione però non vale nel senso opposto ossia date due basi diverse non necessariamente queste corrispondono a SBA diverse

$$\mathcal{B}_a \neq \mathcal{B}_b \not\Rightarrow x_a \neq x_b. \quad (4.45)$$

A titolo di esempio si consideri un problema PL con  $n = 5$  variabili e  $m = 3$  vincoli, la cui matrice tecnologica  $A$  ed il vettore di termini noti  $d$  sono i seguenti:

$$A = \begin{bmatrix} 1 & 2 & 1 & 0 & 0 \\ 0 & 3 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 & 1 \end{bmatrix} \quad d = \begin{bmatrix} 0 \\ 6 \\ 5 \end{bmatrix}. \quad (4.46)$$

Scegliendo la base  $\mathcal{B}_a = \{A_1, A_4, A_5\}$  e quindi definendo  $\beta(1) = 1, \beta(2) = 4$  e  $\beta(3) = 5$  ed imponendo  $x_2 = x_3 = 0$ , si ottiene la matrice

$$B_a = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 2 & 0 & 1 \end{bmatrix} \Rightarrow B_a^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -2 & 0 & 1 \end{bmatrix}$$

da cui si ottiene  $x_a = B_a^{-1}d = (0, 0, 0, 6, 5)$ . Ora, scegliendo una base diversa  $\mathcal{B}_b = \{A_3, A_4, A_5\}$  e quindi definendo  $\beta(1) = 3, \beta(2) = 4$  e  $\beta(3) = 5$  e quindi imponendo  $x_1 = x_2 = 0$ , si ottiene la matrice

$$B_b = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \Rightarrow B_b^{-1} = I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

da cui si ottiene  $x_b = B_b^{-1}d = (0, 0, 0, 6, 5) = x_a$ . Dall'esempio si vede che due basi diverse corrispondono alla stessa soluzione base, in questo caso entrambe ammissibili. Esaminando le SBA  $x_a$  ed  $x_b$  si può osservare che hanno entrambe più di  $n - m = 2$  componenti uguali a zero. Le SBA con questa caratteristica sono dette

**Definizione 4.7.** *SBA degeneri una SBA si dice degenera se contiene più di  $n - m$  componenti uguali a zero*

e vale il seguente teorema

**Teorema 4.3.** *Se 2 basi distinte  $\mathcal{B}_a$  e  $\mathcal{B}_b$  corrispondono alla stessa SBA  $x$ , questa è degenera.*

**Dim.:** Poiché  $x$  è la SBA corrispondente a  $\mathcal{B}_a$  deve avere componenti nulle per le variabili non in tale base e lo stesso deve valere per le variabili fuori base rispetto a  $\mathcal{B}_b$ . Essendo le due basi diverse esiste almeno una variabile che appartiene ad una base ed è fuori base per l'altra e tale variabile nella SBA deve necessariamente avere valore nullo.  $\square$

Dal punto di vista algoritmico la degenerazione della base si verifica quando nella scelta della variabile che entra nella base, effettuato utilizzando la formula (4.43), il valore minimo corrisponde a più di una variabile presente nella base. In tal caso facendo entrare nella base la variabile prescelta a valore  $\vartheta$  più di una variabile della base precedente diventa uguale a zero. Tali variabili sono tutte candidate per l'uscita dalla base e facendo uscire dalla base una di queste, le altre rimarranno in base a valore zero definendo una condizione di degenerazione della base.

In base a quanto fin qui discusso possiamo osservare che, in presenza di degenerazione della base, cambiando la base non è detto che si cambi SBA e quindi che ci si sposti da un vertice ad un altro del poliedro. Infatti è facile verificare che in presenza di una base degenere, nell'eseguire una operazione di pivoting il valore minimo del rapporto  $\frac{\hat{d}_i}{A_{ih}}$  della formula (4.43) corrisponderà ad una delle variabili in base a valore zero, ossia con  $\hat{d}_i = 0$  e di conseguenza si avrà  $\vartheta = 0$  ed il valore della funzione obiettivo non cambierà rispetto all'iterazione precedente. Tale caratteristica è indicativa di una situazione patologica dei problemi PL che può causare un problema di convergenza dell'algoritmo del simplex, dato che può succedere che l'algoritmo cicli indefinitamente tra le basi degeneri corrispondenti al vertice corrente. In questo caso si è in presenza della cosiddetta “degenerazione ciclante” che fortunatamente può essere evitata con opportuni accorgimenti per la scelta delle variabili coinvolte nell'operazione di pivoting, quali la regola di Bland discussa nel seguito.

#### 4.6.1 Interpretazione geometrica della degenerazione

Dal punto di vista modellistico la degenerazione si verifica quando il vertice corrente risulta essere sovradefinito come illustrato nella figura 4.7 in cui il vertice  $A$  risulta essere all'intersezione dei tre vincoli  $v_1, v_2$  e  $v_3$  del problema. Si noti che il problema illustrato ha due variabili decisionali e tre vincoli ed una volta posto in forma standard avrà complessivamente  $n = 5$  variabili decisionali, ossia le due originali e le tre variabili slack dei vincoli, e  $m = 3$  vincoli, oltre a quelli di non negatività delle variabili. Come già discusso un vertice di tale poliedro in  $\mathbb{R}^2$  si trova in generale all'intersezione di  $n - m$  vincoli del problema le cui variabili slack associate avranno valore zero nella corrispondente soluzione aumentata. Nel caso del vertice  $A$  della figura le variabili slack che sono a zero nella soluzione aumentata sono in realtà tre e non solo due come per gli altri vertici del poliedro. Due di tali variabili slack saranno fuori base ma la terza rimarrà in base a valore zero e quindi la soluzione base corrispondente è degenere.

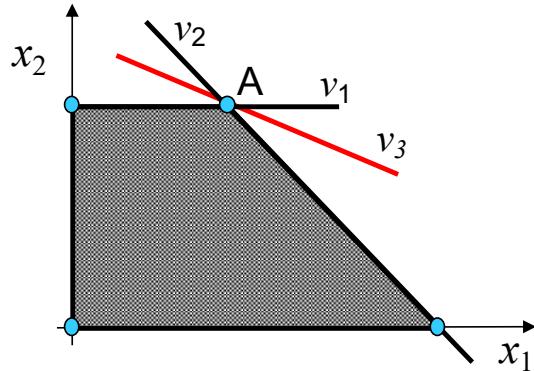


Figura 4.7: Esempio di vertice sovradefinito che causa degenerazione della base.

Come osservato nel paragrafo 3.3.3 il vincolo  $v_3$  è in realtà ridondante e potrebbe essere rimosso senza alterare la regione ammissibile e, di conseguenza, evitando il rischio di degenerazione della base. Purtroppo l'individuazione di vincoli ridondanti, che è immediata dal disegno della regione ammissibile per un problema con due o tre variabili decisionali, non è in generale possibile per problemi generali con molte variabili e vincoli in cui la presenza di vincoli ridondanti è frequente nelle relative formulazioni. In tali casi è quindi spesso inevitabile che in una delle iterazioni dell'algoritmo del simplex si incontri una base degenere.

#### 4.6.2 Regola di Bland

La regola di Bland stabilisce come scegliere le variabili interessate da una operazione di pivoting in modo che sia evitato il rischio di degenerazione ciclante. In pratica garantisce di esaminare le basi in ordine lessicografico in modo da evitare di esaminare due volte la stessa base e quindi evitando i rischi di cicli in caso di degenerazione della base per l'algoritmo del simplex. Data la base corrente Una operazione di pivoting è completamente definita individuando la coppia di variabili coinvolte: la prima è la variabile fuori base scelta per l'ingresso nella base e la seconda è la variabile attualmente in base che deve uscirne. La regola di Bland prevede due passi per definire tali variabili:

1. **Scelta della variabile che entra nella base:** si sceglie la variabile a costo ridotto negativo di **indice minimo**.
2. **Scelta della variabile che esce dalla base:** si sceglie la variabile che minimizza il rapporto  $\frac{\hat{d}_i}{\hat{A}_{ih}}$ ,  $i = 1, \dots, m : \hat{A}_{ih} > 0$  (si veda l'equazione (4.43)). In caso di parità nel calcolo del minimo, **esce dalla base la variabile di indice minimo**.

Più precisamente la variabile  $\beta(\ell)$  che esce dalla base è determinata dalla

$$\beta(\ell) = \min \left\{ \beta(i) : \hat{A}_{ih} > 0, \frac{\hat{d}_i}{\hat{A}_{ih}} \leq \frac{\hat{d}_k}{\hat{A}_{kh}} \quad \forall i, k = 1, \dots, m : \hat{A}_{kh} > 0 \right\}. \quad (4.47)$$

## 4.7 Determinazione della base iniziale: Metodo delle due fasi

Per completare la descrizione dell'algoritmo del simplex vediamo come determinare la base iniziale ammissibile da cui partire. Negli esempi fin qui visti la base iniziale ammissibile era facile da determinare in quanto i problemi esaminati erano caratterizzati da vincoli sotto forma di disequazioni di tipo minore o uguale e termini noti negativi. In tal caso è facile verificare che la base iniziale costituita dalle variabili slack dei vincoli corrisponde ad una soluzione base ammissibile in cui le variabili slack assumono i valori dei corrispondenti termini noti. In generale però non è detto che il problema offra facilmente una base corrispondente ad una SBA da cui partire, sempre che questa esista ossia che il problema non sia impossibile, per cui occorre un metodo generale in grado di determinare tale base in modo automatico. Il metodo che illustreremo è denominato *metodo delle due fasi* ed in una prima fase risolve un problema ausiliario la cui soluzione è la base iniziale che, nella seconda fase, è utilizzata per inizializzare l'algoritmo del simplex.

### 4.7.1 Fase 1

Consideriamo un problema PL di minimizzazione espresso in forma standard  $P = \min\{c^T x, x \in \Re^n, Ax = d, x \geq 0\}$ , e supponiamo che  $d \geq 0$ , eventualmente moltiplicando per  $-1$  i vincoli con termine noto negativo. Nella prima fase dell'algoritmo viene cercata una base, ossia un insieme di  $m$  colonne di  $A$  tra loro linearmente indipendenti, che corrisponda ad una soluzione base ammissibile. Se tale base non è immediatamente disponibile, ad esempio riconoscendo in  $A$  delle colonne che formino una matrice identità, viene definito un problema ausiliario in cui vengono inserite  $m$  variabili aggiuntive  $x^a$  una per ogni vincolo.

$$\begin{cases} Ax = d \\ x \geq 0 \end{cases} \Rightarrow \begin{cases} Ax + Ix^a = d \\ x, x^a \geq 0 \end{cases} \quad (4.48)$$

In tale problema è facile verificare che le variabili  $x^a$  costituiscono una base dato che le colonne a queste corrispondenti sono una matrice identità. Inoltre, dato che  $d \geq 0$  la soluzione base  $x^a = d$  e  $x = 0$  è una soluzione base ammissibile. Ovviamente tale soluzione base non contiene nessuna variabile originale  $x$  per cui non è un punto di partenza per l'algoritmo del simplex. È però facile verificare che una soluzione del problema ausiliario in cui  $x^a = 0$  e  $x \geq 0$  è una soluzione base ammissibile del problema originario.

Per ricercare tale soluzione possiamo utilizzare delle operazioni di pivoting per muoverci dalla soluzione base corrente  $x^a = d$  ad una soluzione base che

contenga solo le variabili originali, ossia in cui  $x^a = 0$ . A tal fine per guidare l'algoritmo alla ricerca di una base che non contenga le variabili  $x^a$  possiamo utilizzare una funzione obiettivo artificiale da minimizzare e che contenga solo tali variabili. Il problema di Fase 1 è quindi:

$$\begin{cases} \min w = x^a \\ Ax + Ix^a = d \\ x, x^a \geq 0 \end{cases} \quad (4.49)$$

Lo scopo di tale problema di ottimizzazione è proprio quello di rimuovere dalla base le variabili  $x^a$  sostituendole con variabili originali  $x$  che non comparendo nella funzione obiettivo artificiale non danno contributo. Il problema artificiale è chiaramente un problema PL di cui disponiamo di una SBA iniziale  $x^a = d$  e che può essere risolto con l'algoritmo del simplex. La risoluzione del problema artificiale di Fase 1 può portare a diversi casi che esamineremo nel dettaglio.

**Caso 1 ( $w > 0$ ):** Se la soluzione ottima del problema di Fase 1 ha un valore della funzione obiettivo positivo,  $w > 0$ , significa che il sistema a destra nell'equazione (4.48) non è risolubile utilizzando le sole  $x$  e quindi il problema originale  $P$  è **impossibile**.

**Caso 2a ( $w = 0$  e base ottima con solo  $x$ ):** Se la soluzione ottima del problema di Fase 1 ha un valore della funzione obiettivo nullo e la base è costituita solo da variabili originali  $x$  allora tale base è una SBA del problema originale e possiamo utilizzarla per inizializzare la Fase 2 dell'algoritmo.

**Caso 2b ( $w = 0$  e base ottima con anche  $x^a$ ):** Se la soluzione ottima del problema di Fase 1 ha un valore della funzione obiettivo nullo e nella base sono ancora presenti delle variabili  $x^a$ , tale base è necessariamente degenere dato che le variabili in base devono avere valore nullo affinché la funzione obiettivo  $w$  valga zero. Per eliminare tali variabili possiamo fare ulteriori operazioni di pivoting progettate per sostituire le variabili  $x^a$  eventualmente presenti in base con variabili originali. A tal fine consideriamo un vincolo del problema in cui sia attualmente in base una variabile  $x^a$ . Supponiamo che tale vincolo sia l' $i$ -simo e che la variabile in base su tale vincolo sia  $x_h^a$ . La corrispondente riga del sistema in forma canonica rispetto alla base corrente è

$$x_h^a = \hat{d}_i - \hat{A}_{ij}x_j \quad (4.50)$$

in cui per semplicità sono rappresentate solo le colonne corrispondenti alle variabili originali e non quelle corrispondenti alle variabili artificiali  $x^a$  che comunque non saranno mai interessate ad ulteriori operazioni di pivoting in questa fase.

Possono presentarsi due ulteriori casi:

**Caso 2b<sub>1</sub> (pivoting efficace):** se sulla riga c'è almeno un coefficiente  $\hat{A}_{ij} \neq 0$  allora può essere eseguita una operazione di pivoting facendo entrare in base la variabile  $x_j$  corrispondente a tale coefficiente. Al termine di tale operazione di

pivoting la variabile  $x_h^a$  uscirà dalla base e sarà sostituita dalla variabile  $x_j$ . Si noti che, diversamente dalle operazioni di pivoting che si eseguono nell'algoritmo del simplexso, in questo caso possono essere considerati anche elementi cardine  $\hat{A}_{ij}$  con valori negativi dato che, essendo la variabile attualmente in base su tale riga uguale a zero questa può anche essere divisa per un valore negativo senza rendere la nuova soluzione base inammissibile (ossia negativa).

**Caso 2b<sub>2</sub> (vincolo ridondante):** se sulla riga tutti i coefficienti  $\hat{A}_{ij}$  sono nulli, allora il vincolo corrispondente a tale riga è combinazione lineare degli altri vincoli del problema e la matrice  $A$  non ha rango pieno come ipotizzato per l'applicazione dell'algoritmo. In tal caso il vincolo può essere rimosso dal problema eliminando in tal modo la riga e la variabile  $x_h^a$  attualmente in base.

In entrambi i casi se tutte le variabili  $x^a$  sono uscite dalla base si può procedere con la Fase 2, altrimenti si eseguono ulteriori operazioni di pivoting del Caso 2b.

#### 4.7.2 Fase 2

Nel caso il problema non sia impossibile (Caso 1) e dopo aver eventualmente eliminato le variabili  $x^a$  ancora presenti nella base ottima degenere di Fase 1, la base corrente è costituita da sole variabili originale e può quindi essere utilizzata per inizializzare l'algoritmo del simplexso. A questo punto possiamo eliminare dal problema le variabili artificiali  $x^a$  aggiunte nella Fase 1, possiamo ripristinare la funzione obiettivo originale del problema e possiamo proseguire con l'esecuzione dell'algoritmo del simplexso sul problema originale.

### 4.8 Versione completa dell'algoritmo del simplexso

Nel seguito riportiamo lo pseudo-codice dell'algoritmo del simplexso algebrico descritto nei paragrafi precedenti. Per brevità è stata omessa la descrizione della Fase 1 eventualmente necessaria alla definizione della base iniziale presentata nel paragrafo 4.7.1.

**Algorithm 3** Versione completa dell'algoritmo del simplex.

---

```

1: procedure SIMPLEX( $(c, A, d)$ )
   Inizializzazione:
2:   sia  $\mathcal{B} = \{A_{\beta(1)}, \dots, A_{\beta(m)}\}$  una base del sistema  $Ax = d$ 
   corrispondente ad una SBA, eventualmente determinata con la Fase 1;
3:   ottimo  $\leftarrow$  false;
4:   illimitato  $\leftarrow$  false;
5:   repeat
6:      $B \leftarrow [A_{\beta(1)}, \dots, A_{\beta(m)}]$ 
7:      $x^* \leftarrow \begin{bmatrix} x_B^* \\ x_F^* \end{bmatrix} = \begin{bmatrix} B^{-1}d \\ 0 \end{bmatrix} = \begin{bmatrix} \hat{d} \\ 0 \end{bmatrix}$   $\triangleright$  calcola la SBA

   Test di ottimalità:
8:      $\tilde{c} \leftarrow c_F^T - c_B^T B^{-1} F$   $\triangleright$  calcola i costi ridotti
9:     if  $\tilde{c}_h = c_h^T - c_B^T B^{-1} A_h \geq 0, \quad \forall A_h \notin \mathcal{B}$  then
10:       ottimo  $\leftarrow$  true;
11:     else

      Pivoting (con regola di Bland):
12:        $h \leftarrow \min\{j : \tilde{c}_j < 0 \quad \forall A_j \notin \mathcal{B}\}$   $\triangleright$  entra in base
13:       if  $\hat{A}_{ij} \leq 0 \quad \forall i = 1, \dots, m$  then
14:         illimitato  $\leftarrow$  true
15:         Break  $\triangleright$  vai a 22:
16:       end if
17:        $\ell \leftarrow \arg \min \left\{ \frac{\hat{d}_i}{\hat{A}_{i,h}}, i = 1, \dots, m : \hat{A}_{i,h} > 0 \right\}$   $\triangleright$  esce dalla base
      in caso di parità  $\ell$  tale che  $\beta(\ell)$  variabile in base di indice minimo
18:        $\beta(\ell) \leftarrow \min \left\{ \beta(i) : \hat{A}_{ih} > 0, \frac{\hat{d}_i}{\hat{A}_{ih}} \leq \frac{\hat{d}_k}{\hat{A}_{kh}} \quad \forall i, k = 1, \dots, m : \hat{A}_{kh} > 0 \right\}$ 
      Aggiornamento base:
19:        $\mathcal{B} = \mathcal{B} \setminus A_{\beta(\ell)} \cup A_h$ 
20:        $\beta(\ell) = h$ 
21:       ordina il vettore  $\beta$  per valori crescenti
22:     until ottimo = true or illimitato = true
23:     return  $x$  (soluzione ottima)
24: end procedure

```

---

## 4.9 Esempi di applicazione dell'algoritmo del simplex

### 4.9.1 Esempio 1

Si consideri il problema

$$\max z = x_1 + x_2 \quad (4.51)$$

$$\text{s.t.} \quad 6x_1 + 4x_2 \leq 24 \quad (4.52)$$

$$3x_1 - 2x_2 \leq 6 \quad (4.53)$$

$$x_1, x_2 \geq 0 \quad (4.54)$$

la cui espressione in forma standard è

$$-\min -z = -x_1 - x_2 \quad (4.55)$$

$$\text{s.t.} \quad 6x_1 + 4x_2 + x_3 = 24 \quad (4.56)$$

$$3x_1 - 2x_2 + x_4 = 6 \quad (4.57)$$

$$x_1, x_2, x_3, x_4 \geq 0 \quad (4.58)$$

La regione ammissibile del problema è illustrata in figura 4.8.

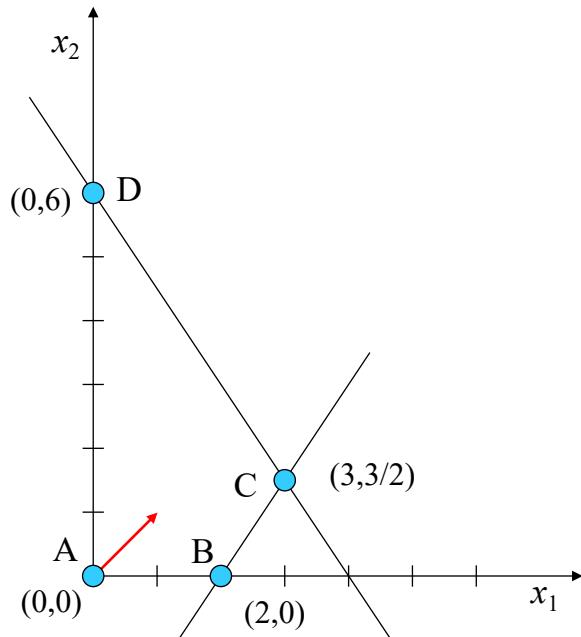


Figura 4.8: Regione Ammissibile del problema PL.

Esaminando il problema in forma standard (4.55)-(4.58) è possibile verificare facilmente che le colonne  $A_3$  e  $A_4$  sono

1. linearmente indipendenti, in quanto costituiscono una matrice identità, e pertanto possono essere una base;

2. corrispondenti ad una soluzione base ammissibile perché i termini noti sono entrambi non negativi.

Pertanto la base  $\mathcal{B} = \{A_3, A_4\}$  con  $\beta(1) = 3$  e  $\beta(2) = 4$  è una base iniziale corretta per l'algoritmo del simplex e non è necessario applicare la Fase 1 per determinarla.

### Iterazione 1

Data la base corrente la matrice di base e quella corrispondente alle variabili fuori base sono:

$$B = [A_3, A_4] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = B^{-1} \quad F = \begin{bmatrix} 6 & 4 \\ 3 & -2 \end{bmatrix}$$

Utilizzando la formula

$$x^* \leftarrow \begin{bmatrix} x_B^* \\ x_F^* \end{bmatrix} = \begin{bmatrix} B^{-1}d \\ 0 \end{bmatrix} = \begin{bmatrix} \tilde{d} \\ 0 \end{bmatrix} \quad (4.59)$$

è possibile calcolare le  $x_B = B^{-1}d = (24, 6)$  e la SBA completa  $x = (0, 0, 24, 6)$  che corrisponde al vertice  $A = (0, 0)$  in figura 4.8. La soluzione corrente ha valore  $z = 0$ .

I costi ridotti sono definiti dalla relazione

$$\tilde{c} = c_F^T - c_B^T B^{-1} F \quad (4.60)$$

che, utilizzando le matrici attuali diventa

$$\tilde{c} = [-1 \ -1] - [0 \ 0] \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 6 & 4 \\ 3 & -2 \end{bmatrix} = [-1 \ -1]$$

da cui è possibile verificare che la SBA corrente non è ottima essendovi almeno una variabile fuori base a costo ridotto negativo.

Utilizzando la regola di Bland per l'operazione di pivoting necessaria all'aggiornamento della base si vede che la variabile che entra nella base è  $x_1$ , dato che è quella di indice minimo tra le variabili fuori base a costo ridotto negativo. Per determinare la variabile che esce dalla base si applica la formula

$$\beta(\ell) = \min \left\{ \beta(i) : \hat{A}_{ih} > 0, \frac{\hat{d}_i}{\hat{A}_{ih}} \leq \frac{\hat{d}_k}{\hat{A}_{kh}} \quad \forall i, k = 1, \dots, m : \hat{A}_{kh} > 0 \right\} \quad (4.61)$$

in cui, essendo  $\hat{d} = (24, 6)$  e  $\hat{A}_1 = B^{-1}A_1 = A_1 = (6, 3)$ , si vede che il rapporto minimo tra  $(24/6, 6/3)$  corrisponde al secondo vincolo. Su tale vincolo è in base la variabile  $x_{\beta(2)} = x_4$  che esce dalla base.

### Iterazione 2

La base corrente è ora  $\mathcal{B} = \{A_1, A_3\}$  con  $\beta = \{1, 3\}$  e le corrispondenti matrici sono:

$$B = [A_1, A_3] = \begin{bmatrix} 6 & 1 \\ 3 & 3 \end{bmatrix}, \quad B^{-1} = \begin{bmatrix} 0 & 1/3 \\ 1 & -2 \end{bmatrix}, \quad F = \begin{bmatrix} 4 & 0 \\ -2 & 1 \end{bmatrix}$$

Utilizzando la formula 4.59 è possibile calcolare le  $x_B = B^{-1}d = (2, 12)$  e la SBA completa  $x = (2, 0, 12, 0)$  che corrisponde al vertice  $B = (2, 0)$  in figura 4.8. La soluzione corrente ha valore  $z = 2$ .

I costi ridotti sono definiti dalla relazione (4.60) che, utilizzando le matrici attuali diventa

$$\tilde{c} = c_F^T - c_B^T B^{-1} F = [-1 \ 0] - [-1 \ 0] \begin{bmatrix} 0 & 1/3 \\ 1 & -2 \end{bmatrix} \begin{bmatrix} 4 & 0 \\ -2 & 1 \end{bmatrix} = [-5/2 \ 1/3]$$

da cui è possibile verificare che la SBA corrente non è ottima essendovi almeno una variabile fuori base a costo ridotto negativo.

Utilizzando la regola di Bland per l'operazione di pivoting necessaria all'aggiornamento della base si vede che la variabile che entra nella base è  $x_2$ , dato che è quella di indice minimo tra le variabili fuori base a costo ridotto negativo. Per determinare la variabile che esce dalla base si applica la formula (4.61) in cui, essendo  $\hat{d} = (2, 12)$  e  $\hat{A}_2 = B^{-1}A_2 = (-2/3, 8)$ , si vede che il rapporto minimo  $12/8$  corrisponde al secondo vincolo. Su tale vincolo è in base la variabile  $x_{\beta(2)} = x_3$  che esce dalla base. Si noti che nel calcolo del rapporto minimo il primo vincolo è stato ignorato dato che il valore  $-2/3$  a questo corrispondente non è positivo.

### Iterazione 3

La base corrente è ora  $\mathcal{B} = \{A_1, A_2\}$  con  $\beta = \{1, 2\}$  e le corrispondenti matrici sono:

$$B = [A_1, A_2] = \begin{bmatrix} 6 & 4 \\ 3 & -2 \end{bmatrix}, \quad B^{-1} = \begin{bmatrix} 1/12 & 1/6 \\ 1/8 & -1/4 \end{bmatrix}, \quad F = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Utilizzando la formula 4.59 è possibile calcolare le  $x_B = B^{-1}d = (3, 3/2)$  e la SBA completa  $x = (3, 3/2, 0, 0)$  che corrisponde al vertice  $C = (3, 3/2)$  in figura 4.8. La soluzione corrente ha valore  $z = 9/2$ .

I costi ridotti sono definiti dalla relazione (4.60) che, utilizzando le matrici attuali diventa

$$\tilde{c} = c_F^T - c_B^T B^{-1} F = [0 \ 0] - [-1 \ -1] \begin{bmatrix} 1/12 & 1/6 \\ 1/8 & -1/4 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = [5/24 \ -1/12]$$

da cui è possibile verificare che la SBA corrente non è ottima essendovi almeno una variabile fuori base a costo ridotto negativo.

Utilizzando la regola di Bland per l'operazione di pivoting necessaria all'aggiornamento della base si vede che la variabile che entra nella base è  $x_4$ , dato

che è quella di indice minimo tra le variabili fuori base a costo ridotto negativo. Per determinare la variabile che esce dalla base si applica la formula (4.61) in cui, essendo  $\hat{d} = (3, 3/2)$  e  $\hat{A}_4 = B^{-1}A_4 = (1/6, -1/4)$ , si vede che il rapporto minimo  $3 * 6$  corrisponde al primo vincolo. Su tale vincolo è in base la variabile  $x_{\beta(1)} = x_1$  che esce dalla base. Si noti che nel calcolo del rapporto minimo il secondo vincolo è stato ignorato dato che il valore  $-1/4$  a questo corrispondente non è positivo.

#### Iterazione 4

La base corrente è ora  $\mathcal{B} = \{A_2, A_4\}$  con  $\beta = \{2, 4\}$  e le corrispondenti matrici sono:

$$B = [A_2, A_4] = \begin{bmatrix} 4 & 0 \\ -2 & 1 \end{bmatrix}, \quad B^{-1} = \begin{bmatrix} 1/4 & 0 \\ 1/2 & 1 \end{bmatrix}, \quad F = \begin{bmatrix} 6 & 1 \\ 3 & 0 \end{bmatrix}$$

Utilizzando la formula 4.59 è possibile calcolare le  $x_B = B^{-1}d = (6, 18)$  e la SBA completa  $x = (0, 6, 0, 18)$  che corrisponde al vertice  $D = (0, 6)$  in figura 4.8.

I costi ridotti sono definiti dalla relazione (4.60) che, utilizzando le matrici attuali diventa

$$\tilde{c} = c_F^T - c_B^T B^{-1} F = [-1 \ 0] - [-1 \ 0] \begin{bmatrix} 1/4 & 0 \\ 1/2 & 1 \end{bmatrix} \begin{bmatrix} 6 & 1 \\ 3 & 0 \end{bmatrix} = [1/4 \ 1/2]$$

da cui è possibile verificare che la SBA corrente è ottima dato che tutte le variabili fuori base hanno costo ridotto maggiore o uguale a zero. La soluzione ottima determinata ha valore  $z = 6$ .

## 4.10 La forma tableau

La forma tableau è una rappresentazione tabellare dei dati di un problema PL che rende più agevole la risoluzione dei problemi con l'algoritmo del simplex.

Il tableau viene inizializzato con i dati del problema originale come illustrato nella tabella seguente

	0	$x_1$	$x_2$	$\dots$	$x_n$
0	0			$c^T$	
1					
$\dots$	$d$			$A$	
$m$					

Nel seguito la prima riga del tableau corrispondente alla funzione obiettivo è indicata come riga 0. Analogamente la prima colonna corrispondente ai termini noti è indicata come colonna 0.

Ora consideriamo che sia data una base del sistema e per semplicità questa coincida con le prime  $m$  colonne della matrice  $A$ , ossia  $\mathcal{B} = \{A_1, \dots, A_m\}$

$$A = [\underbrace{A_1, \dots, A_m}_{\text{in base}} \mid \underbrace{A_{m+1}, \dots, A_n}_{\text{fuori base}}] = [B \mid F]$$

Il problema PL con tutti i dati può essere rappresentato come segue

$$\begin{cases} z &= c_B^T x_B + c_F^T x_F \\ d &= Bx_B + Fx_F \end{cases} \quad (4.62)$$

Il tableau quindi può essere riscritto esplicitando la base

0	$c_B^T$	$c_F^T$
$d$	$B$	$F$

Il medesimo sistema in forma canonica rispetto alla base  $\mathcal{B} = \{A_1, \dots, A_m\}$  conterrà nella prima riga il valore corrente della SBA e i costi ridotti, e nella parte rimanente i valori delle variabili base rispetto alle variabili fuori base come mostrato nei paragrafi precedenti.

$$\begin{cases} -c_B^T B^{-1}d &= 0 & +\tilde{c}_F^T x_F \\ B^{-1}d &= B^{-1}Bx_B + B^{-1}Fx_F \end{cases} \quad (4.63)$$

Riportando le medesime informazioni nel tableau si ottiene

$-c_B^T B^{-1}d$	0	$\tilde{c}_F^T$
$B^{-1}d$	$B^{-1}B = I$	$B^{-1}F$

(4.64)

Quindi un tableau che contiene i dati del problema in forma canonica rispetto ad una base contiene tutte le informazioni necessarie per eseguire le operazioni richieste dall'algoritmo del simplex. In particolare:

- Nella prima riga:
  - nella prima colonna il valore corrente della funzione obiettivo, cambiato di segno;
  - nelle colonne delle variabili base dei valori a 0;
  - nelle colonne delle variabili fuori base i corrispondenti costi ridotti.
- Nelle righe corrispondenti ai vincoli:
  - nella prima colonna il valore corrente delle variabili in base  $B^{-1}d = \hat{d}$
  - nelle colonne delle variabili base una matrice identità
  - nelle colonne delle variabili fuori base i corrispondenti coefficienti della matrice  $B^{-1}F = \hat{F}$

Per ricondurci dal tableau con i dati iniziali ad un tableau che contiene le informazioni del problema in forma canonica è possibile manipolare il tableau stesso mediante operazioni elementari di riga che è ben noto trasformano un sistema in un sistema equivalente. In particolare le operazioni elementari di riga che possiamo utilizzare sono:

- moltiplicazione di una riga per una costante;
- somma algebrica di righe eventualmente moltiplicate per una costante.

Dato il tableau contenente i dati iniziali, note le variabili della base desiderata, possiamo quindi effettuare su di esso operazioni elementari di riga per:

- azzerare i costi delle variabili base nella prima riga
- trasformare i coefficienti della matrice  $B$  in una matrice identità.

È possibile verificare che ottenendo tale trasformazione sulle colonne di base del tableau, nelle colonne rimanenti sono presenti le informazioni del sistema in forma canonica come rappresentato nel tableau (4.64).

Per esemplificare tale trasformazione consideriamo ancora il problema di produzione di sedie il cui modello espresso in forma standard è:

$$-\min -3x_1 - 5x_2 \quad (4.65)$$

$$\text{s.t.} \quad x_1 + x_3 = 4 \quad (4.66)$$

$$+x_2 + x_4 = 6 \quad (4.67)$$

$$3x_1 + 2x_2 + x_5 = 18 \quad (4.68)$$

$$x_1, x_2, x_3, x_4, x_5 \geq 0 \quad (4.69)$$

Il tableau iniziale corrispondente è quindi

	$-z$	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	
$r_0$	0	-3	-5	0	0	0	
$r_1$	4	1	0	1	0	0	
$r_2$	6	0	1	0	1	0	
$r_3$	18	3	2	0	0	1	

(4.70)

Si può osservare che tale tableau è già in forma canonica rispetto alla base  $\mathcal{B} = \{A_3, A_4, A_5\}$  ma supponiamo di volerlo mettere in forma canonica rispetto alla base  $\mathcal{B} = \{A_1, A_2, A_3\}$ . A tal fine possiamo osservare che la colonna corrispondente alla variabile  $x_3$  è già nella forma desiderata, contenendo un 0 nella prima riga e la prima colonna della matrice identità nelle righe successive. Dovremo quindi manipolare le due colonne corrispondenti ad  $x_1$  e  $x_2$  per azzerare i coefficienti nella prima riga e definire le rimanenti colonne della matrice identità.

Per azzerare il coefficiente nella prima riga della colonna  $x_1$ , che attualmente vale 3 possiamo sommare a tale riga la riga 3 corrispondente all'ultimo vincolo da cui si ottiene.

	$-z$	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	
$r_0$	18	0	-3	0	0	1	
$r_1$	4	1	0	1	0	0	
$r_2$	6	0	1	0	1	0	
$r_3$	18	3	2	0	0	1	

(4.71)

e successivamente possiamo sommare la riga 2 moltiplicata per 3 ottenendo

	$-z$	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	
$r_0$	36	0	0	0	3	1	
$r_1$	4	1	0	1	0	0	
$r_2$	6	0	1	0	1	0	
$r_3$	18	3	2	0	0	1	

(4.72)

In questo modo abbiamo completato il primo passo della trasformazione ed abbiamo portato a zero i coefficienti della riga 0 per le variabili della base desiderata. Di conseguenza nella colonna 0 troviamo il valore della SBA del

problema in forma standard cambiato di segno (ossia  $-z = 36$ ) e nelle colonne delle variabili fuori base i corrispondenti costi ridotti.

Trasformiamo ora la parte rimanente del tableau per costruire le colonne della matrice identità mancanti. Osservando che la colonna  $x_2$  contiene uno zero sulla prima riga ed un 1 sulla seconda riga, per completare la trasformazione di tale colonna è sufficiente azzerare il coefficiente della terza riga. A tal fine possiamo sottrarre dalla riga 3 la riga 2, moltiplicata per 2, ottenendo

	$-z$	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	
$r_0$	36	0	0	0	3	1	
$r_1$	4	1	0	1	0	0	
$r_2$	6	0	1	0	1	0	
$r_3$	6	3	0	0	-2	1	

(4.73)

Dobbiamo infine trasformare la colonna  $x_1$  azzerando il coefficiente nella riga 1 e portando a 1 quello della riga 3. A tal fine possiamo dapprima dividere per 3 la terza riga, ottenendo

	$-z$	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	
$r_0$	36	0	0	0	3	1	
$r_1$	4	1	0	1	0	0	
$r_2$	6	0	1	0	1	0	
$r_3$	2	1	0	0	$-\frac{2}{3}$	$\frac{1}{3}$	

(4.74)

e successivamente sottrarre dalla riga 1 la riga 3 ottenendo

	$-z$	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	
	36	0	0	0	3	1	
$x_3$	2	0	0	1	$\frac{2}{3}$	$-\frac{1}{3}$	
$x_2$	6	0	1	0	1	0	
$x_1$	2	1	0	0	$-\frac{2}{3}$	$\frac{1}{3}$	

(4.75)

Esaminando il tableau (4.75) si può verificare che in base alle posizioni degli 1 nella matrice identità si può individuare la riga in cui le variabili base sono esplicitate e quindi la riga nella quale si vede il loro valore attuale in colonna 0. In particolare la colonna in base sulla riga 1 è  $x_3$  che vale quindi 2, la colonna in

base sulla riga 2 è  $x_2$  che vale 6 e la colonna in base sulla riga 3 è  $x_1$  che vale 2. La SBA corrente è quindi  $(2, 6, 2, 0, 0)$  e vale, per il problema in forma standard,  $-36$ . Infine esaminando i costi ridotti delle variabili fuori base si può verificare che a SBA corrente è ottima dato che sono entrambi non negativi.

#### 4.10.1 Inversa della base ricavabile dal tableau NEW: (Ver 5.0)

Ricordando che un tableau in forma canonica rispetto alla base corrente è il tableau iniziale moltiplicato per l'inversa della base stessa, è facile verificare che nelle colonne della base iniziale che corrispondono ad una matrice identità possono essere lette le colonne dell'inversa della base corrente. In particolare in figura 4.9 è illustrato il tableau iniziale in cui le colonne della base iniziale sono una matrice identità e le colonne di una base differente, ad esempio la base ottima, che per comodità sono assunte completamente diverse da quelle della base iniziale. Nella medesima figura è illustrato il tableau finale nel quale è possibile reperire la matrice di base nelle colonne della base iniziale.

$0$	$c^T$		
$d$	$I$		$B$
	$\underbrace{\hspace{1cm}}$ base iniziale	$\underbrace{\hspace{1cm}}$ base corrente	

$-z_0$	$c'$		
$x_0$	$B^{-1}$		$I$
	$\underbrace{\hspace{1cm}}$ base iniziale	$\underbrace{\hspace{1cm}}$ base corrente	

Figura 4.9: Inversa della base corrente nel tableau.

Si osservi che se la base iniziale non fosse disponibile nel problema originale e fosse stato necessario applicare la Fase 1 per determinarla, al fine di ottenere l'inversa è necessario mantenere le colonne delle variabili artificiali anche nella Fase 2 ed eseguire le operazioni di pivoting anche su di esse.

### 4.10.2 Algoritmo del simplex in forma tableau

Descriviamo ora l'evoluzione dell'algoritmo del simplex adottando la rappresentazione in forma tableau. A tal fine definiamo una matrice  $Y$  di dimensione  $(m+1) \times (n+1)$  che conterrà gli elementi del tableau corrente.

#### Inizializzazione:

La matrice  $Y$  viene inizializzata come segue:

- $y_{00} = 0$ ;
- costi nella riga 0:  $y_{0j} = c_j, j = 1, \dots, n$ ;
- termini noti nella colonna 0:  $y_i = d_i, i = 1, \dots, m$ ;
- coefficienti della matrice tecnologica nel resto della matrice:  $y_{ij} = A_{ij}, i = 1, \dots, m, j = 1, \dots, n$ ;

A questo punto, data la base iniziale  $\mathcal{B} = \{A_{\beta(1)}, \dots, A_{\beta(m)}\}$ , poniamo il tableau in forma canonica rispetto a tale base come mostrato precedentemente, ossia azzerando i costi ridotti delle variabili base in riga 0 e definendo una matrice identità nelle colonne della base.

Si ricordi che dato il tableau in forma canonica il contenuto delle diverse parti del tableau è il seguente:

- $y_{00}$  contiene il valore della SBA corrente, cambiato di segno;
- nella colonna 0 si leggono i valori correnti delle variabili base. Per ogni riga la variabile in base su tale riga si determina considerando quale colonna della base corrente abbia il valore ad 1 in corrispondenza di tale riga;
- nella riga 0 in corrispondenza delle variabili fuori base si hanno i relativi costi ridotti;
- nelle rimanenti righe in corrispondenza delle variabili fuori base si hanno i relativi coefficienti della matrice  $\hat{F} = B^{-1}F$ .

#### Test di ottimalità:

Se tutti i costi ridotti delle variabili base sono non negativi, ossia se  $y_{0j} \geq 0, \forall j \notin \mathcal{B}$ , la SBA corrente è ottima e l'algoritmo termina. Altrimenti bisogna effettuare l'operazione di pivoting.

#### Pivoting:

Applicando la regola di Bland, si sceglie la colonna fuori base di indice minimo con costo ridotto negativo

$$h = \arg \min \{j \notin \beta; y_{0j} < 0\}. \quad (4.76)$$

Se  $y_{ih} \leq 0, \forall i = 1, \dots, m$  allora il problema è illimitato e l'algoritmo termina. Altrimenti, la colonna che esce dalla base si determina calcolando il minimo dei rapporti tra i valori correnti delle variabili base ed i valori correnti di  $\hat{A}_h$  ossia

$$\ell = \arg \min \{y_{0i}/y_{hi} : y_{hi} > 0, i = 1, \dots, m\}. \quad (4.77)$$

In caso di parità si sceglie  $\ell$  in modo tale che esca dalla base la variabile di indice minimo.

#### Aggiornamento del tableau:

La nuova base contiene la colonna  $A_h$  al posto della colonna  $A_{\beta(\ell)}$ . Per mettere il tableau in forma canonica rispetto alla nuova base corrente sarà sufficiente trasformare la colonna  $h$  nella appropriata colonna di matrice identità, ossia corrispondente alla colonna che esce dalla base. Le altre colonne della base rimarranno indenni dalle operazioni elementari di riga effettuate in questa fase. In particolare:

- Nella riga  $\ell$  bisogna definire  $y_{\ell h} = 1$  per cui tale riga va interamente divisa per  $y_{\ell h}$ :  $y_{\ell j} = y_{\ell j}/y_{\ell h}, \forall j = 0, \dots, n$ .
- In tutte le altre righe della colonna  $h$ , inclusa la riga 0, gli elementi devono essere annullati. Allo scopo bisogna sottrarre da tale riga la riga  $\ell$ , che in colonna  $h$  ora contiene un 1, moltiplicata per il coefficiente in colonna  $h$  su tale riga:  $y_{ij} = y_{ij} - y_{\ell j} * y_{ih}, \forall i = 0, \dots, m : i \neq \ell, j = 0, \dots, n$ .

#### 4.10.3 Esempio di risoluzione in forma tableau con solo Fase 2

Consideriamo nuovamente il problema esaminato nel paragrafo 4.9 che riportiamo nel seguito

$$\max z = x_1 + x_2 \quad (4.78)$$

$$\text{s.t.} \quad 6x_1 + 4x_2 \leq 24 \quad (4.79)$$

$$3x_1 - 2x_2 \leq 6 \quad (4.80)$$

$$x_1, x_2 \geq 0 \quad (4.81)$$

la cui espressione in forma standard è

$$-\min -z = -x_1 - x_2 \quad (4.82)$$

$$\text{s.t.} \quad 6x_1 + 4x_2 + x_3 = 24 \quad (4.83)$$

$$3x_1 - 2x_2 + x_4 = 6 \quad (4.84)$$

$$x_1, x_2, x_3, x_4 \geq 0 \quad (4.85)$$

La regione ammissibile del problema è illustrata in figura 4.10.

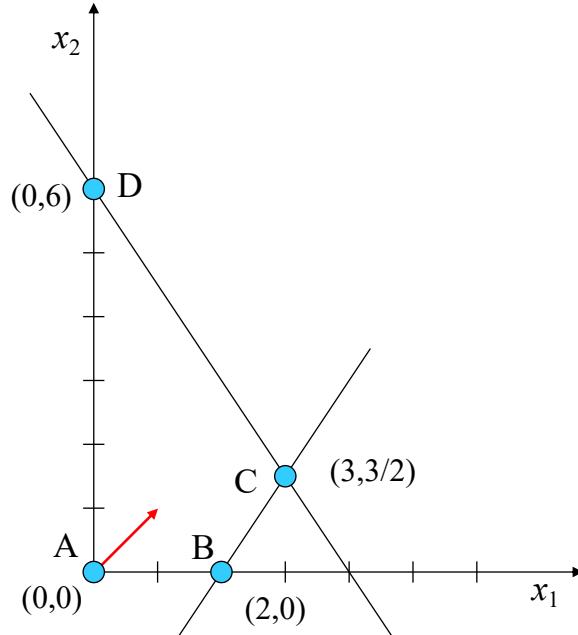


Figura 4.10: Regione Ammissibile del problema PL.

Il tableau iniziale del problema in forma standard è:

$-z$	$x_1$	$x_2$	$x_3$	$x_4$	
0	-1	-1	0	0	
$x_3$	24	6	4	1	0
$x_4$	6	3	-2	0	1

(4.86)

Anche in questo caso iniziamo dalla base  $\mathcal{B} = \{A_3, A_4\}$  ed osserviamo che il tableau (4.86) è già in forma canonica rispetto a tale base che non è ottima dato che i costi ridotti di entrambe le variabili fuori base sono negativi.

#### Iterazione 1:

Seguendo la regola di Bland scegliamo per il pivoting la colonna 1 e, di conseguenza la riga 2, che corrisponde al minimo dei rapporti  $\frac{24}{6}$  e  $\frac{6}{3}$ . Esce quindi dalla base la colonna  $A_4$  che viene sostituita dalla colonna  $A_1$ . Il nuovo tableau, in forma canonica rispetto alla nuova base si ottiene con le seguenti operazioni elementari di riga:

- dividendo la riga 2 per 3;

- sommando alla riga 0 la nuova riga 2;
- sottraendo dalla riga 0 la nuova riga 2, moltiplicata per 6.

Il nuovo tableau è

	$-z$	$x_1$	$x_2$	$x_3$	$x_4$	
	2	0	$-\frac{5}{3}$	0	$\frac{1}{3}$	
$x_3$	12	0	8	1	-2	
$x_1$	2	1	$-\frac{2}{3}$	0	$\frac{1}{3}$	

(4.87)

La SBA corrente è  $(2, 0, 12, 0)$  e vale  $-2$  ma non è ottima in quanto la variabile fuori base  $x_2$  ha costo ridotto negativo. Sarà quindi necessaria una ulteriore operazione di pivoting.

### Iterazione 2:

Seguendo la regola di Bland scegliamo per il pivoting la colonna 2 e, di conseguenza la riga 1, che corrisponde al minimo dei rapporti considerando gli elementi positivi in tale colonna. Esce quindi dalla base la colonna  $A_3$  che viene sostituita dalla colonna  $A_2$ . Il nuovo tableau, in forma canonica rispetto alla nuova base si ottiene con le seguenti operazioni elementari di riga:

- dividendo la riga 1 per 8;
- sommando alla riga 0 la nuova riga 1 moltiplicata per  $\frac{5}{3}$ ;
- sommando alla riga 2 la nuova riga 1, moltiplicata per  $\frac{2}{3}$ .

Il nuovo tableau è

	$-z$	$x_1$	$x_2$	$x_3$	$x_4$	
	$\frac{9}{2}$	0	0	$\frac{5}{24}$	$-\frac{1}{12}$	
$x_2$	$\frac{3}{2}$	0	1	$\frac{1}{8}$	$-\frac{1}{4}$	
$x_1$	3	1	0	$\frac{1}{12}$	$\frac{1}{6}$	

(4.88)

La SBA corrente è  $(3, \frac{3}{2}, 0, 0)$  e vale  $-\frac{9}{2}$  ma non è ottima in quanto la variabile fuori base  $x_4$  ha costo ridotto negativo. Sarà quindi necessaria una ulteriore operazione di pivoting.

**Iterazione 3:**

Seguendo la regola di Bland scegliamo per il pivoting la colonna 4 e, di conseguenza la riga 2, che corrisponde al minimo dei rapporti considerando gli elementi positivi in tale colonna. Esce quindi dalla base la colonna  $A_1$  che viene sostituita dalla colonna  $A_4$ . Il nuovo tableau, in forma canonica rispetto alla nuova base si ottiene con le seguenti operazioni elementari di riga:

- dividendo la riga 2 per  $\frac{1}{6}$ , ossia moltiplicandola per 6;
- sommando alla riga 0 la nuova riga 2, moltiplicata per  $\frac{1}{12}$ ;
- sommando alla riga 1 la nuova riga 2, moltiplicata per  $\frac{1}{4}$ .

Il nuovo tableau è

$-z$	$x_1$	$x_2$	$x_3$	$x_4$	
6	$\frac{1}{2}$	0	$\frac{1}{4}$	0	
$x_2$	6	$\frac{3}{2}$	1	$\frac{1}{4}$	0
$x_4$	18	6	0	$\frac{1}{2}$	1

(4.89)

La SBA corrente è  $(0, 6, 0, 18)$  e vale  $-6$  ed è ottima in quanto entrambe le variabili fuori base ha costo ridotto maggiore o uguale a zero. Si noti che il valore della soluzione ottima per il problema originale che era di massimizzazione è uguale a 6.

**4.10.4 Esempio di risoluzione in forma tableau con Fase 1**

Consideriamo il seguente problema PL

$$\min z = x_1 + x_3 \quad (4.90)$$

$$\text{s.t.} \quad x_1 + 2x_2 \leq 5 \quad (4.91)$$

$$x_2 + 2x_3 = 6 \quad (4.92)$$

$$x_1, x_2, x_3 \geq 0 \quad (4.93)$$

la cui espressione in forma standard è

$$-\min -z = -x_1 + x_3 \quad (4.94)$$

$$\text{s.t.} \quad x_1 + 2x_2 + x_4 = 5 \quad (4.95)$$

$$x_2 + 2x_3 = 6 \quad (4.96)$$

$$x_1, x_2, x_3, x_4 \geq 0 \quad (4.97)$$

**Fase 1:**

Il tableau iniziale della Fase 1 del problema in forma standard si ottiene considerando due variabili artificiali  $x_1^a$  e  $x_2^a$  e la relativa funzione obiettivo di Fase 1 ed è:

$-z$	$x_1$	$x_2$	$x_3$	$x_4$	$x_1^a$	$x_2^a$	
0	0	0	0	0	1	1	
$x_1^a$	5	1	2	0	1	0	
$x_2^a$	6	0	1	2	0	0	1

(4.98)

Per procedere con la Fase 1 è necessario porre il tableau in forma canonica rispetto alla base costituita dalle sole variabili artificiali  $\mathcal{B} = \{x_1^a, x_2^a\}$ . A tal fine dobbiamo azzerare i coefficienti di tali variabili nella riga 0 e questo può essere facilmente ottenuto sottraendo da tale riga le righe 1 e 2 del tableau ottenendo:

$-z$	$x_1$	$x_2$	$x_3$	$x_4$	$x_1^a$	$x_2^a$	
-11	-1	-3	-2	-1	0	0	
$x_1^a$	5	1	2	0	1	0	
$x_2^a$	6	0	1	2	0	0	1

(4.99)
**Iterazione 1:**

Seguendo la regola di Bland scegliamo per il pivoting la colonna 1 e, di conseguenza la riga 1, che corrisponde al minimo dei rapporti considerando i soli elementi positivi della colonna. Esce quindi dalla base la colonna  $x_1^a$  che viene sostituita dalla colonna  $A_1$ . Il nuovo tableau, in forma canonica rispetto alla nuova base si ottiene sommando alla riga 0 la riga 1 per azzerare il costo ridotto. Il resto della colonna è già nel formato desiderato ed il nuovo tableau è

$-z$	$x_1$	$x_2$	$x_3$	$x_4$	$x_1^a$	$x_2^a$	
-6	0	-1	-2	0	1	0	
$x_1$	5	1	2	0	1	0	
$x_2^a$	6	0	1	2	0	0	1

(4.100)

La SBA corrente è  $(5, 0, 0, 0|0, 6)$  e vale 6 ma non è ottima in quanto esistono variabili originali fuori base a costo ridotto negativo. Sarà quindi necessaria una ulteriore operazione di pivoting.

**Iterazione 2:**

Seguendo la regola di Bland scegliamo per il pivoting la colonna  $A_2$  e, di conseguenza la riga 1, che corrisponde al minimo dei rapporti,  $\frac{5}{2}$  e  $\frac{6}{1}$ , considerando gli elementi positivi in tale colonna. Esce quindi dalla base la colonna  $A_1$  che viene sostituita dalla colonna  $A_2$ . Il nuovo tableau, in forma canonica rispetto alla nuova base si ottiene con le seguenti operazioni elementari di riga:

- dividendo la riga 1 per 2;
- sommando alla riga 0 la nuova riga 1;
- sottraendo dalla riga 2 la nuova riga 1.

Il nuovo tableau è

$-z$	$x_1$	$x_2$	$x_3$	$x_4$	$x_1^a$	$x_2^a$	
$-\frac{7}{2}$	$\frac{1}{2}$	0	-2	$\frac{1}{2}$	$\frac{3}{2}$	0	
$x_2$	$\frac{5}{2}$	$\frac{1}{2}$	1	0	$\frac{1}{2}$	$\frac{1}{2}$	0
$x_2^a$	$\frac{7}{2}$	$-\frac{1}{2}$	0	2	$-\frac{1}{2}$	$-\frac{1}{2}$	1

(4.101)

La SBA corrente è  $(0, \frac{5}{2}, 0, 0 | 0, \frac{7}{2})$  e vale  $\frac{7}{2}$  ma non è ottima in quanto la variabile fuori base  $x_3$  ha costo ridotto negativo. Sarà quindi necessaria una ulteriore operazione di pivoting.

**Iterazione 3:**

Seguendo la regola di Bland scegliamo per il pivoting la colonna  $A_3$  e, di conseguenza la riga 2, che corrisponde al minimo dei rapporti considerando gli elementi positivi in tale colonna. Esce quindi dalla base la colonna  $x_2^a$  che viene sostituita dalla colonna  $A_3$ . Il nuovo tableau, in forma canonica rispetto alla nuova base si ottiene con le seguenti operazioni elementari di riga:

- dividendo la riga 2 per 2;
- sommando alla riga 0 la nuova riga 2, moltiplicata per 2.

Il nuovo tableau è

$-z$	$x_1$	$x_2$	$x_3$	$x_4$	$x_1^a$	$x_2^a$	
0	0	0	0	0	1	1	
$x_2$	$\frac{5}{2}$	$\frac{1}{2}$	1	0	$\frac{1}{2}$	$\frac{1}{2}$	0
$x_3$	$\frac{7}{4}$	$-\frac{1}{4}$	0	1	$-\frac{1}{4}$	$-\frac{1}{4}$	$\frac{1}{2}$

(4.102)

La SBA corrente è  $(0, \frac{5}{2}, \frac{7}{4}, 0 | 0, 0)$  e vale 0 ed è ottima in quanto tutte le variabili fuori base hanno costo ridotto maggiore o uguale a zero. Si noti inoltre che tutte le variabili artificiali sono uscite dalla base che è ora costituita da sole variabili originali. Si può pertanto utilizzare la SBA corrente per inizializzare la Fase 2 dell'algoritmo.

Si noti che nel tableau iniziale di Fase 1 (4.98) la colonna  $A_4$  era già una colonna di matrice identità e quindi adatta per costituire la base iniziale. In questo caso sarebbe quindi stato sufficiente aggiungere la sola variabile artificiale  $x_2^a$  per ottenere una base completa ed eseguire la Fase 1.

### Fase 2:

Per inizializzare la Fase 2 si rimuovono dal tableau (4.102) le variabili artificiali e si ripristina la funzione obiettivo originale del problema:

	$-z$	$x_1$	$x_2$	$x_3$	$x_4$	
	0	1	0	1	0	
$x_2$	$\frac{5}{2}$	$\frac{1}{2}$	1	0	$\frac{1}{2}$	
$x_3$	$\frac{7}{4}$	$-\frac{1}{4}$	0	1	$-\frac{1}{4}$	

(4.103)

Si noti che il tableau non è completamente in forma canonica rispetto alla base  $B = \{A_2, A_3\}$  dato che il costo ridotto della colonna  $A_3$  non è nullo. A tal fine possiamo sottrarre dalla riga 0 la riga 2 ottenendo il tableau:

	$-z$	$x_1$	$x_2$	$x_3$	$x_4$	
	$-\frac{7}{4}$	$\frac{1}{4}$	0	0	$\frac{1}{4}$	
$x_2$	$\frac{5}{2}$	$\frac{1}{2}$	1	0	$\frac{1}{2}$	
$x_3$	$\frac{7}{4}$	$-\frac{1}{4}$	0	1	$-\frac{1}{4}$	

(4.104)

Osservando i costi ridotti si può notare che sono tutti non negativi e quindi la base corrente è ottima. La SBA corrente è  $(0, \frac{5}{2}, \frac{7}{4}, 0)$  e vale  $\frac{7}{4}$ .

## Capitolo 5

# Analisi di sensitività per PL

In un problema PL i coefficienti che lo definiscono sono assunti come valori deterministici noti con precisione. Nella realtà purtroppo i coefficienti che definiscono un problema PL, siano essi i costi delle variabili nella funzione obiettivo o i termini noti dei vincoli sono spesso il risultato di estrazioni di dati da archivi o di misure o stime e quindi sono naturalmente affetti da errori ed imprecisioni.

Da quanto discusso nei capitoli precedenti è evidente che la soluzione ottima di un problema PL è ottenuta mediante l'elaborazione algoritmica dei coefficienti del problema e da questi strettamente dipende. Appare quindi lecito chiedersi quanto la soluzione ottima determinata sia sensibile ad eventuali errori nella valorizzazione dei coefficienti. Tale analisi, detta *analisi di sensitività* è comune nei procedimenti ingegneristici e nel caso della PL può essere condotta in modo agevole permettendo quindi di avere informazioni utili alla valutazione della affidabilità delle soluzioni determinate. In particolare per analisi di sensitività di un problema PL si intende l'analisi della *stabilità della base ottima* (ossia dell'insieme di decisioni attive nella soluzione ottima) determinata con l'algoritmo del simplex rispetto alla variazione dei parametri del problema. Più precisamente, per ogni coefficiente del problema è possibile calcolare l'intervallo di variazione del suo valore per cui la base ottima determinata dall'algoritmo del simplex rimane tale. Tali intervalli possono essere agevolmente calcolati per la variazione di

- coefficienti della funzione obiettivo
- termini noti dei vincoli

ma possono essere condotti anche per specifici elementi della matrice tecnologica  $A$  del problema. Si noti che gli intervalli determinati sono relativi alla variazione di un singolo coefficiente alla volta del problema, ossia gli altri coefficienti sono fissati ad un valore costante.

Nel caso della PL l'analisi di sensitività è particolarmente agevole dato che le informazioni elaborate nell'algoritmo del simplex permettono di determinare

in modo automatico gli intervalli di variazione dei coefficienti della funzione obiettivo e dei termini noti.

Al fine di illustrare in via intuitiva in cosa consiste l'analisi di sensitività per un problema PL consideriamo il problema dell'esercizio 6 relativo alla produzione di due fertilizzanti. Il modello matematico del problema è

$$\begin{aligned} \max z = & 200 x_1 + 300 x_2 \\ x_1 + 2 x_2 & \leq 80 \\ 3 x_1 + x_2 & \leq 90 \\ x_1, x_2 & \geq 0 \end{aligned}$$

in cui i vincoli sono stati moltiplicati per 10. La soluzione grafica del problema è riportata in figura 5.1 da cui si può verificare che la soluzione ottima del problema è associata al vertice C che corrisponde ad  $x_1 = 30$  e  $x_2 = 40$  ed ha un valore pari 13000 €.

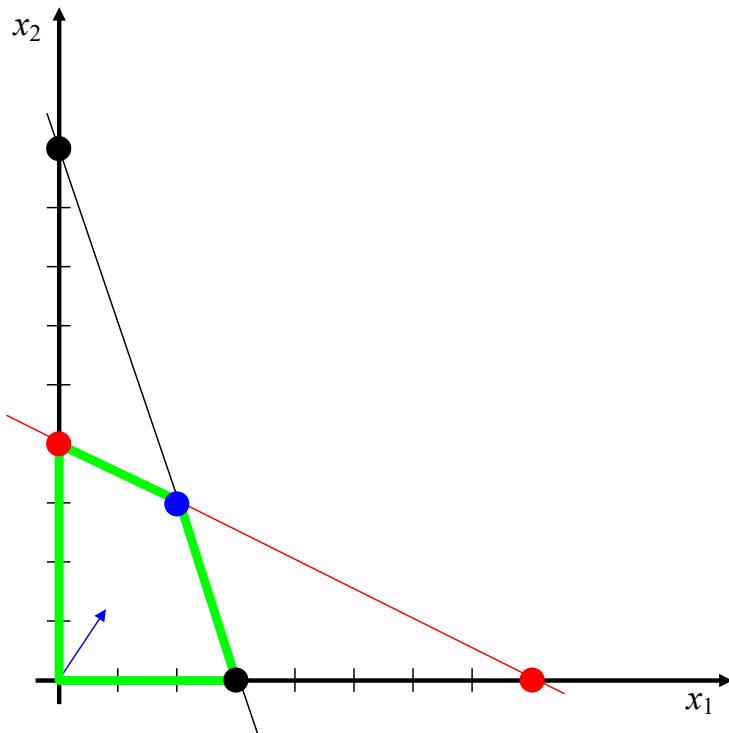


Figura 5.1: Regione ammissibile del problema di produzione fertilizzanti

Esaminiamo dapprima il caso della variazione di un coefficiente della funzione obiettivo. Ad esempio consideriamo il coefficiente  $c_1$  della variabile  $x_1$  che nel problema originario vale 200. È noto che i coefficienti della funzione obiettivo determinano la direzione del gradiente e quindi stabiliscono quale dei vertici della regione ammissibile corrisponda alla soluzione ottima. Nel nostro caso, variando  $c_1$  e mantenendo costanti gli altri coefficienti della funzione obiettivo il gradiente cambierà la propria inclinazione come mostrato in figura 5.2. In particolare, aumentando  $c_1$  rispetto a 200 la direzione del gradiente diventerà più orizzontale e diminuendone il valore questo diventerà più verticale. La soluzione ottima corrente (e di conseguenza la base ottima corrente) rimarrà però tale fintanto che la proiezione del gradiente sul CD della regione ammissibile ha una componente positiva verso il vertice ottimo corrente (vertice C). Tale situazione rimane verificata finché l'aumento di  $c_1$  non sarà tale da rendere il gradiente perpendicolare al vincolo CD. In corrispondenza di tale valore il vertice corrente è ancora ottimo come lo sono tutti i punti del lato CD incluso il vertice D. Con semplici considerazioni grafiche è possibile verificare che il valore di  $c_1$  che rende il gradiente perpendicolare a CD è pari a 900 unità. Aumentando ulteriormente  $c_1$  la soluzione ottima del problema cambia e coincide con il vertice D. Il massimo aumento di  $c_1$  rispetto al valore attuale compatibile con l'ottimalità della base corrente è quindi pari a 700. Con considerazioni analoghe si può determinare la massima diminuzione di  $c_1$  che corrisponde al valore che rende il gradiente perpendicolare al vincolo BC ed è pari a 50 unità.

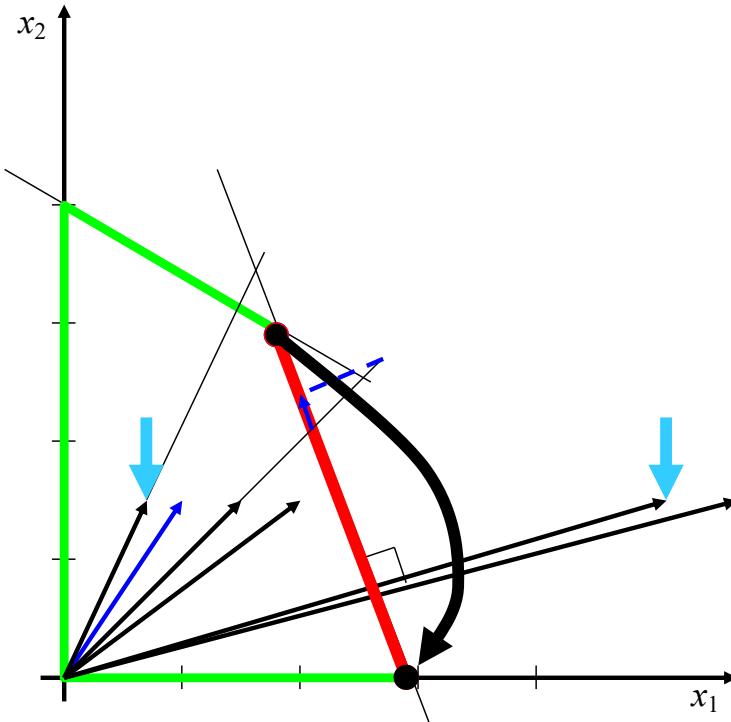


Figura 5.2: Effetto della variazione del coefficiente  $c_1$  sulla soluzione ottima.

Si noti che variando un coefficiente di costo la soluzione ottima non cambia, rimanendo uguale ad  $x_1 = 30, x_2 = 40$ , ma varia il suo valore  $z$ .

Consideriamo ora la variazione di un termine noto, ad esempio  $b_1$  che attualmente vale 80. È noto che la variazione di un termine noto altera la regione ammissibile traslando il corrispondente vincolo parallelamente a se stesso come illustrato nella figura 5.3. Poiché il gradiente rimane immutato aumentando  $b_1$  la soluzione rimane corrispondente al vertice C (e quindi la base ottima rimane la stessa) fintanto che il valore di  $b_1$  non aumenta al punto di far coincidere il vertice C con il vertice B. Da tale valore in poi la base ottima cambia e con considerazioni analoghe si può procedere nel caso della diminuzione di  $b_1$ .

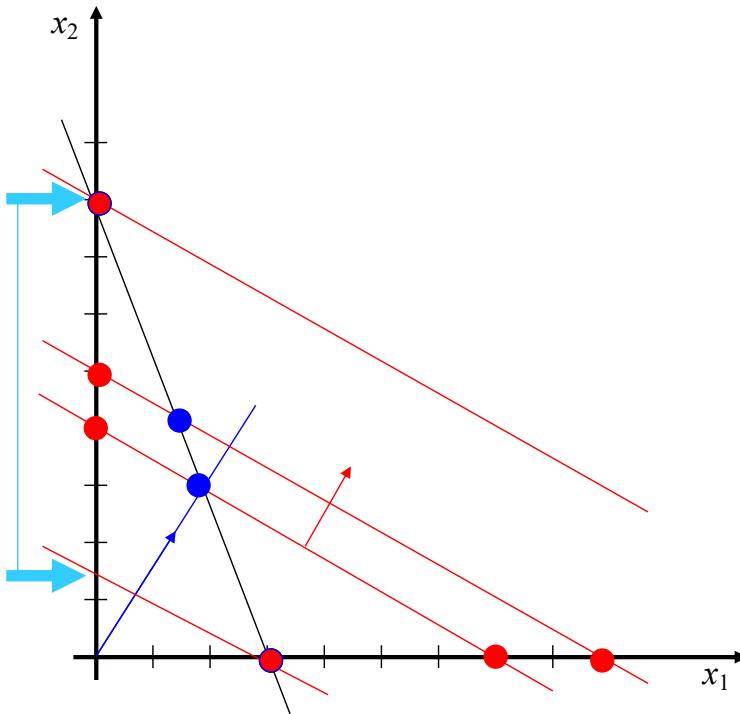


Figura 5.3: Effetto della variazione del termine noto  $b_1$  sulla soluzione ottima.

Si noti che variando un termine noto la regione ammissibile si deforma e pertanto la soluzione ottima cambia di valore. Ai nostri fini però è importante che la base ottima rimanga la stessa ossia preveda di produrre entrambi i fertilizzanti.

## 5.1 Definizione Analitica degli intervalli NEW: (Ver. 4.0)

Al fine di determinare in maniera analitica gli estremi degli intervalli di variazione dei coefficienti ricordiamo che dato un problema di PL  $P = \min\{c^T x, x \in \Re^n, Ax = d\}$  e data una base  $\mathcal{B} = \{A_{\beta(1)}, \dots, A_{\beta(m)}\}$  le variabili base  $x_B$  in funzione delle variabili fuori base  $x_F$  sono date dalla formula

$$x_B = B^{-1}d - B^{-1}F x_F. \quad (5.1)$$

La soluzione base  $x_B = B^{-1}d$  e  $x_F = 0$  è ammissibile se  $x_B \geq 0$  ed è ottima se  $\tilde{c} = c_F^T - c_B^T B^{-1} F \geq 0$ .

### 5.1.1 Variazione di un termine noto

Si noti che la variazione dei termini noti ha effetto solo sul valore delle variabili base e quindi la loro variazione può unicamente avere effetto sull'ammissibilità della base stessa.

Consideriamo il termine noto della  $k$ -sima equazione del modello e supponiamo di alterarne il valore  $d_k \rightarrow (d_k + \varepsilon)$ . La soluzione base diviene dunque

$$x_B = B^{-1}(d + I_k \varepsilon), \quad (5.2)$$

dove  $I_k$  è la  $k$ -sima colonna di una matrice identità, con il coefficiente ad 1 in corrispondenza della riga  $k$ . Se indichiamo con  $b_{ij}$  gli elementi dell'inversa della matrice di base  $B^{-1}$ , dalla (5.2) la generica variabile base è espressa da:

$$x_{\beta(i)} = \sum_{j=1}^m b_{ij} d_j + \varepsilon b_{ik}. \quad (5.3)$$

Affinché, per effetto della variazione del valore del termine noto  $d_k$ , la base rimanga ammissibile le variabili base espresse dalla (5.3) devono rimanere non negative per  $i = 1, \dots, m$ . Si osservi che il primo termine della (5.3) è il valore attuale della variabile base  $x_{\beta(i)}$  ed è quindi non negativo per definizione. Pertanto la base rimane ammissibile se anche il termine  $\varepsilon_k b_{ik}$  è non negativo. In dipendenza del valore di  $b_{ik}$  possono verificarsi tre casi:

- a)  $b_{ik} = 0$ : in tal caso anche  $\varepsilon_k b_{ik} = 0$  e la condizione non influenza il valore di  $\varepsilon_k$ .
- b)  $b_{ik} > 0$ : in tal caso, per l'ammissibilità della base  $\varepsilon_k$  deve essere tale che

$$\varepsilon_k \geq \frac{-\sum_{j=1}^m b_{ij} d_j}{b_{ik}} \quad \forall i : b_{ik} > 0. \quad (5.4)$$

- c)  $b_{ik} < 0$ : in tal caso, per l'ammissibilità si della base  $\varepsilon_k$  deve essere tale che

$$\varepsilon_k \leq \frac{-\sum_{j=1}^m b_{ij} d_j}{b_{ik}} \quad \forall i : b_{ik} < 0. \quad (5.5)$$

In base a quanto discusso affinché la base corrente rimanga ottima la variazione del termine noto  $d_k$  deve essere compresa nell'intervallo

$$\varepsilon_k^- = \max_{i:b_{ik}>0} \left\{ \frac{-\sum_{j=1}^m b_{ij} d_j}{b_{ik}} \right\} \leq \varepsilon_k \leq \varepsilon_k^+ = \min_{i:b_{ik}<0} \left\{ \frac{-\sum_{j=1}^m b_{ij} d_j}{b_{ik}} \right\}. \quad (5.6)$$

ossia  $d_k + \varepsilon_k^- \leq d_k \leq d_k + \varepsilon_k^+$ .

### 5.1.2 Variazione del costo di una variabile fuori base

Variando il costo delle variabili si agisce sui costi ridotti di tali variabili. Supponiamo ora di variare il costo di una variabile fuori base  $c_h \rightarrow c_h + \varepsilon_h$ . Si noti che tale variazione di costo ha effetto unicamente sul costo ridotto di tale variabile.

Il costo ridotto originale della variabile  $x_h$  è

$$\tilde{c}_h = c_h - [c_B B^{-1} F]_h$$

dove l'ultimo termine è l' $h$ -sima colonna della matrice  $c_B B^{-1} F$ . Per effetto della variazione del costo il nuovo costo ridotto diviene

$$c_h + \varepsilon_h - [c_B B^{-1} F]_h = \tilde{c}_h + \varepsilon_h.$$

Pertanto la base rimane ottima se

$$\varepsilon_h \geq -\tilde{c}_h.$$

### 5.1.3 Variazione del costo di una variabile base

Supponiamo ora di variare il costo di una variabile base  $c_h \rightarrow c_h + \varepsilon_h$ . Si noti che tale variazione di costo ha effetto sul costo ridotto di tutte le variabili fuori base.

Il costo ridotto originale della generica variabile fuori base  $x_j$  è

$$\tilde{c}_j = c_j - ([c_B B^{-1} F]_j + \varepsilon_h b_h A_j)$$

dove  $b_h$  è l' $h$ -sima colonna della matrice di base. Osservando che i primi due termini di tale espressione costituiscono il costo ridotto attuale della variabile, che per la base ottima è  $\geq 0$ , si ha che il nuovo costo ridotto rimane non negativo se

$$\tilde{c}_j - \varepsilon_h b_h A_j \geq 0$$

In dipendenza del valore di  $\tilde{a}_{hj} = b_h A_j$  possono verificarsi tre casi:

- a)  $\tilde{a}_{hj} = 0$ : in tal caso il costo ridotto non cambia e rimane non negativo.
- b)  $\tilde{a}_{hj} > 0$ : in tal caso,  $\varepsilon_h$  deve essere tale che

$$\varepsilon_h \leq \frac{\tilde{c}_j}{\tilde{a}_{hj}} \quad \forall j : \tilde{a}_{hj} > 0. \quad (5.7)$$

- c)  $\tilde{a}_{hj} < 0$ : in tal caso,  $\varepsilon_h$  deve essere tale che

$$\varepsilon_h \geq \frac{\tilde{c}_j}{\tilde{a}_{hj}} \quad \forall j : \tilde{a}_{hj} < 0. \quad (5.8)$$

In base a quanto discusso affinché i costi ridotti delle variabili fuori base rimangano non negativi e quindi la base corrente rimanga ottima, la variazione del costo  $c_h$  deve essere compresa nell'intervallo

$$\varepsilon_h^- = \max_{j: \tilde{a}_{hj} < 0} \left\{ \frac{\tilde{c}_j}{\tilde{a}_{hj}} \right\} \leq \varepsilon_h \leq \varepsilon_h^+ = \min_{j: \tilde{a}_{hj} > 0} \left\{ \frac{\tilde{c}_j}{\tilde{a}_{hj}} \right\}. \quad (5.9)$$

ossia  $c_h + \varepsilon_h^- \leq c_h \leq c_h + \varepsilon_h^+$ .

### 5.1.4 Analisi di sensitività dal tableau ottimo

Le informazioni per il calcolo degli intervalli dell'analisi di sensitività sono tutte facilmente reperibili dal tableau ottimo. A tal proposito si consideri il tableau ottimo dell'esercizio 6 e verifichiamo come calcolare gli intervalli relativi ai termini noti. Come illustrato nella figura 5.4 nelle colonne della base iniziale sono reperibili le colonne dell'inversa della base ottima  $B^{-1}$  mentre il numeratore delle frazioni che è il valore ottimo delle variabili base è reperibile nella colonna zero. Gli intervalli di variazione, definiti dalla (5.6), sono quindi

$d_1: \varepsilon_1 \geq -30/(3/5) = -50$  e  $\varepsilon_1 \leq -20/(-1/5) = 100$ , da cui ricordando che  $d_1 = 80$  si ottiene  $30 \leq d_1 \leq 180$ .

$d_2: \varepsilon_2 \geq -20/(2/5) = -50$  e  $\varepsilon_2 \leq -30/(-1/5) = 150$ , da cui ricordando che  $d_2 = 90$  si ottiene  $40 \leq d_2 \leq 240$ .

		$x_1$	$x_2$	$x_3$	$x_4$
$z$	13000	0	0	140	20
$x_2$	30	0	1	$3/5$	$-1/5$
$x_1$	20	1	0	$-1/5$	$2/5$

$$\varepsilon_k \geq \max_{b_{ik} > 0} \frac{-\sum_{j=1,m} b_{ij}d_j}{b_{ik}}$$

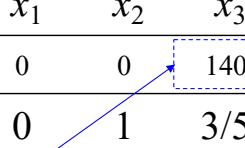
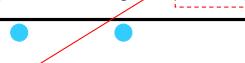
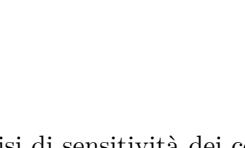
Figura 5.4: Informazioni per l'analisi di sensitività dei termini noti dal tableau

In maniera del tutto analoga si possono reperire le informazioni nel tableau ottimo anche le informazioni per l'analisi di sensitività rispetto ai costi. In particolare come illustrato in figura 5.5 i costi ridotti sono reperibili nella riga 0 mentre i coefficienti  $\tilde{a}$  si leggono direttamente dal tableau nelle righe associate ai vincoli. Gli intervalli di variazione per i costi delle variabili  $x_1$  e  $x_2$ , definiti dalla (5.9), sono quindi

$c_1$ : la variabile  $x_1$  è in base sulla riga 2, pertanto  $\varepsilon_1 \geq 140/(-1/5) = -700$  e  $\varepsilon_1 \leq 20/(2/5) = 50$ , da cui ricordando che  $c_1 = -200$  si ottiene  $-900 \leq c_1 \leq -150$ .

$c_2$ : la variabile  $x_2$  è in base sulla riga 1, pertanto  $\varepsilon_2 \geq 20/(-1/5) = -100$  e  $\varepsilon_2 \leq 140/(3/5) = 233.33\dots$ , da cui ricordando che  $c_2 = -300$  si ottiene  $-400 \leq c_2 \leq -66.66\dots$ .

		$x_1$	$x_2$	$x_3$	$x_4$
$z$	13000	0	0	140	20
$x_2$	30	0	1	$3/5$	$-1/5$
$x_1$	20	1	0	$-1/5$	$2/5$

$c'_j$        $a'_{hj}$

$$\varepsilon_h \geq \max_{a'_{hj} < 0} \frac{c'_j}{a'_{hj}}$$

Figura 5.5: Informazioni per l'analisi di sensitività dei costi dal tableau



## Capitolo 6

# Esercizi di Programmazione Lineare

### Esercizio 6.1. Produzione di fertilizzanti.

Un consorzio agrario può produrre due tipi di fertilizzanti. Ogni quintale di fertilizzante di tipo *A* contiene 0.1 quintali di azoto e 0.3 quintali potassio ed ha un prezzo di vendita di 200€. Ogni quintale di fertilizzante di tipo *B* contiene 0.2 quintali di azoto e 0.1 quintali di potassio e viene venduto a 300€. Il consorzio dispone di 8 quintali di azoto e di 9 quintali di potassio.

- a) Definire il modello di programmazione lineare per determinare le quantità da produrre di fertilizzante di ciascun tipo per massimizzare il ricavo conseguibile.
- b) Determinare la soluzione di massimo ricavo mediante l'algoritmo del simplesso e la regola di Bland.
- c) Disegnare la regione ammissibile.

### Soluzione

- a) Vengono utilizzate due variabili  $x_1$  ed  $x_2$ , che costituiscono il numero di quintali da produrre di fertilizzante di tipo *A* e *B*, rispettivamente. Il modello di programmazione lineare è pertanto

$$\begin{aligned} \max z = & 200 x_1 + 300 x_2 \\ & 0.1 x_1 + 0.2 x_2 \leq 8 \\ & 0.3 x_1 + 0.1 x_2 \leq 9 \\ & x_1, x_2 \geq 0 \end{aligned}$$

in cui il ricavo totale  $z$  è espresso in €. Moltiplicando i vincoli per 10 al fine di ottenere coefficienti interi e ponendo il modello in forma standard, si ottiene

$$\begin{aligned} \min -z = & -200x_1 - 300x_2 \\ & + 2x_2 + x_3 = 80 \\ 3x_1 + & x_2 + x_4 = 90 \\ x_1, & x_2, x_3, x_4 \geq 0 \end{aligned}$$

**b)** Poichè le ultime due colonne del tableau costituiscono una base iniziale ammissibile, non è necessario applicare la Fase 1.

		$x_1$	$x_2$	$x_3$	$x_4$
$z$	0	-200	-300	0	0
$x_3$	80	1	2	1	0
$x_4$	90	(3)	1	0	1

		$x_1$	$x_2$	$x_3$	$x_4$
$z$	6000	0	$-\frac{700}{3}$	0	$\frac{200}{3}$
$x_3$	50	0	( $\frac{5}{3}$ )	1	$-\frac{1}{3}$
$x_1$	30	1	$\frac{1}{3}$	0	$\frac{1}{3}$

		$x_1$	$x_2$	$x_3$	$x_4$
$z$	13000	0	0	140	20
$x_2$	30	0	1	$\frac{3}{5}$	$-\frac{1}{5}$
$x_1$	20	1	0	$-\frac{1}{5}$	$\frac{2}{5}$

La soluzione ottima consiste dunque nel produrre 20 quintali di fertilizzante di tipo A e 30 quintali di fertilizzante di tipo B; si ottiene in tal modo un ricavo complessivo di 13000€.

**c)** La regione ammissibile del problema e la direzione del gradiente della funzione obiettivo sono mostrati in figura 6.1.

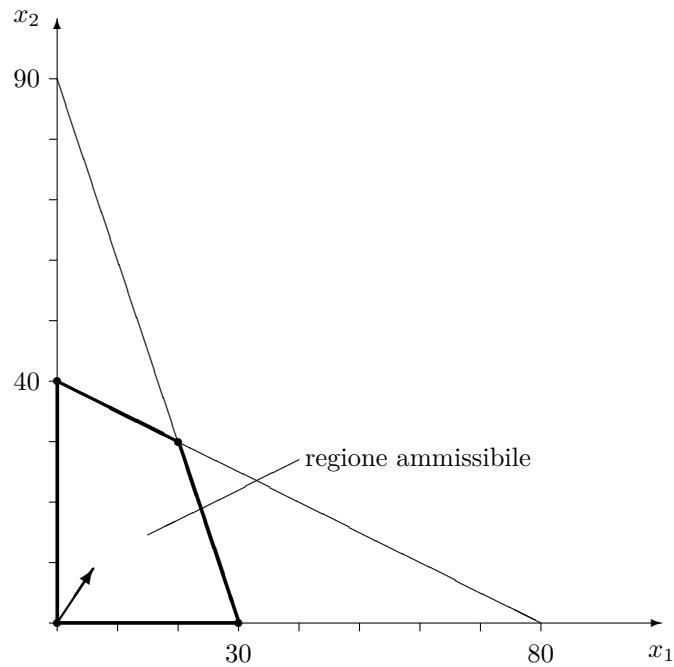


Figura 6.1: Regione ammissibile

### Esercizio 6.2. Miscelazione di mangimi.

La dieta alimentare degli animali di un allevamento richiede la presenza di 4 sostanze base:  $A$ ,  $B$ ,  $C$  e  $D$ . La quantità minima giornaliera di cui ogni animale necessita è: 0.4 Kg di  $A$ , 0.6 Kg di  $B$ , 2 Kg di  $C$  ed 1.3 Kg di  $D$ . Il cibo è ottenuto mescolando due tipi di mangime:  $M$  ed  $N$ . Ciascun Kg di  $M$  contiene 100 grammi di  $A$ , 100 grammi di  $C$ , 100 grammi di  $D$  e nulla di  $B$ ; ciascun Kg di  $N$  contiene 100 grammi di  $B$ , 100 grammi di  $D$ , 200 grammi di  $C$  e nulla di  $A$ . Con una spesa di 30€ è possibile acquistare 15 Kg di mangime  $M$  oppure 10 Kg di mangime  $N$ .

- Definire il modello di programmazione lineare per determinare le quantità giornaliere di mangime  $M$  ed  $N$  (variabili  $x_1$  ed  $x_2$ , rispettivamente) da acquistare per alimentare un animale a costo minimo.
- Disegnare la regione ammissibile e determinare per via grafica i valori ottimi delle variabili  $x_1$  ed  $x_2$ .
- Porre il problema in forma standard e determinare la soluzione base corrispondente alla base costituita da  $x_1$ ,  $x_2$  e dalle variabili surplus associate ai vincoli relativi alle sostanze  $C$  e  $D$ .
- Discutere l'ammissibilità della soluzione ottenuta al punto c).

### Soluzione

- a) Esprimendo  $x_1$  ed  $x_2$  in Kg, ed osservando che il costo al Kg dei mangimi  $M$  ed  $N$  è di 2€ e 3€, rispettivamente, si ottiene

$$\begin{aligned} \min z = & \quad 2x_1 + 3x_2 \\ 100x_1 & \geq 400 \\ 100x_2 & \geq 600 \\ 100x_1 + 200x_2 & \geq 2000 \\ 100x_1 + 100x_2 & \geq 1300 \\ x_1, \quad x_2 & \geq 0 \end{aligned}$$

- b) La regione ammissibile (illimitata), la direzione del gradiente della funzione obiettivo e la soluzione ottima sono mostrati in figura 6.2.

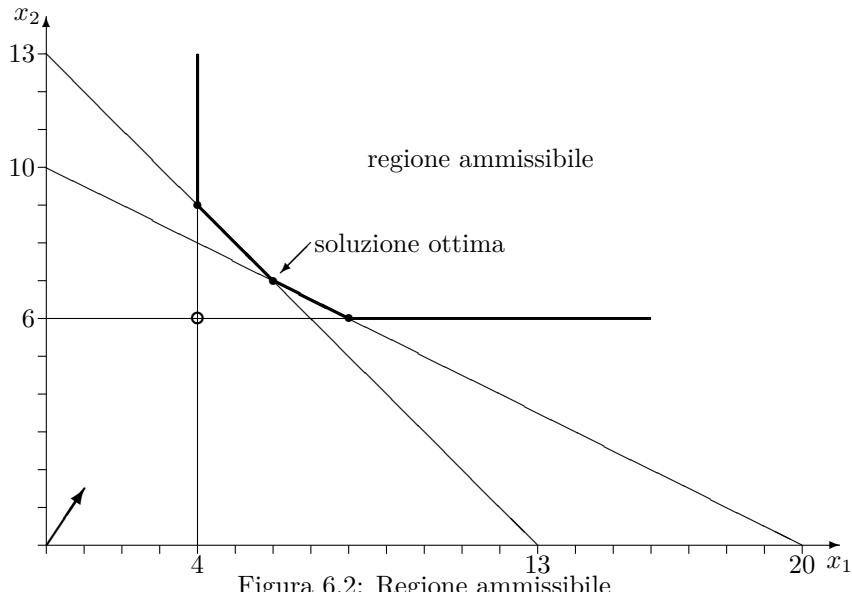


Figura 6.2: Regione ammissibile

I valori ottimi di  $x_1$  ed  $x_2$  sono dati dal sistema definito dalle equazioni corrispondenti agli ultimi due vincoli:

$$\begin{cases} x_1 + 2x_2 = 20 \\ x_1 + x_2 = 13 \end{cases}$$

Si ottiene  $x_1 = 6$ ,  $x_2 = 7$ ; il costo della dieta ottima ammonta pertanto a 33 € giornalieri per animale.

c) Dividendo i vincoli per 100 e ponendo il modello in forma standard, si ottiene

$$\begin{aligned}
 \min z &= 2x_1 + 3x_2 \\
 x_1 - x_3 &= 4 \\
 x_2 - x_4 &= 6 \\
 x_1 + 2x_2 - x_5 &= 20 \\
 x_1 + x_2 - x_6 &= 13 \\
 x_1, x_2, x_3, x_4, x_5, x_6 &\geq 0
 \end{aligned}$$

Uguagliando a 0 le variabili fuori base  $x_3$  ed  $x_4$  si ha il sistema

$$\left\{ \begin{array}{rcl} x_1 & = & 4 \\ x_2 & = & 6 \\ x_1 + 2x_2 - x_5 & = & 20 \\ x_1 + x_2 - x_6 & = & 13 \end{array} \right.$$

da cui si ricava facilmente:  $x_1 = 4$ ,  $x_2 = 6$ ,  $x_5 = -4$ ,  $x_6 = -3$ . La soluzione base richiesta è quindi  $x = (4, 6, 0, 0, -4, -3)$ .

- d)** La soluzione ottenuta non è ammissibile in quanto due variabili hanno valore negativo. Dal punto di vista grafico si nota d'altronde che tale soluzione, indicata con  $\circ$  in figura 6.2, si trova al di fuori della regione ammissibile.

### Esercizio 6.3.

Sia dato il problema di programmazione lineare

$$\begin{aligned} \min z = & x_1 - x_2 - 2x_3 - 3x_4 \\ & x_2 + 2x_3 + 3x_4 = 15 \\ & 2x_2 + x_3 + 5x_4 = 20 \\ & x_1 + x_2 + 2x_3 + x_4 = 10 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

- a) Semplificare, mediante un semplice ragionamento la funzione obiettivo.
- b) Risolvere il problema con l'algoritmo del simplex utilizzando il metodo delle due fasi ed introducendo solo le variabili artificiali necessarie. Si scelga per il pivoting la colonna con il costo relativo negativo di massimo valore assoluto.

### Soluzione

- a) Si osservi che, a causa del primo vincolo, in ogni soluzione ammissibile la somma degli ultimi tre termini della funzione obiettivo vale 15. Pertanto questa può essere semplificata come segue:

$$\min z = -15 + \min x_1.$$

- b) La prima colonna della matrice dei coefficienti del problema è una colonna di matrice identità, per cui è sufficiente aggiungere al problema due sole variabili ausiliarie.

**Fase 1:**

	$x_1^a$	$x_2^a$	$x_1$	$x_2$	$x_3$	$x_4$
$-w$	0	1	1	0	0	0
$x_1^a$	15	1	0	0	1	2
$x_2^a$	20	0	1	0	2	1
$x_1$	10	0	0	1	1	2

	$x_1^a$	$x_2^a$	$x_1$	$x_2$	$x_3$	$x_4$
$-w$	-35	0	0	0	-3	-3
$x_1^a$	15	1	0	0	1	2
$x_2^a$	20	0	1	0	2	1
$x_1$	10	0	0	1	1	2

	$x_1^a$	$x_2^a$	$x_1$	$x_2$	$x_3$	$x_4$
$-w$	-3	0	$\frac{8}{5}$	0	$\frac{1}{5}$	$-\frac{7}{5}$
$x_1^a$	3	1	$-\frac{3}{5}$	0	$-\frac{1}{5}$	$\frac{7}{5}$
$x_4$	4	0	$\frac{1}{5}$	0	$\frac{2}{5}$	$\frac{1}{5}$
$x_1$	6	0	$-\frac{1}{5}$	1	$\frac{3}{5}$	$\frac{9}{5}$

	$x_1^a$	$x_2^a$	$x_1$	$x_2$	$x_3$	$x_4$
$-w$	0	1	1	0	0	0
$x_3$	$\frac{15}{7}$	$\frac{5}{7}$	$-\frac{3}{7}$	0	$-\frac{1}{7}$	1
$x_4$	$\frac{25}{7}$	$-\frac{1}{7}$	$\frac{2}{7}$	0	$\frac{3}{7}$	0
$x_1$	$\frac{15}{7}$	$-\frac{9}{7}$	$\frac{4}{7}$	1	$\frac{6}{7}$	0

Fase 2:

	$x_1$	$x_2$	$x_3$	$x_4$
$-z$	0	1	0	0
$x_3$	$\frac{15}{7}$	0	$-\frac{1}{7}$	1
$x_4$	$\frac{25}{7}$	0	$\frac{3}{7}$	0
$x_1$	$\frac{15}{7}$	1	$\frac{6}{7}$	0

		$x_1$	$x_2$	$x_3$	$x_4$
$-z$	$-\frac{15}{7}$	0	$-\frac{6}{7}$	0	0
$x_3$	$\frac{15}{7}$	0	$-\frac{1}{7}$	1	0
$x_4$	$\frac{25}{7}$	0	$\frac{3}{7}$	0	1
$x_1$	$\frac{15}{7}$	1	$\frac{6}{7}$	0	0

		$x_1$	$x_2$	$x_3$	$x_4$
$-z$	0	1	0	0	0
$x_3$	$\frac{5}{2}$	$\frac{1}{6}$	0	1	0
$x_4$	$\frac{5}{2}$	$-\frac{1}{2}$	0	0	1
$x_2$	$\frac{5}{2}$	$\frac{7}{6}$	1	0	0

La soluzione ottima è dunque  $x_1 = 0, x_2 = x_3 = x_4 = \frac{5}{2}$  e vale  $-15$ .

### Esercizio 6.4. Mix di Produzione.

Una società produce tre articoli  $A$ ,  $B$  e  $C$  i cui prezzi di vendita sono, rispettivamente, 30, 40 e 20 € per pezzo. I pezzi di tipo  $A$  producono un profitto pari al 10% del prezzo di vendita, mentre quelli di tipo  $B$  e  $C$  producono un profitto pari al 5% del prezzo di vendita. La società desidera ottenere un fatturato mensile non inferiore ad 8 milioni di €. La produzione comporta 20 grammi di materiale di scarto per ogni pezzo di tipo  $A$  e 10 grammi per ogni pezzo di tipo  $B$  o  $C$  prodotto. In un mese non è possibile smaltire più di 2 tonnellate di materiale di scarto.

Si definisca il modello di programmazione lineare in grado di determinare i livelli mensili di produzione dei tre articoli in modo da massimizzare il profitto globale. A tal fine si esprimano i livelli di produzione in centinaia di migliaia di pezzi e si assuma che le frazioni di pezzo siano trascurabili ai fini della determinazione della soluzione ottima. Si risolva il problema utilizzando il metodo delle due fasi, introducendo solo le variabili artificiali necessarie ed utilizzando la regola di Bland.

### Soluzione

a) Si utilizzano tre variabili  $x_1, x_2$  ed  $x_3$ , che costituiscono il numero di pezzi da produrre in un mese (espresso in centinaia di migliaia) degli articoli  $A, B$  e  $C$ , rispettivamente. Esprimendo i profitti in centinaia di migliaia di €, il modello di programmazione lineare è

$$\begin{aligned} \max z = & 3x_1 + 2x_2 + x_3 \\ & 2x_1 + x_2 + x_3 \leq 2 \\ & 3x_1 + 4x_2 + 2x_3 \geq 8 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

Ponendo il modello in forma standard, si ottiene

$$\begin{aligned} \min -z = & -3x_1 - 2x_2 - x_3 \\ & 2x_1 + x_2 + x_3 + x_4 = 2 \\ & 3x_1 + 4x_2 + 2x_3 - x_5 = 8 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0 \end{aligned}$$

La quarta colonna della matrice dei coefficienti del problema è una colonna di matrice identità, per cui è sufficiente aggiungere al problema una sola variabile ausiliaria.

**Fase 1:**

	$x_1^a$	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$
$-w$	0	1	0	0	0	0
$x_4$	2	0	2	1	1	0
$x_1^a$	8	1	3	4	2	0

	$x_1^a$	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$
$-w$	-8	0	-3	-4	-2	0
$x_4$	2	0	(2)	1	1	1
$x_1^a$	8	1	3	4	2	0

	$x_1^a$	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$
$-w$	-5	0	0	$-\frac{5}{2}$	$-\frac{1}{2}$	$\frac{3}{2}$
$x_1$	1	0	1	( $\frac{1}{2}$ )	$\frac{1}{2}$	$\frac{1}{2}$
$x_1^a$	5	1	0	$\frac{5}{2}$	$\frac{1}{2}$	$-\frac{3}{2}$

	$x_1^a$	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$
$-w$	0	0	5	0	2	4
$x_2$	2	0	2	1	1	0
$x_1^a$	0	1	-5	0	-2	-4

Poichè la variabile artificiale è nella base finale, occorre un ulteriore pivoting. Scegliendo la variabile  $x_5$  (che comporta una minore quantità di calcoli) si ottiene

	$x_1^a$	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$
$-w$	0	1	0	0	0	0
$x_2$	2	0	2	1	1	0
$x_5$	0	-1	5	0	2	4

**Fase 2:**

	$x_1^a$	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$
$z$	0	0	-3	-2	-1	0
$x_2$	2	0	2	1	1	0
$x_5$	0	-1	5	0	2	4

	$x_1^a$	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$
$z$	4	0	1	0	1	2
$x_2$	2	0	2	1	1	0
$x_5$	0	-1	5	0	2	4

La soluzione ottima, che prevede la produzione di 200 000 pezzi dell'articolo  $B$  e di nessun pezzo degli articoli  $A$  e  $C$ , consente un profitto di 400 000 €.

## Esercizio 6.5.

Un'azienda produce due articoli  $A$  e  $B$  che consentono un profitto per ogni pezzo venduto di 3€ e 1€, rispettivamente. La lavorazione di ciascun articolo di tipo  $A$  richiede due unità di materia prima  $M$  e produce sei unità di sottoprodotto  $P$ , mentre ciascun articolo di tipo  $B$  richiede una unità di  $M$  e tre unità di  $P$ . In magazzino sono disponibili 4000 unità di  $M$  e 10 000 unità di  $P$ .

Il 10% delle confezioni dell'articolo  $A$  ed il 20% delle confezioni dell'articolo  $B$  contengono un omaggio a sorpresa. Il numero degli omaggi inseriti nelle confezioni prodotte non deve essere inferiore a 400.

Si suppongano accettabili soluzioni frazionarie.

- Definire il modello di programmazione lineare per determinare la produzione di massimo profitto. Si esprimano i quantitativi prodotti in migliaia di articoli ed i profitti in migliaia di €.
- Risolvere il problema mediante l'algoritmo del simplex, utilizzando il metodo delle due fasi ed introducendo solo le variabili artificiali necessarie. In entrambe le fasi si scelga per il pivoting la colonna avente il costo relativo negativo con il massimo valore assoluto.
- Disegnare la regione ammissibile ed indicare le soluzioni distinte corrispondenti ai tableau di entrambe le fasi.

## Soluzione

- a) Si utilizzano due variabili  $x_1$  ed  $x_2$ , che costituiscono le migliaia di articoli  $A$  e  $B$ , rispettivamente, che debbono essere prodotti:

$$\begin{aligned} \max z = & 3x_1 + x_2 \\ & 2x_1 + x_2 \leq 4 \\ & -6x_1 + 3x_2 \leq 10 \\ & x_1 + 2x_2 \geq 4 \\ & x_1, x_2 \geq 0 \end{aligned}$$

- b) Ponendo il modello in forma standard, si ottiene

$$\begin{aligned} \min -z = & -3x_1 - x_2 \\ & 2x_1 + x_2 + x_3 = 4 \\ & -6x_1 + 3x_2 + x_4 = 10 \\ & x_1 + 2x_2 - x_5 = 4 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0 \end{aligned}$$

È sufficiente aggiungere al problema una sola variabile ausiliaria (per il terzo vincolo):

**Fase 1:**

	$x_1^a$	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$
$-w$	0	1	0	0	0	0
$x_3$	4	0	2	1	1	0
$x_4$	10	0	-6	3	0	1
$x_1^a$	4	1	1	2	0	0

	$x_1^a$	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$
$-w$	-4	0	-1	-2	0	0
$x_3$	4	0	2	1	1	0
$x_4$	10	0	-6	3	0	1
$x_1^a$	4	1	1	(2)	0	0

	$x_1^a$	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$
$-w$	0	1	0	0	0	0
$x_3$	2	$-\frac{1}{2}$	$\frac{3}{2}$	0	1	0
$x_4$	4	$-\frac{3}{2}$	$-\frac{15}{2}$	0	0	1
$x_2$	2	$\frac{1}{2}$	$\frac{1}{2}$	1	0	0

**Fase 2:**

	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$
$z$	0	-3	-1	0	0
$x_3$	2	$\frac{3}{2}$	0	1	0
$x_4$	4	$-\frac{15}{2}$	0	0	1
$x_2$	2	$\frac{1}{2}$	1	0	0

	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$
$z$	2	$-\frac{5}{2}$	0	0	$-\frac{1}{2}$
$x_3$	2	( $\frac{3}{2}$ )	0	1	0
$x_4$	4	$-\frac{15}{2}$	0	0	1
$x_2$	2	$\frac{1}{2}$	1	0	0

	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$
$z$	$\frac{16}{3}$	0	0	$\frac{5}{3}$	0
$x_1$	$\frac{4}{3}$	1	0	$\frac{2}{3}$	0
$x_4$	14	0	0	5	1
$x_2$	$\frac{4}{3}$	0	1	$-\frac{1}{3}$	0

La soluzione ottima consiste dunque nel produrre  $\frac{4}{3}$  di migliaio di articolo  $A$  e  $\frac{4}{3}$  di migliaio di articolo  $B$ . Tale soluzione determina un profitto complessivo di  $\frac{16}{3}$  migliaia di €.

- c) La regione ammissibile e la direzione del gradiente della funzione obiettivo sono mostrati in figura 6.3. Le lettere  $\alpha$ ,  $\beta$  e  $\gamma$  indicano, rispettivamente, le soluzioni corrispondenti al primo, terzo e sesto tableau.

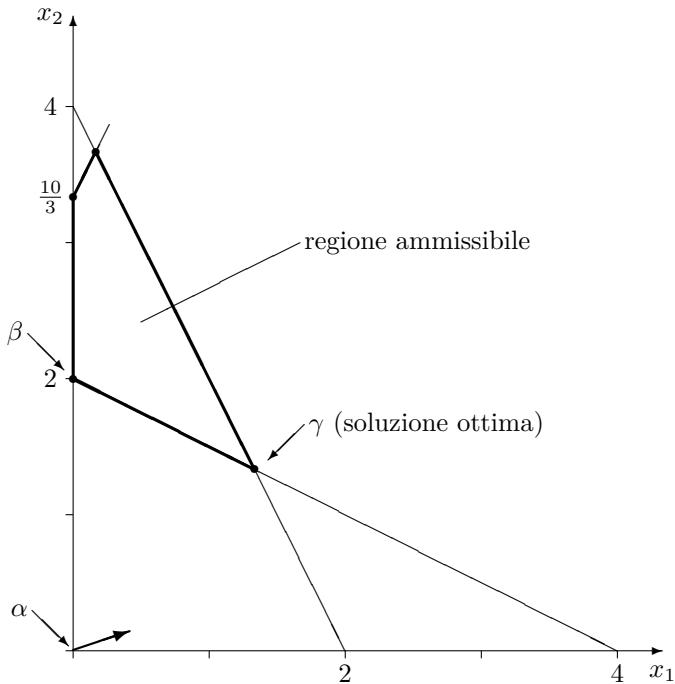


Figura 6.3: Regione ammissibile

### Esercizio 6.6.

Sia dato il problema di programmazione lineare

$$\begin{aligned} \max z = & 30x_1 + 20x_2 \\ & x_2 \leq 4 \\ & x_1 + 2x_2 \leq 10 \\ & 2x_1 + x_2 \leq 12 \\ & x_1, x_2 \geq 0 \end{aligned}$$

- a) Riscriverlo ponendolo in forma standard.
- b) Disegnare la regione ammissibile e determinare per via grafica la soluzione ottima ed il suo valore.
- c) Etichettare i vertici del politopo con le lettere  $A, B, C, \dots$  partendo dall'origine e procedendo in senso orario. Mediante semplici considerazioni stabilire, per ciascun vertice del politopo, quali variabili appartengono alla base nella soluzione base ammissibile corrispondente.
- d) Considerare la funzione obiettivo parametrica

$$30x_1 + \alpha x_2$$

dove  $\alpha$  può assumere qualunque valore positivo. Mediante considerazioni di tipo geometrico determinare per quali valori di  $\alpha$  la soluzione base individuata al punto b) rimane ottima.

### Soluzione

- a) Introducendo le variabili slack  $x_3, x_4$  ed  $x_5$  si ottiene il problema in forma standard

$$\begin{aligned} \min -z = & -30x_1 - 20x_2 \\ & x_2 + x_3 = 4 \\ & x_1 + 2x_2 + x_4 = 10 \\ & 2x_1 + x_2 + x_5 = 12 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0 \end{aligned}$$

- b) La regione ammissibile del problema e la direzione del gradiente della funzione obiettivo sono mostrate in figura 6.4. La soluzione ottima coincide dunque con il vertice  $D$ .

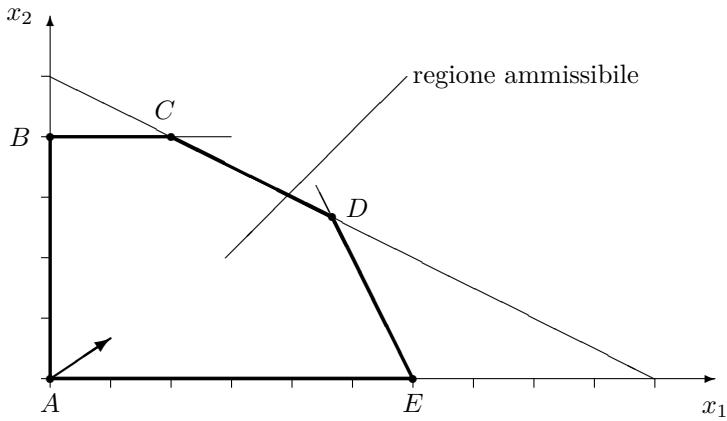


Figura 6.4: Regione ammissibile

Le coordinate del punto D si ricavano dal sistema

$$\begin{cases} x_1 + 2x_2 = 10 \\ 2x_1 + x_2 = 12 \end{cases}$$

Si ottiene la soluzione  $x_1 = \frac{14}{3}$  ed  $x_2 = \frac{8}{3}$ . Pertanto il valore della soluzione ottima è  $z = \frac{580}{3}$ .

c) Si osservi innanzitutto che in corrispondenza di un vertice del politopo la variabile slack associata ad un vincolo non attivo (ossia non verificato con uguaglianza) deve avere valore diverso da zero e pertanto deve far parte della base corrispondente. Fanno ovviamente parte della base anche le variabili del problema originale che assumono in corrispondenza del vertice valore diverso da zero.

Le variabili in base nella soluzione base ammissibile corrispondente a ciascun vertice del politopo sono quindi quelle riportate nella seguente tabella:

vertice	variabili in base
A	$x_3, x_4, x_5$
B	$x_2, x_4, x_5$
C	$x_1, x_2, x_5$
D	$x_1, x_2, x_3$
E	$x_1, x_3, x_4$

**d)** Al variare di  $\alpha$  varia la direzione del gradiente della funzione obiettivo. Tale direzione, come è noto, è perpendicolare alle rette cui corrisponde valore costante della funzione obiettivo.

Come si è visto, il vertice  $D$  corrispondente alla soluzione ottima individuata al punto b) è dato dall'intersezione degli iperpiani

$$x_1 + 2x_2 = 10 \quad (\text{spigolo } CD) \text{ e}$$

$$2x_1 + x_2 = 12 \quad (\text{spigolo } DE)$$

Pertanto se il valore di  $\alpha$  viene aumentato rispetto al valore corrente 20, il vertice  $D$  continua a produrre la soluzione ottima fino a quando la famiglia di rette equi-costo non diviene parallela allo spigolo  $CD$ , ossia fino a quando il gradiente  $(30, \alpha)$  non diviene perpendicolare a tale spigolo.

La condizione limite si ha dunque quando il prodotto tra il gradiente ed il vettore associato alla direzione dello spigolo è nullo, ossia

$$(30 \quad \alpha) \begin{pmatrix} 10 \\ -5 \end{pmatrix} = 300 - 5\alpha = 0 \Rightarrow \alpha = 60$$

Con analoghe considerazioni si verifica che, diminuendo il valore di  $\alpha$ , la condizione limite si ha quando il prodotto tra il gradiente ed il vettore associato alla direzione dello spigolo  $DE$  è nullo, ossia

$$(30 \quad \alpha) \begin{pmatrix} 6 \\ -12 \end{pmatrix} = 180 - 12\alpha = 0 \Rightarrow \alpha = 15$$

Pertanto, come è illustrato nella figura 6.5, il vertice  $D$  rimane ottimo per tutti i valori di  $\alpha$  compresi tra 15 e 60.

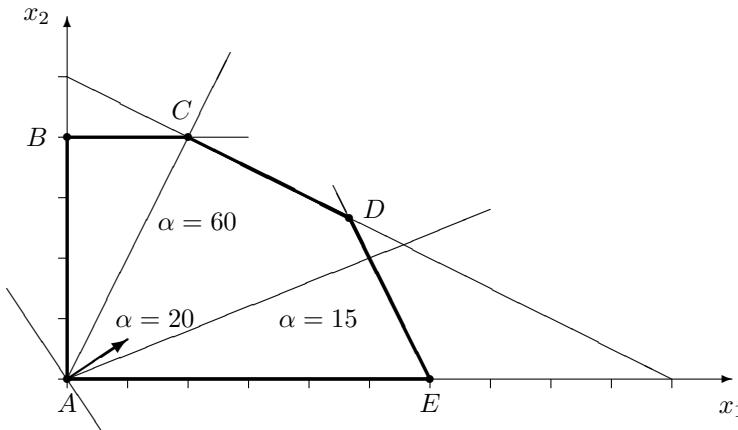


Figura 6.5: Analisi di sensitività

### Esercizio 6.7. Miscelazione di caffè.

Una ditta produce miscele di caffè di due diverse qualità (extra ed economica) utilizzando due varietà base (arabica e robusta). Per la miscela extra vengono utilizzate tre parti di arabica e due di robusta, per quella economica due parti di arabica e tre di robusta. La ditta dispone di 12 tonnellate di ciascuna varietà. I prezzi di vendita per tonnellata sono 6000€ per la miscela extra e 4000€ per la miscela economica.

- Definire il modello di programmazione lineare che massimizza l'introito complessivo e risolverlo con il metodo del simplex, scegliendo per il primo pivoting la variabile corrispondente alla qualità extra.
- Per motivi di mercato è preferibile che entrambe le miscele vengano effettivamente prodotte. Nel caso tale condizione non sia verificata nella soluzione trovata, dire, osservando il tableau, se esiste una soluzione che la verifica e produce lo stesso introito. Se la risposta è affermativa, determinare la nuova soluzione agendo direttamente sul tableau ottimo del punto a), senza introdurre ulteriori vincoli.
- Disegnare la regione ammissibile e commentare, in base a considerazioni di tipo geometrico, il risultato del punto b).

### Soluzione

- Si utilizzano due variabili  $x_1$  ed  $x_2$ , che costituiscono il numero di tonnellate da produrre di caffè di qualità extra ed economica, rispettivamente. Il modello è

pertanto

$$\begin{aligned} \max z = & 6x_1 + 4x_2 \\ & \frac{2}{5}x_1 + \frac{3}{5}x_2 \leq 12 \\ & \frac{3}{5}x_1 + \frac{2}{5}x_2 \leq 12 \\ & x_1, x_2 \geq 0 \end{aligned}$$

in cui il ricavo totale  $z$  è espresso in migliaia di €. Moltiplicando i vincoli per 5 (al fine di ottenere coefficienti interi) si ricava il modello in forma standard:

$$\begin{aligned} \min -z = & -6x_1 - 4x_2 \\ & 2x_1 + 3x_2 + x_3 = 60 \\ & 3x_1 + 2x_2 + x_4 = 60 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

Poichè le ultime due colonne del tableau costituiscono una base iniziale ammissibile, non è necessario applicare la Fase 1.

	$x_1$	$x_2$	$x_3$	$x_4$
$z$	0	-6	-4	0
$x_3$	60	2	3	1
$x_4$	60	(3)	2	0

	$x_1$	$x_2$	$x_3$	$x_4$
$z$	120	0	0	2
$x_3$	20	0	( $\frac{5}{3}$ )	1
$x_1$	20	1	$\frac{2}{3}$	0

La soluzione ottima consiste dunque nel produrre 20 tonnellate di caffè di qualità extra e nessuna di qualità economica; il relativo ricavo è di 120000 €.

**b)** La soluzione ottima trovata al punto a) non produce entrambe le qualità di caffè. Esaminando il tableau ottimo, si osserva che la colonna corrispondente alla variabile fuori base  $x_2$  ha costo relativo nullo. È quindi possibile far entrare tale variabile in base con un'operazione di pivoting, ottenendo una soluzione con lo stesso ricavo di quella del punto a).

	$x_1$	$x_2$	$x_3$	$x_4$
$z$	120	0	0	2
$x_2$	12	0	1	$\frac{3}{5}$ $-\frac{2}{5}$
$x_1$	12	1	0	$-\frac{2}{5}$ $\frac{3}{5}$

La nuova soluzione prevede di produrre 12 tonnellate di ciascuna qualità di caffè.

c) La regione ammissibile del problema è mostrata in figura. Si osservi che il gradiente della funzione obiettivo è perpendicolare allo spigolo  $AB$ . Tutte le soluzioni corrispondenti ai punti di tale spigolo (ed in particolare le soluzioni base corrispondenti ai vertici  $A$  e  $B$ ) sono dunque soluzioni ottime equivalenti del problema.

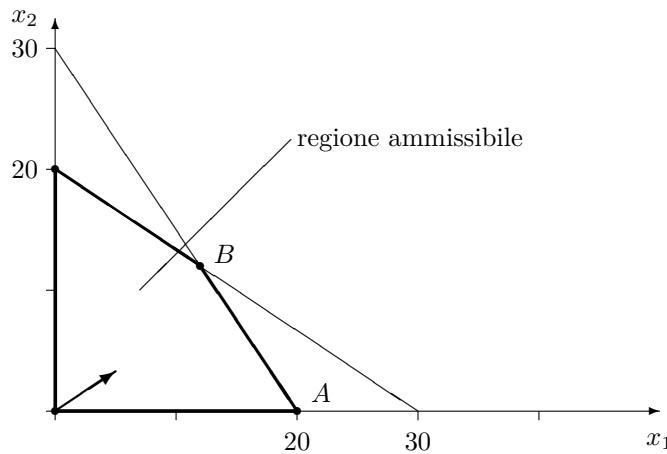


Figura 6.6: Regione Ammissibile

### Esercizio 6.8. Mix di produzione parametrico.

Una fabbrica produce 2 tipi di materiale ( $A$  e  $B$ ). La produzione di ogni chilogrammo di materiale  $A$  richiede 3 ore di preparazione e 2 di rifinitura, mentre per la produzione di ogni chilogrammo di materiale  $B$  sono necessarie 5 ore di preparazione e 2 di rifinitura. I macchinari addetti alla preparazione sono disponibili per non più di 40 ore giornaliere complessive, quelli addetti alla rifinitura per non più di 30. Il profitto dato dal prodotto  $B$  è  $k$  volte quello dato dal prodotto  $A$ , dove  $k$ , a seconda del mercato, può assumere qualunque valore compreso tra 1 e 2.

Determinare, servendosi di un modello di programmazione lineare e dell'algoritmo del simplex con regola di Bland, i quantitativi giornalieri da produrre, a seconda del valore di  $k$ , per massimizzare il profitto complessivo.

### Soluzione

Si utilizzano due variabili  $x_1$  ed  $x_2$ , che costituiscono, rispettivamente, la produzione giornaliera ottima del prodotto  $A$  e del prodotto  $B$ .

Assumendo che il prodotto  $A$  abbia profitto unitario, il modello di programmazione lineare è il seguente:

$$\begin{aligned} \max z = & x_1 + k x_2 \\ & 3 x_1 + 5 x_2 \leq 40 \\ & 2 x_1 + 2 x_2 \leq 30 \\ & x_1, x_2 \geq 0 \end{aligned}$$

che, posto in forma standard, diventa

$$\begin{aligned} \min -z = & -x_1 - k x_2 \\ & 3 x_1 + 5 x_2 + x_3 = 40 \\ & 2 x_1 + 2 x_2 + x_4 = 30 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

Poichè le ultime due colonne del tableau costituiscono una base iniziale ammissibile, non è necessario applicare la Fase 1.

		$x_1$	$x_2$	$x_3$	$x_4$
$z$	0	-1	$-k$	0	0
$x_3$	40	(3)		5	1
$x_4$	30	2	2	0	1

	$x_1$	$x_2$	$x_3$	$x_4$
$z$	$\frac{40}{3}$	$0 - k + \frac{5}{3}$	$\frac{1}{3}$	0
$x_1$	$\frac{40}{3}$	1	$\frac{5}{3}$	$\frac{1}{3}$
$x_4$	$\frac{10}{3}$	0	$-\frac{4}{3}$	$-\frac{2}{3}$

Tutti i coefficienti della riga 0 risultano non negativi, con l'eventuale eccezione di quello dipendente dal parametro  $k$ . Occorre pertanto considerare due casi.

**Caso 1:**  $k \leq \frac{5}{3}$

Il costo relativo della colonna 2 risulta non negativo, per cui in questo caso il tableau ottenuto produce la soluzione ottima

$$x_1 = \frac{40}{3}, \quad x_2 = 0$$

Il valore di questa soluzione è  $z = \frac{40}{3}$ , ossia  $\frac{40}{3}$  del profitto del prodotto A. Si osservi che tale valore è indipendente dal parametro  $k$ .

**Caso 2:**  $k > \frac{5}{3}$

Il costo relativo della colonna 2 risulta negativo, per cui occorre proseguire nell'applicazione dell'algoritmo del simplesso.

	$x_1$	$x_2$	$x_3$	$x_4$
$z$	$\frac{40}{3}$	$0 - k + \frac{5}{3}$	$\frac{1}{3}$	0
$x_1$	$\frac{40}{3}$	1	$\frac{5}{3}$	$\frac{1}{3}$
$x_4$	$\frac{10}{3}$	0	$-\frac{4}{3}$	$-\frac{2}{3}$

	$x_1$	$x_2$	$x_3$	$x_4$
$z$	$\frac{3}{5}k - 1$	0	$\frac{1}{5}k$	0
$x_2$	8	$\frac{3}{5}$	1	$\frac{1}{5}$
$x_4$	14	$\frac{4}{5}$	0	$-\frac{2}{5}$

Poichè siamo nell'ipotesi  $k > \frac{5}{3}$ , il costo relativo della colonna 1 risulta non negativo, per cui il tableau produce la soluzione ottima

$$x_1 = 0, \quad x_2 = 8$$

Il valore di questa soluzione è  $z = 8k$ , ossia  $8k$  volte il profitto del prodotto  $A$ . Si osservi che tale valore dipende dal parametro  $k$ .

### Esercizio 6.9. Vincolo ridondante in Fase 1

Un'azienda produce quattro tipi di solvente utilizzando, tra le altre, tre sostanze chimiche altamente tossiche ( $A$ ,  $B$  e  $C$ ) nelle seguenti percentuali:

- solvente 1: 10 % di  $A$ , 50 % di  $B$ , 30 % di  $C$ ;
- solvente 2: 20 % di  $A$ , 50 % di  $B$ , 10 % di  $C$ ;
- solvente 3: 10 % di  $A$ , 20 % di  $B$ ;
- solvente 4: 10 % di  $B$ , 10 % di  $C$ .

I profitti, in € per quintale, sono 500, 400, 100 e 100 rispettivamente per i solventi 1, 2, 3 e 4. L'azienda dispone di tre serbatoi contenenti rispettivamente una tonnellata di  $A$ , tre tonnellate di  $B$  ed una tonnellata di  $C$ : al termine della produzione si devono pulire tutti i serbatoi, per cui tali sostanze debbono essere completamente utilizzate nella produzione, dato che la loro tossicità non ne consente l'eliminazione in altro modo.

Determinare la produzione ottima mediante metodo delle due fasi e regola di Bland.

### Soluzione

Utilizzando quattro variabili  $x_1$ ,  $x_2$ ,  $x_3$  ed  $x_4$  (numero di tonnellate da produrre di solvente di tipo 1, 2, 3 e 4, rispettivamente), si ottiene il modello di programmazione lineare

$$\begin{aligned} \max z = & 5000 x_1 + 4000 x_2 + 1000 x_3 + 1000 x_4 \\ & 0.1 x_1 + 0.2 x_2 + 0.1 x_3 = 1 \\ & 0.5 x_1 + 0.5 x_2 + 0.2 x_3 + 0.1 x_4 = 3 \\ & 0.3 x_1 + 0.1 x_2 + 0.1 x_4 = 1 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

Esprimendo i profitti in milgiaia di € e moltiplicando i vincoli per 10 si ha

$$\begin{aligned} \min -z = & -5 x_1 - 4 x_2 - x_3 - x_4 \\ & x_1 + 2 x_2 + x_3 = 10 \\ & 5 x_1 + 5 x_2 + 2 x_3 + x_4 = 30 \\ & 3 x_1 + x_2 + x_4 = 10 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

**Fase 1:**

	$x_1^a$	$x_2^a$	$x_3^a$	$x_1$	$x_2$	$x_3$	$x_4$
$-w$	0	1	1	1	0	0	0
$x_1^a$	10	1	0	0	1	2	1
$x_2^a$	30	0	1	0	5	5	2
$x_3^a$	10	0	0	1	3	1	0

	$x_1^a$	$x_2^a$	$x_3^a$	$x_1$	$x_2$	$x_3$	$x_4$
$-w$	-50	0	0	0	-9	-8	-3
$x_1^a$	10	1	0	0	1	2	1
$x_2^a$	30	0	1	0	5	5	2
$x_3^a$	10	0	0	1	3	1	0

	$x_1^a$	$x_2^a$	$x_3^a$	$x_1$	$x_2$	$x_3$	$x_4$
$-w$	-20	0	0	3	0	-5	-3
$x_1^a$	$\frac{20}{3}$	1	0	$-\frac{1}{3}$	0	$\frac{5}{3}$	$-\frac{1}{3}$
$x_2^a$	$\frac{40}{3}$	0	1	$-\frac{5}{3}$	0	$\frac{10}{3}$	$-\frac{2}{3}$
$x_1$	$\frac{10}{3}$	0	0	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{3}$

	$x_1^a$	$x_2^a$	$x_3^a$	$x_1$	$x_2$	$x_3$	$x_4$
$-w$	0	3	0	2	0	0	0
$x_2$	4	$\frac{3}{5}$	0	$-\frac{1}{5}$	0	1	$\frac{3}{5}$
$x_2^a$	0	-2	1	-1	0	0	0
$x_1$	2	$-\frac{1}{5}$	0	$\frac{2}{5}$	1	0	$-\frac{1}{5}$

Al termine della Fase 1, pur essendo  $w = 0$ , la variabile  $x_2^a$  è in base sulla riga 2. Poichè tutti i coefficienti della riga 2 corrispondenti alle variabili del problema originale sono nulli, la riga 2 risulta ridondante (infatti essa è ottenibile come combinazione lineare delle righe 1 e 3 con coefficienti 2 ed 1, rispettivamente).

Si elimina pertanto la riga 2, ottenendo così una base iniziale ammissibile, e si applica la Fase 2.

	$x_1$	$x_2$	$x_3$	$x_4$
$z$	0	-5	-4	-1
$x_2$	4	0	1	$\frac{3}{5}$
$x_1$	2	1	0	$-\frac{1}{5}$

	$x_1$	$x_2$	$x_3$	$x_4$
$z$	26	0	0	$\frac{2}{5}$
$x_2$	4	0	1	$-\frac{1}{5}$
$x_1$	2	1	0	$-\frac{1}{5}$

La soluzione ottima consiste dunque nel produrre 2 tonnellate di solvente di tipo 1 e 4 tonnellate di solvente di tipo 2; il relativo profitto è di 26000€.

### Esercizio 6.10. Problema illimitato

Un'azienda produce due prodotti chimici,  $A$  e  $B$ . Ogni quintale di prodotto  $A$  richiede materie prime e mano d'opera per un costo complessivo di 3000€ e fornisce come sottoprodotto 3 Kg di acido. La produzione di un quintale di prodotto  $B$  richiede invece 4 Kg di acido e non prevede ulteriori costi. L'acido eventualmente necessario può essere acquistato ad un costo di 2000€ al Kg, mentre non è possibile produrre acido in eccesso dato che esso non può essere smaltito dall'azienda. Ogni quintale di prodotto  $A$  e  $B$  viene venduto ad un prezzo di 1000€ e 9000€, rispettivamente. Esigenze di mercato richiedono infine che la quantità del prodotto  $B$  prodotta non superi quella del prodotto  $A$ .

Supponendo che non esistano vincoli tecnologici sui massimi quantitativi producibili, determinare la produzione di massimo profitto, utilizzando non più di due variabili, impiegando il metodo delle due fasi e la regola di Bland.

Commentare il risultato ottenuto.

### Soluzione

Vengono utilizzate due variabili  $x_1$  ed  $x_2$ , che costituiscono il numero di quintali da produrre di prodotto  $A$  e  $B$ , rispettivamente.

Il profitto ottenibile producendo un quintale di prodotto  $A$  è pari alla differenza tra prezzo di vendita e costo, più il valore dell'acido ottenuto (che altrimenti dovrebbe essere acquistato all'esterno), ossia  $1000 - 3000 + 6000 = 4000$ . Analogamente il profitto ottenibile producendo un quintale di prodotto  $B$  è pari al prezzo di vendita meno il costo dell'acido necessario (indipendentemente dal fatto che questo sia acquistato all'esterno od ottenuto dalla produzione di  $A$ ), ossia  $9000 - 8000 = 1000$ . Il modello di programmazione lineare è pertanto il seguente:

$$\begin{aligned} \max z = & 4x_1 + x_2 \\ & -x_1 + x_2 \leq 0 \\ & 3x_1 - 4x_2 \leq 0 \\ & x_1, x_2 \geq 0 \end{aligned}$$

dove il secondo vincolo esprime la condizione che la soluzione non preveda produzione di acido in eccesso.

Ponendo il modello in forma standard, si ottiene

$$\begin{aligned} \min -z = & -4x_1 - x_2 \\ & -x_1 + x_2 + x_3 = 0 \\ & 3x_1 - 4x_2 + x_4 = 0 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

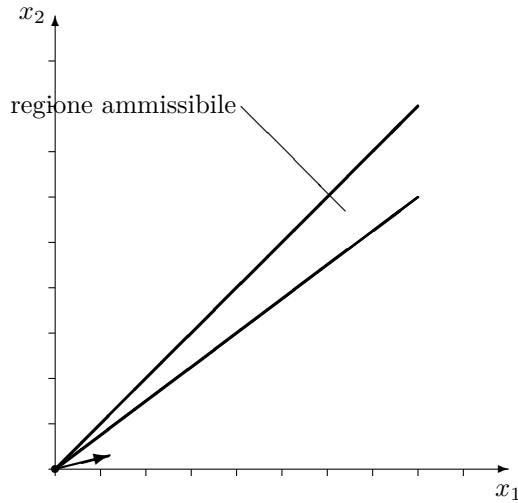
Poichè le ultime due colonne del tableau costituiscono una base iniziale ammissibile, non è necessario applicare la Fase 1.

	$x_1$	$x_2$	$x_3$	$x_4$	
$z$	0	-4	-1	0	0
$x_3$	0	-1	1	1	0
$x_4$	0	(3)	-4	0	1

	$x_1$	$x_2$	$x_3$	$x_4$	
$z$	0	0	$-\frac{19}{3}$	0	$\frac{4}{3}$
$x_3$	0	0	$-\frac{1}{3}$	1	$\frac{1}{3}$
$x_1$	0	1	$-\frac{4}{3}$	0	$\frac{1}{3}$

Poichè tutti i coefficienti della colonna corrispondente alla variabile  $x_2$ , che ha costo relativo negativo, sono minori o uguali a zero, è possibile concludere che la soluzione del problema è illimitata.

La regione ammissibile del problema formulato e la direzione del gradiente della funzione obiettivo sono infatti le seguenti:



Si conclude pertanto che per determinare una produzione effettiva occorrerebbe introdurre nel modello anche vincoli tecnologici sulle capacità produttive dell'azienda.

### Esercizio 6.11.

Un'azienda vinicola produce due tipi di vino: DOC e “da tavola”. Ogni quintale di vino DOC richiede 2 quintali di uva di prima scelta e mezzo quintale di uva di seconda scelta. Ogni quintale di vino da tavola richiede 80 chili di uva di prima scelta, 80 chili di uva di seconda scelta e 10 litri d'acqua. L'azienda dispone di 800 quintali di uva di prima scelta e di 400 quintali di uva di seconda scelta, oltreché, ovviamente, di una quantità illimitata di acqua. Ogni quintale di vino DOC dà un profitto di 50€, ogni quintale di vino da tavola un profitto di 30€. L'azienda ritiene di non poter mettere in commercio più di 300 quintali di vino DOC.

- Definire il modello di programmazione lineare per determinare le quantità da produrre di ciascun tipo di vino per massimizzare il profitto conseguibile.
- Determinare la soluzione ottima mediante l'algoritmo del simplesso e la regola di Bland.
- L'azienda apprende che è possibile che vengano intensificati i controlli anti sofisticazione. In tale ipotesi essa dovrebbe rinunciare all'aggiunta di acqua nel vino da tavola, per cui il corrispondente profitto diminuirebbe a 10€ per quintale. Partendo dal tableau finale ottenuto al punto b), determinare la nuova soluzione ottima.
- Disegnare la regione ammissibile ed indicare i vertici corrispondenti alle soluzioni ottime dei punti b) e c).

### Soluzione

**a)** Vengono utilizzate due variabili  $x_1$  ed  $x_2$ , che costituiscono il numero di quintali da produrre di vino DOC e da tavola, rispettivamente. Il modello di programmazione lineare è

$$\begin{aligned} \max z = & 5x_1 + 3x_2 \\ & 2x_1 + 0.8x_2 \leq 800 \\ & 0.5x_1 + 0.8x_2 \leq 400 \\ & x_1 \leq 300 \\ & x_1, x_2 \geq 0 \end{aligned}$$

in cui il profitto totale  $z$  è espresso in decine di €. Si noti che ovviamente non occorre introdurre il vincolo relativo all'acqua, che avrebbe la forma  $x_2 \leq +\infty$ . Moltiplicando il secondo ed il terzo vincolo per 10 (al fine di ottenere coefficienti interi) e ponendo il modello in forma standard, si ottiene

$$\begin{aligned}
 \min -z = & - 5x_1 - 3x_2 \\
 20x_1 + 8x_2 + x_3 &= 8000 \\
 5x_1 + 8x_2 + x_4 &= 4000 \\
 x_1 + x_5 &= 300 \\
 x_1, x_2, x_3, x_4, x_5 &\geq 0
 \end{aligned}$$

**b)** Poichè le ultime tre colonne del tableau costituiscono una base iniziale

ammisibile, non è necessario applicare la Fase 1.

	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$
$z$	0	-5	-3	0	0
$x_3$	8000	20	8	1	0
$x_4$	4000	5	8	0	1
$x_5$	300	(1)	0	0	1

	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$
$z$	1500	0	-3	0	0
$x_3$	2000	0	(8)	1	0
$x_4$	2500	0	8	0	1
$x_1$	300	1	0	0	1

	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$
$z$	2250	0	0	$\frac{3}{8}$	$0$
$x_2$	250	0	1	$\frac{1}{8}$	$0$
$x_4$	500	0	0	-1	1
$x_1$	300	1	0	0	1

	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$
$z$	$\frac{7000}{3}$	0	0	$\frac{5}{24}$	$\frac{1}{6}$
$x_2$	$\frac{1000}{3}$	0	1	$-\frac{1}{24}$	$\frac{1}{6}$
$x_5$	$\frac{100}{3}$	0	0	$-\frac{1}{15}$	$\frac{1}{15}$
$x_1$	$\frac{800}{3}$	1	0	$\frac{1}{15}$	$-\frac{1}{15}$

La soluzione ottima consiste dunque nel produrre  $\frac{800}{3}$  di quintale di vino DOC e  $\frac{1000}{3}$  di quintale di vino da tavola; si ottiene in tal modo un profitto

complessivo di  $\frac{70}{3}$  migliaia di €, ossia circa 23300€.

c) Nell'ipotesi considerata i vincoli non cambiano, mentre la funzione obiettivo in forma standard diviene

$$\min -z = -5x_1 - x_2$$

Inserendo tale funzione nel tableau finale del punto b) si ottiene

	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$
0	-5	-1	0	0	0
$x_2$	$\frac{1000}{3}$	0	1	$-\frac{1}{24}$	$\frac{1}{6}$
$x_5$	$\frac{100}{3}$	0	0	$-\frac{1}{15}$	$\frac{1}{15}$
$x_1$	$\frac{800}{3}$	1	0	$\frac{1}{15}$	$-\frac{1}{15}$

Per azzerare i costi relativi corrispondenti alle variabili in base si sottraggono alla riga 0 la riga 3 moltiplicata per 5 e la riga 1:

	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$
$z$	$\frac{5000}{3}$	0	0	$\frac{7}{24}$	$-\frac{1}{6}$
$x_2$	$\frac{1000}{3}$	0	1	$-\frac{1}{24}$	$\frac{1}{6}$
$x_5$	$\frac{100}{3}$	0	0	$-\frac{1}{15}$	$\frac{1}{15}$
$x_1$	$\frac{800}{3}$	1	0	$\frac{1}{15}$	$-\frac{1}{15}$

	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$
$z$	1750	0	0	$\frac{1}{8}$	$0$
$x_2$	250	0	1	$\frac{1}{8}$	$0$
$x_4$	500	0	0	-1	1
$x_1$	300	1	0	0	1

La nuova soluzione ottima prevede di produrre 300 quintali di vino DOC e 250 quintali di vino da tavola; il profitto complessivo scende a 17500€.

d) La regione ammissibile del problema è riportata nella figura 6.7.

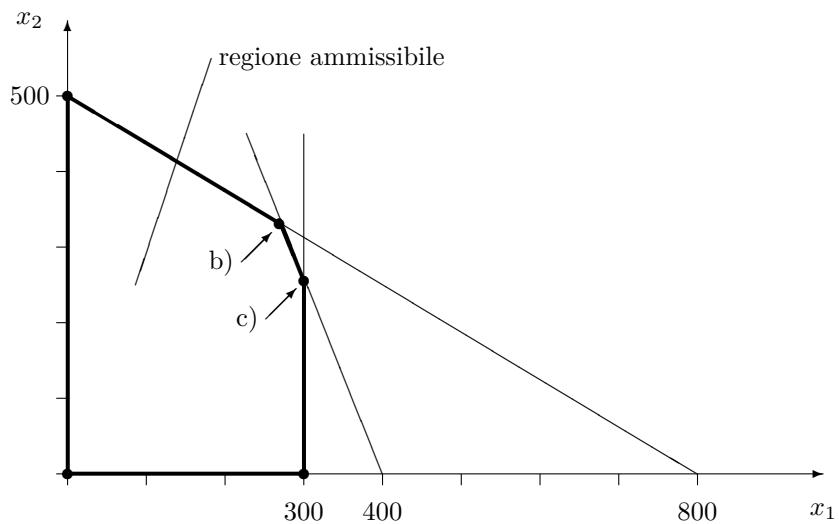


Figura 6.7: Regione ammissibile

### Esercizio 6.12.

Sia dato il problema di programmazione lineare

$$\begin{aligned} \min z = & \quad x_1 - 2x_2 + x_3 \\ & x_1 + 3x_3 = 1 \\ & 3x_1 + 3x_2 - x_3 = 4 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

Risolvere il problema mediante l'algoritmo del simplex, utilizzando il metodo delle due fasi e la regola di Bland. Nella Fase 1 si introducano solo le variabili artificiali necessarie.

### Esercizio 6.13.

Un'azienda produce lotti di due articoli  $A$  e  $B$  utilizzando un materiale  $M$ , di cui sono disponibili otto quintali. La produzione di ogni lotto richiede un quintale di  $M$ : un lotto di  $A$  è costituito da materiale  $M$  puro, mentre in un lotto di  $B$  il materiale  $M$  è diluito in un peso uguale di materia inerte. Esigenze produttive impongono che il peso complessivo dei lotti prodotti sia di almeno cinque quintali, mentre per ragioni di mercato il numero dei lotti di  $A$  prodotti deve superare quello dei lotti di  $B$  prodotti di almeno sei unità. Il profitto ottenibile da un lotto di  $A$  è di 2000€, mentre quello ottenibile da un lotto di  $B$  è di 4000€.

- a) Definire il modello di programmazione lineare per determinare la produzione di massimo profitto.
- b) Stabilire, mediante un semplice ragionamento, se uno dei vincoli è superfluo e, in tal caso, eliminarlo.
- c) Risolvere il problema risultante mediante l'algoritmo del simplex, utilizzando il metodo delle due fasi ed introducendo, nella Fase 1, solo le variabili artificiali necessarie. Si scelga per il pivoting la colonna cui corrisponde il costo relativo negativo di massimo valore assoluto.

### Esercizio 6.14.

Un'azienda chimica produce due composti diversi ( $A$  e  $B$ ) elaborando un unico elemento base  $X$ . Per produrre un quintale di composto  $A$  si consumano tre quintali di  $X$ , mentre per produrre un quintale di composto  $B$  si consumano due quintali di  $X$ . Il profitto unitario dato da  $A$  è doppio di quello dato da  $B$ . L'azienda dispone di sei quintali di  $X$  e, a causa di impegni precedentemente assunti con i propri clienti, deve produrre almeno un quintale di ciascun composto.

- a) Definire il modello di programmazione lineare per determinare la produzione di massimo profitto.
- b) Risolvere il problema mediante l'algoritmo del simplex, utilizzando il metodo delle due fasi ed introducendo, nella Fase 1, solo le variabili artificiali necessarie. Si scelga per il pivoting la colonna cui corrisponde il costo relativo negativo di massimo valore assoluto.
- c) Disegnare la regione ammissibile ed indicarvi le soluzioni corrispondenti a ciascun tableau (di entrambe le fasi).

## Capitolo 7

# Programmazione Lineare Intera

In questo capitolo vengono introdotte le proprietà fondamentali dei problemi di programmazione lineare così definiti

**Definizione 7.1 (Problema PLI).**  $(F, \varphi)$  è un problema di programmazione lineare intera (PLI, o integer linear programming ILP) se la funzione obiettivo  $\varphi$  è lineare

$$\varphi(x) = c^T x = c_1 x_1 + c_2 x_2 + \cdots + c_n x_n,$$

la regione  $F \subseteq \mathbb{R}^n$  è definita da  $g_i(x) \leq 0$  e  $h_j(x) = 0$  ( $i = 1, \dots, m$ ;  $j = 1, \dots, p$ ) con  $g_i, h_j : \mathbb{R}^n \rightarrow \mathbb{R}$  lineare  $\forall i, j$

$$g_i(x) : a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n \geq d_i$$

$$h_j(x) : a_{j1}x_1 + a_{j2}x_2 + \dots + a_{jn}x_n = d_j$$

e le variabili possono assumere solo valori interi.

Nel caso speciale in cui le variabili possano assumere solo valori binari, ossia 0 oppure 1, il problema si dice di programmazione lineare binaria (PLB). Se non tutte le variabili sono vincolate ad essere intere il problema si dice di programmazione lineare mista (PLM).

Generalmente i problemi PLI si esprimono in forma matriciale

$$(PL) \quad \min c^T x \tag{7.1}$$

$$\text{s.t.} \quad Ax \geq d, \tag{7.2}$$

$$x \geq 0 \quad \text{intere} \tag{7.3}$$

Per esprimere l'interezza delle variabili decisionali sotto forma di vincolo è possibile utilizzare per le variabili intere, una equazione nonlineare del tipo

$$\sin \pi x_j = 0 \tag{7.4}$$

e per le variabili binarie una equazione quadratica

$$x_j \cdot (x_j - 1) = x_j^2 - x_j = 0 \quad j = 1, \dots, n. \quad (7.5)$$

I problemi PLI, PLB e PLM sono quindi assimilabili a problemi nonlineare per la presenza dei vincoli di interezza delle variabili. In realtà però la nonlinearità è “concentrata” nella prescrizione di interezza delle variabili mentre tutto il resto del problema è lineare e come vedremo questo permette di mettere a punto algoritmi efficaci per la risoluzione di tali problemi.

## 7.1 Difficoltà dei problemi PLI

I problemi PLI appartengono ad una classe di problemi computazionalmente difficili nota come classe dei problemi NP-difficili (NP-hard). Per tali problemi non sono noti algoritmi in grado di trovare la soluzione in tempi efficienti, ossia in un tempo di calcolo la cui crescita è limitata da una funzione polinomiale della dimensione del problema. Per tali problemi sono però stati messi a punto algoritmi, principalmente basati sul metodo Branch-and-Bound che sarà illustrato nel seguito. Tali algoritmi negli ultimi anni sono stati oggetto di intensi sforzi di ricerca che ne hanno migliorate le prestazioni al punto che gli attuali codici commerciali per la risoluzione di problemi PLI, quali ad esempio Cplex, Gurobi, Xpress, sono in grado di risolvere in molti casi problemi di grande dimensione in tempi contenuti e compatibili con le applicazioni. La difficoltà computazionale per la risoluzione di tali problemi rimane però un ostacolo insormontabile per problemi di grande scala o per il trattamento di alcuni vincoli che caratterizzano le applicazioni reali. In tali casi, rinunciando alla garanzia di ottimalità vengono messi a punto algoritmi euristici che permettono di determinare soluzioni ammissibili del problema, in generale di buona qualità ed in tempi di calcolo compatibili con le esigenze delle applicazioni. Nel seguito non esamineremo la definizione di algoritmi euristici e ci limiteremo all’illustrazione dei concetti di base della tecnica Branch-and-Bound che sarà esaminata nel Capitolo ??.

### 7.1.1 Proprietà dei problemi PLI

Al fine di risolvere i problemi PLI è di grande importanza considerare il problema PL associato, ottenuto rimuovendo il vincolo di interezza delle variabili. Tale problema è detto *rilassamento continuo* del problema PLI.

**Definizione 7.2 (Rilassamento continuo).** *Dato un problema PLI,  $P : z_P = \min\{c^T x, x \in \Re^n, Ax = d, x \geq 0\}$  intere} si dice rilassamento continuo il problema PL,  $CP : z_{CP} = \min\{c^T x, x \in \Re^n, Ax = d, x \geq 0\}$  ottenuto rimuovendo il vincolo di interezza.*

Il rilassamento continuo ha un ruolo fondamentale nella risoluzione dei problemi PLI perché la sua soluzione ottima è in stretta relazione con quella del problema PLI cui è associato. In particolare, vale il seguente

**Teorema 7.1.** *Dato un problema PLI  $P : z_P = \min\{c^T x, x \in \mathbb{R}^n, Ax = d, x \text{ intere}\}$  ed il rilassamento continuo associato,  $CP : z_{CP} = \min\{c^T x, x \in \mathbb{R}^n, Ax = d, x \geq 0\}$  e detti  $z_P$  e  $z_{CP}$  i valori delle rispettive soluzioni ottime, si ha che*

$$z_{CP} \leq z_P. \quad (7.6)$$

**Dim.** La regione ammissibile di  $CP$  contiene la regione ammissibile di  $P$  dato che è ottenuta da questa rimuovendo il vincolo di interezza e la funzione obiettivo dei due problemi è la stessa. Pertanto la soluzione ottima di  $CP$  non può essere peggiore di quella di  $P$ .  $\square$

La relazione tra problema PLI e rilassamento continuo è illustrata nella figura 7.1 in cui i vincoli  $Ax = d$  sono indicati dalle rette che contengono un insieme di punti interi che rappresentano le soluzioni ammissibili del problema PLI. In figura è indicata la soluzione ottima  $x_P$  del problema  $P$  e la soluzione ottima  $x_{CP}$  del suo rilassamento continuo. In figura è anche indicata la soluzione  $\lfloor x_{CP} \rfloor$ , ottenuta arrotondando le coordinate del rilassamento continuo che è possibile verificare sia una soluzione ammissibile ma non ottima. In altre parole il rilassamento continuo consente di calcolare una approssimazione (detta anche *bound*) del valore della soluzione ottima del problema PLI: nel caso di problemi di minimo sarà una approssimazione per difetto (*lower bound*) mentre nel caso di problemi di massimo l'approssimazione è per eccesso (*upper bound*).

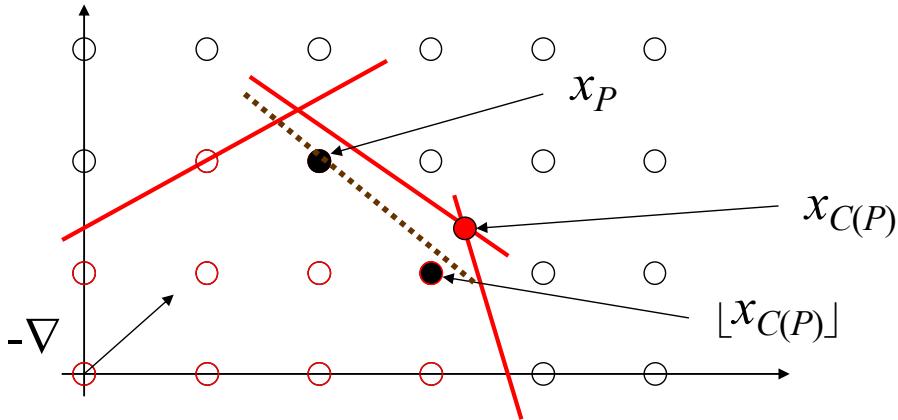


Figura 7.1: Rilassamento continuo di un problema PLI.

Vale inoltre il seguente

**Teorema 7.2.** *Dato un problema PLI  $P = \min\{c^T x, x \in \mathbb{R}^n, Ax = d, x \text{ intere}\}$  ed il rilassamento continuo associato,  $CP = \min\{c^T x, x \in \mathbb{R}^n, Ax = d, x \geq 0\}$  se la soluzione ottima  $x_{CP}$  è ammissibile per  $P$ , ossia ha coordinate intere, allora è la soluzione ottima anche di  $P$  e  $z_{CP} = z_P$ .*

**Dim.** Dal teorema 7.1 sappiamo che La soluzione ottima di  $CP$  è tale che  $z_{CP} \leq z_P$ . Tale soluzione è però anche soluzione ammissibile per  $P$  e quindi il

suo valore è maggiore o uguale a quello di  $z_P$ . Quindi si ha che  $z_P \leq z_{CP} \leq z_P$  e quindi i valori delle due soluzioni non possono che coincidere.  $\square$

## 7.2 Formulazioni equivalenti di PLI

A differenza dei problemi PL per cui, a meno dell'introduzione di vincoli ridondanti, la formulazione è univoca, per i problemi PLI la formulazione non è univoca. Infatti, la regione ammissibile di un problema PLI è costituita da un insieme  $X$  di punti a coordinate intere che generalmente viene definita “catturando” tali punti con un insieme di vincoli lineari come mostrato nella figura 7.1. Più precisamente dato un problema PLI  $P : z_P = \min\{c^T x, x \in X\}$  questo viene formulato mediante un insieme di vincoli lineari come  $P : z_P = \min\{c^T x, x \in \mathbb{R}^n, Ax = d, x \text{ intere}\}$ . Dato un insieme  $X$  esistono molti modi per “racchiudere” tale insieme con un insieme di vincoli lineari. Come mostrato in figura 7.2 dato un insieme  $X = \{(1, 1), (1, 2), (2, 1), (2, 2)\}$  formato da quattro punti interi che rappresentano la regione ammissibile di un problema di massimizzazione, questo può essere delimitato da diverse formulazioni equivalenti che utilizzano 4 disequazioni lineari:

$$1) \frac{1}{2} \leq x_1 \leq \frac{3}{2}, \quad \frac{1}{2} \leq x_2 \leq \frac{3}{2};$$

$$2) 1 \leq x_1 \leq \frac{5}{4}, \quad 1 \leq x_2 \leq \frac{5}{4};$$

$$3) 1 \leq x_1 \leq 2, \quad 1 \leq x_2 \leq 2.$$

Come è possibile verificare tali formulazioni, così come altre che utilizzano un numero anche maggiore di vincoli, sono tutte equivalenti nel senso che contengono tutti solo i punti interi appartenenti all'insieme  $X$ . È però possibile anche verificare che i rilassamenti continui, la cui soluzione ottima è indicata in figura 7.2 non sono tra loro equivalenti e rappresentano approssimazioni via via più accurate della soluzione ottima del problema  $P$  fino ad individuarne la soluzione ottima stessa.

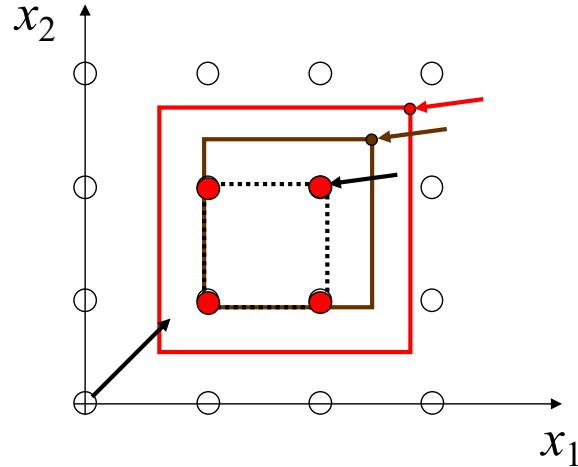


Figura 7.2: Esempi di formulazioni equivalenti di un problema PLI.

Quanto fin qui discusso porta ad individuare un criterio che consente di confrontare fra di loro formulazioni equivalenti al fine di determinare se una sia migliore, o come spesso viene definita più “tight” (attillata) di un’altra ad essa equivalente.

**Definizione 7.3 (Dominanza tra formulazioni).** *Dato un problema PLI e due formulazioni equivalenti  $Q^1 = \{A^1x = d^1, x \geq 0\}$  e  $Q^2 = \{A^2x = d^2, x \geq 0\}$  del problema, la formulazione  $Q^1$  domina la formulazione  $Q^2$  se  $Q^1 \subset Q^2$ . In tal caso se il problema è in forma di minimo si ha che tra le soluzioni ottime dei due rilassamenti continui vale la seguente relazione*

$$z_{CP^1} \geq z_{CP^2},$$

ossia la formulazione  $\varphi_1$  produce una stima per difetto (lower bound) migliore rispetto all’altra.

La relazione di dominanza tra le formulazioni appena definita è illustrata nella figura 7.3. A titolo di esempio si consideri il seguente problema KP01 con uno zaino di capacità  $W = 6$  e tre oggetti di peso  $w = \{4, 3, 2\}$ . La regione ammissibile del problema è costituita dall’insieme

$$X = \{(0, 0, 0), (1, 0, 0), (1, 0, 1), (0, 1, 0), (0, 1, 1), (0, 0, 1)\}.$$

Consideriamo due formulazioni valide per tale problema, la prima delle quali è quella naturale con il solo vincolo di capacità

$$Q^1 = \{x \in \mathbb{R}^3 : 0 \leq x \leq 1, 4x_1 + 3x_2 + 2x_3 \leq 6\},$$

mentre la seconda include un vincolo che sancisce l'incompatibilità tra i primi due oggetti il cui peso complessivo supera la capacità

$$Q^2 = \{x \in \mathbb{R}^3 : 0 \leq x \leq 1, 4x_1 + 3x_2 + 2x_3 \leq 6, x_1 + x_2 \leq 1\}.$$

È facile verificare che  $Q^2 \subset Q^1$  e pertanto la formulazione  $Q^2$  domina la formulazione  $Q^1$ .

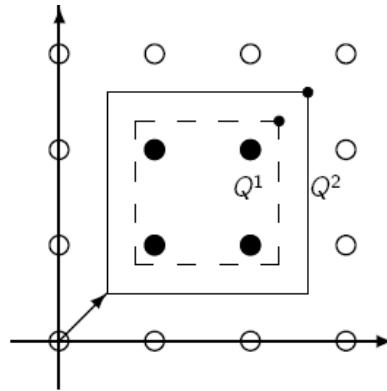


Figura 7.3: Relazione di dominanza tra formulazioni equivalenti di un problema PLI.

Possiamo ora domandarci se dato un problema PLI esista per esso una formulazione “ideale” che risulti migliore delle altre. A tal fine introduciamo la seguente

**Definizione 7.4 (Guscio convesso (Convex hull).** *Dato un insieme  $S \subseteq \mathbb{R}^n$  si dice guscio convesso (convex hull) il più piccolo insieme convesso  $\text{conv}(S)$  che contiene  $S$ .*

Come illustrato in figura 7.4 vale il seguente

**Teorema 7.3.** *Se  $S$  è un insieme di punti a coordinate intere e finite<sup>1</sup>,  $\text{conv}(S)$  è un politopo i cui vertici sono tutti punti a coordinate intere.*

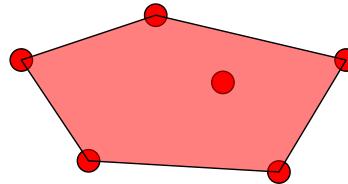


Figura 7.4: Guscio convesso di un insieme di punti.

---

<sup>1</sup>Il teorema è generalizzabile anche al caso di insiemi  $S$  illimitati

Si noti che la conseguenza del teorema 7.3 è che dato un problema PLI e la sua formulazione che corrisponde al guscio convesso della regione ammissibile, dato che tutti i vertici del politopo risultante hanno coordinate intere sarebbe possibile risolvere il problema intero come problema continuo utilizzando l'algoritmo del simplex. Purtroppo in generale la formulazione del guscio convesso di un insieme di punti richiede un numero di disequazioni molto elevato e che cresce esponenzialmente con la dimensione del problema da formulare. Quindi non è in generale possibile utilizzare direttamente il guscio convesso per formulare e risolvere il problema PLI. In realtà però i metodi risolutivi esistenti utilizzano almeno in parte questi concetti per arrivare alla soluzione del problema PLI in un tempo di calcolo accettabile. Alcuni di questi metodi quali il metodo dei piani di taglio (cutting plane) e le sue generalizzazioni note come branch-and-cut e branch-and-cut-and-price vanno al di là dello scopo di queste lezioni in cui ci limiteremo ad illustrare alcuni metodi fondamentali quali l'algoritmo branch-and-bound.



## Capitolo 8

# Modelli di Programmazione Lineare Intera NEW: ((Ver. 6.0))

### 8.1 Introduzione

Un modello generale di PL mista (PLM) ha la forma:

$$(P) \quad z(P) = \min c^T x$$
$$\text{s.t.} \quad \begin{array}{ll} a_i^T x = b_i & i = 1, \dots, M \\ a_i^T x \geq b_i & i = 1, \dots, \overline{M} \\ x_j \geq 0 & j \in N \\ x_j \text{ intere} & j \in \overline{N} \end{array}$$

e le sue componenti sono:

- Coefficienti:
  - $A = [m, n] = \{a_{ij}\}$ , matrice tecnologica di dimensione  $(m \times n)$ ;
  - $b = [m]$ , vettore dei termini noti di dimensione  $m$ ;
  - $c = [n]$ , vettore dei costi di dimensione  $n$ ;
- Variabili decisionali:
  - $x_j = [n]$ , vettore di dimensione  $n$ .

Le variabili decisionali possono assumere valori sia continui sia discreti (interi), ossia  $x_j \in N, x_j \in \overline{N}$ . Se l'insieme  $N$  è vuoto  $P$  è un problema di programmazione

lineare intera (PLI), se invece risulta vuoto  $\bar{N}$ , allora il problema  $P$  è un problema di programmazione lineare (continua) (PL). Se entrambi gli insiemi non sono vuoti allora il problema è di programmazione lineare mista (PLM)

I vincoli e la funzione obiettivo sono tutti definiti da funzioni lineari nelle variabili decisionali, ossia combinazioni lineari delle variabili decisionali  $x_j$  e dei coefficienti  $a_{ij}$  e  $c_j$ . In caso contrario il modello è di programmazione non lineare (PNL). Tutti i coefficienti del modello sono valori deterministici noti, in caso contrario il modello è di programmazione stocastica (PS).

## 8.2 Indagine di Mercato

Un'azienda di marketing deve effettuare una indagine telefonica di mercato intervistando utenti di quattro categorie, A, B, C e D. Le interviste possono essere effettuate al mattino o alla sera, al costo di 1€ ed 1.5€ ciascuna, rispettivamente. Per ciascuna categoria è stato definito il numero minimo di utenti che è necessario contattare ed è nota la probabilità che risponda ad una chiamata al mattino o alla sera, come riportato nella seguente tabella.

Categoria	A	B	C	D	-
n. Contatti	150	110	120	100	
mattino	30%	10%	10%	10%	40%
sera	30%	30%	30%	10%	10%

Si noti che il 40% delle telefonate al mattino e il 10% di quelle alla sera non contattano alcun utente.

Si desidera determinare quante telefonate fare al mattino e quante alla sera in modo tale che il numero di contatti di ciascuna categoria sia raggiunto e sia minimizzato il costo complessivo.

**Modello PLI** Per modellare il problema, bisogna per prima cosa identificare le variabili decisionali. In questo caso le decisioni da prendere riguardano il numero di telefonate da eseguire in ciascuna fascia oraria. Saranno quindi necessarie due variabili,  $x_1$  e  $x_2$  che rappresentano il numero di telefonate al mattino e alla sera, rispettivamente e che necessariamente dovranno assumere valori interi.

La funzione obiettivo è costituita dalla somma dei costi delle telefonate, ossia  $x_1 + 1.5x_2$ , e dovrà essere minimizzata. I vincoli del problema richiedono che per ciascuna categoria il numero di utenti raggiunti sia almeno pari al valore richiesto per tale categoria.

$$\min \quad x_1 + 1.5 x_2 \quad (8.1)$$

$$\text{s.t.} \quad 0.3 x_1 + 0.3 x_2 \geq 150 \quad (8.2)$$

$$0.1 x_1 + 0.2 x_2 \geq 110 \quad (8.3)$$

$$0.1 x_1 + 0.3 x_2 \geq 120 \quad (8.4)$$

$$0.1 x_1 + 0.1 x_2 \geq 100 \quad (8.5)$$

$$x_1, \quad x_2 \geq 0 \quad \text{intere} \quad (8.6)$$

### 8.3 Noleggio di Dispositivi

Un'azienda deve noleggiare dei dispositivi per coprire la domanda irregolare del reparto di produzione. Il fabbisogno, in numero di dispositivi, per i successivi sei mesi è riportato nella tabella seguente.

Mese	Gen	Feb	Mar	Apr	Mag	Giu
Fabbisogno	9	5	7	9	10	5

I dispositivi possono essere noleggiati per uno, due o tre mesi al seguente costo

- un mese a 400€
- due mesi a 700€ (ossia 350€ al mese)
- tre mesi a 900€ (ossia 300€ al mese).

Si desidera determinare il piano dei noleggi che minimizza il costo complessivo necessario a soddisfare il fabbisogno minimo. Si osservi che, se conveniente alcuni dispositivi possono essere lasciati inutilizzati durante uno o più mesi. Inoltre, nessun dispositivo sarà necessario dopo i sei mesi considerati.

**Modello PLI** Le decisioni riguardano il numero di dispositivi da noleggiare in ciascun mese e la durata del noleggio scelta. Sarà quindi necessario utilizzare variabili decisionali associate a ciascun mese e durata del noleggio. Per una migliore leggibilità del modello utilizziamo tre gruppi di sei variabili decisionali intere ciascuno e definiti come segue

- $x_i$  = numero di dispositivi noleggiati per un mese nel mese  $i = 1, \dots, 6$ .
- $y_i$  = numero di dispositivi noleggiati per due mesi nel mese  $i = 1, \dots, 6$ .
- $w_i$  = numero di dispositivi noleggiati per tre mesi nel mese  $i = 1, \dots, 6$ .

La funzione obiettivo è costituita dalla somma dei dei costi dei noleggi e dovrà essere minimizzata. I vincoli del problema richiedono che per ciascuna

mese sia disponibile, perché noleggiati nel mese stesso o nei mesi precedenti, un numero di dispositivi almeno pari al fabbisogno di quel mese.

$$\min \quad 300 \sum_{i=1}^6 x_i + 700 \sum_{i=1}^6 y_i + 900 \sum_{i=1}^6 w_i \quad (8.7)$$

$$\text{s.t.} \quad x_1 + y_1 + w_1 \geq 9 \quad (8.8)$$

$$x_2 + y_2 + w_2 + y_1 + w_1 \geq 5 \quad (8.9)$$

$$x_3 + y_3 + w_3 + y_2 + w_2 + w_1 \geq 7 \quad (8.10)$$

$$x_4 + y_4 + w_4 + y_3 + w_3 + w_2 \geq 9 \quad (8.11)$$

$$x_5 + y_5 + w_5 + y_4 + w_4 + w_3 \geq 10 \quad (8.12)$$

$$x_6 + y_6 + w_6 + y_5 + w_5 + w_4 \geq 5 \quad (8.13)$$

$$x_i, y_i, w_i \geq 0, \text{ intere} \quad (i = 1, \dots, 6) \quad (8.14)$$

Si osservi che nel mese di maggio e giugno non è conveniente noleggiare dispositivi per un periodo che vada oltre il mese di giugno pertanto le variabili  $w_5$ ,  $y_6$  e  $w_6$  possono essere eliminate dal modello dato che avranno valore uguale a zero nella soluzione ottima.

## 8.4 Campagna pubblicitaria

Un'agenzia pubblicitaria deve pubblicizzare una nuova iniziativa mediante degli annunci, disponendo di un importo complessivo pari a 150.000€. Per la campagna possono essere utilizzati due tipi di canali: annunci sui giornali e spot presso televisioni locali. L'agenzia prevede di fare al massimo 30 annunci sui giornali, del costo di 1.000€ ciascuno, e 15 spot televisivi del costo di 10.000€ ciascuno. Il numero di nuovi utenti che sono raggiunti da un annuncio o da uno spot dipende in maniera decrescente dal numero di questi come descritto nelle seguenti tabelle ed illustrato, per gli annunci sui giornali in figura 8.1:

Giornali	
N. di annunci	Nuovi utenti raggiunti
1-10	900
11-20	600
21-30	300

Televisione	
N. di spot	Nuovi utenti raggiunti
1-5	10.000
6-10	5.000
11-15	2.000

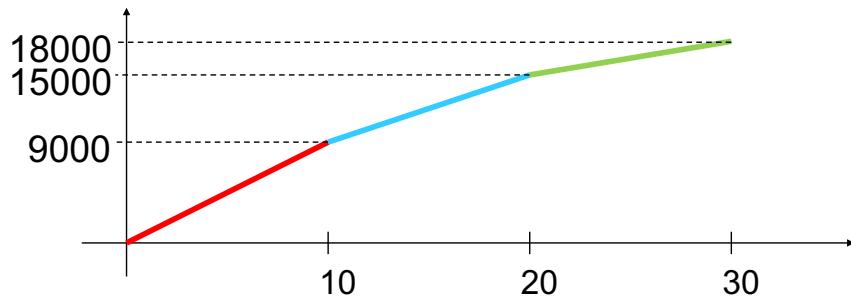


Figura 8.1: Numero di utenti raggiunti in dipendenza del numero di annunci sui giornali fatti.

Si desidera determinare il mix di annunci che massimizza il numero di utenti raggiunti nel rispetto del budget disponibile.

**Modello PLI** Le decisioni riguardano il numero di annunci e di spot da realizzare. L'andamento non lineare, descritto da una funzione lineare a tratti come illustrato in figura 8.1, del numero di utenti raggiunti richiede però di essere modellato adeguatamente. In generale la modellazione di comportamenti non lineari mediante funzioni lineari è un'attività complessa ma in questo caso si può osservare che l'andamento della pendenza della curva da modellare, ossia il numero di nuovi utenti raggiunti, è non-crescente al crescere del valore delle variabili decisionali. Un possibile modo per modellare questo comportamento è quello di utilizzare per ogni tipo di annuncio una diversa variabile decisionale associata ad ogni intervallo della funzione lineare a tratti che rappresenta il numero di annunci effettuati in tale intervallo. Pertanto utilizzeremo le seguenti variabili decisionali:

- $x_1, x_2, x_3$ : numero di annunci sui giornali negli intervalli 0-10, 11-20 e 21-30, rispettivamente;
- $y_1, y_2, y_3$ : numero di spot televisivi negli intervalli 0-5, 6-10 e 11-15, rispettivamente.

Il numero totale di annunci dei due tipi fatti sarà ovviamente dato dalla somma delle variabili associate ai rispettivi intervalli e la funzione obiettivo utilizzerà come coefficiente per ciascuna variabile il numero di utenti nuovi raggiunti per annuncio in tale intervallo. I vincoli del problema sono unicamente relativi al limite di spesa complessiva per gli annunci ed impongono che le variabili decisionali non eccedano il numero massimo di annunci associati al relativo intervallo.

$$\max \quad 9x_1 + 6x_2 + 3x_3 + 100y_1 + 50y_2 + 20y_3 \quad (8.15)$$

$$\text{s.t.} \quad x_1 + x_2 + x_3 + 10y_1 + 10y_2 + 10y_3 \leq 150 \quad (8.16)$$

$$x_1 \leq 10 \quad (8.17)$$

$$x_2 \leq 10 \quad (8.18)$$

$$x_3 \leq 10 \quad (8.19)$$

$$y_1 \leq 5 \quad (8.20)$$

$$y_2 \leq 5 \quad (8.21)$$

$$y_3 \leq 5 \quad (8.22)$$

$$x_1, x_2, x_3, y_1, y_2, y_3 \geq 0, \text{ intere} \quad (8.23)$$

in cui la funzione obiettivo è espressa in centinaia di utenti raggiunti.

Si noti che non è necessario imporre che una variabile decisionale associata ad un intervallo, ad esempio  $x_2$  possa essere maggiore di zero solo se la variabile associata all'intervallo precedente sia completamente saturata, ossia solo se  $x_1 = 10$ . Infatti, non sarebbe conveniente utilizzare una unità di  $x_2$  al posto di una di  $x_1$  dato che il profitto  $x_2$  è inferiore a quello di  $x_1$ .

## 8.5 Turnazione del Personale

Un ospedale deve definire i turni degli infermieri per un reparto il cui fabbisogno è variabile nei diversi giorni della settimana come riportato nella seguente tabella.

Giorno	Lu	Ma	Me	Gi	Ve	Sa	Do
Fabbisogno	22	18	13	14	15	18	25

Per la copertura dei turni il personale può iniziare il proprio turno settimanale in ciascun giorno della settimana e lavora per cinque giorni consecutivi e riposa i due giorni successivi. Ad esempio se un infermiere inizia il proprio turno di martedì lavorerà fino a sabato e poi riposa la domenica ed il lunedì.

Si desidera minimizzare il personale necessario alla copertura del fabbisogno.

**Modello PLI** Le decisioni riguardano il numero di infermieri che iniziano il proprio turno in uno specifico giorno della settimana. Utilizzeremo pertanto sette variabili decisionali  $x_1, \dots, x_7$  che rappresentano il numero di infermieri che iniziano il proprio turno di lavoro nei giorni da lunedì a domenica, rispettivamente.

La funzione obiettivo richiede di minimizzare il numero complessivo di infermieri e sarà quindi uguale alla somma delle variabili decisionali.

I vincoli sul fabbisogno sono uno per ogni giorno della settimana ed impongono che gli infermieri attivi in quel giorno siano almeno pari al fabbisogno stesso. Ad esempio, il fabbisogno di martedì è coperto dagli infermieri che iniziano il turno in tale giorno ma anche da quelli che iniziano il turno nei quattro giorni precedenti.

$$\max \quad x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 \quad (8.24)$$

$$\text{s.t.} \quad x_1 + x_4 + x_5 + x_6 + x_7 \geq 22 \quad (8.25)$$

$$x_1 + x_2 + x_5 + x_6 + x_7 \geq 18 \quad (8.26)$$

$$x_1 + x_2 + x_3 + x_6 + x_7 \geq 13 \quad (8.27)$$

$$x_1 + x_2 + x_3 + x_4 + x_7 \geq 14 \quad (8.28)$$

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_7 \geq 15 \quad (8.29)$$

$$+ x_2 + x_3 + x_4 + x_5 + x_6 \geq 18 \quad (8.30)$$

$$x_3 + x_4 + x_5 + x_6 + x_7 \geq 25 \quad (8.31)$$

$$x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geq 0, \text{ intere} \quad (8.32)$$

**Varianti** Nei problemi reali di turnazione del personale possono essere presenti numerose ulteriori caratteristiche che influenzano la costruzione del modello. Ad esempio, possono essere presenti turni di tipo diverso rispetto al tipo di turno che è stato considerato nell'esercizio (lavora cinque giorni e risposa i due giorni seguenti) e per ciascuno di questi tipi di turno possono essere presenti un numero massimo di addetti che possono utilizzarli.

Il modello descritto in questa sezione considera il personale in modo indistinto ossia considera unicamente il numero di addetti che dovranno seguire un dato turno. In pratica sarà quindi necessario assegnare i turni al personale vero che potrebbe avere propensioni diverse rispetto ai diversi tipi di turno assegnati. Ad esempio alcuni addetti potrebbero preferire lavorare nei fine settimana ed altri no. A tal proposito è possibile provvedere all'assegnazione dei turni al personale reale formulandolo come problema dell'assegnamento, definito nella sezione 8.12, in cui per ogni addetto il costo di assegnamento ad un incarico è modellato come una preferenza da 1 (incarico preferito) a 7 (incarico meno preferito). Risolvendo il problema dell'assegnamento si ottengono in tal modo le assegnazioni dei turni al personale che minimizzano il costo di assegnazione massimizzano le preferenze espresse dagli addetti.

## 8.6 Il problema dello zaino (Knapsack Problem)

**Definizione del problema** Sono dati

- un insieme  $N$  di  $n$  oggetti, il  $j$ -esimo dei quali caratterizzato da un *profitto* positivo  $p_j$  e da un *peso* positivo  $w_j$ ;
- un contenitore (*zaino* o *knapsack*) di capacità  $C$ .

Il problema dello zaino richiede di selezionare un sottoinsieme di oggetti tale che

- la somma dei pesi degli oggetti selezionati non ecceda la capacità dello zaino;
- la somma dei profitti degli oggetti selezionati sia massima.

**Modello PLI** Per modellare il problema, bisogna per prima cosa identificare le variabili decisionali. Un possibile modello per il problema dello zaino introduce, per ciascun oggetto  $j$ , una variable binaria  $x_j$  con il seguente significato:

$$x_j = \begin{cases} 1 & \text{se l'oggetto } j \text{ è selezionato} \\ 0 & \text{altrimenti} \end{cases} \quad (j = 1, \dots, n) \quad (8.33)$$

Si può osservare che, con questa scelta delle variabili decisionali, il profitto associato a ciascun oggetto  $j$  è esprimibile come

$$p_j x_j = \begin{cases} p_j & \text{se l'oggetto } j \text{ è selezionato} \\ 0 & \text{altrimenti} \end{cases} \quad (j = 1, \dots, n)$$

Dato che la funzione obiettivo è costituita dalla somma dei profitti degli oggetti selezionati, la scelta delle variabili dovrà cercare di massimizzare la quantità  $\sum_{j=1}^n p_j x_j$ .

Analogamente, il peso occupato da ciascun oggetto  $j$  nella soluzione è esprimibile come  $w_j x_j$  ed il vincolo di capacità dello zaino imporrà che le variabili soddisfino la condizione  $\sum_{j=1}^n w_j x_j \leq C$ .

Imponendo infine che ciascuna variabile  $x_j$  sia binaria, si ottiene il seguente modello di Programmazione Lineare Intera

$$(KP01) \quad \max \sum_{j=1}^n p_j x_j \quad (8.34)$$

$$\text{s.t.} \quad \sum_{j=1}^n w_j x_j \leq C \quad (8.35)$$

$$x_j \in \{0, 1\} \quad j = 1, \dots, n. \quad (8.36)$$

Il modello matematico descritto sopra coinvolge  $n$  variabili (una per ogni oggetto) ed un solo vincolo, oltre a quelli che descrivono il dominio delle variabili. Si noti che quello proposto è solo un *possibile* modello per il problema dello zaino. In altre parole, esistono modelli alternativi che descrivono esattamente lo stesso problema, ma con variabili e vincoli diversi. La dimensione di questi modelli alternativi, in termini di numero di variabili e vincoli, è in generale diversa dalla dimensione del modello sopra descritto. Più in generale, per qualunque problema di ottimizzazione possono esistere modelli alternativi, che utilizzano variabili e vincoli diversi. Inoltre, la dimensione del problema (numero di oggetti, nel caso del knapsack problem) non definisce univocamente la dimensione del modello matematico che lo descrive.

**Esempio numerico** Consideriamo il seguente esempio numerico. Sono dati  $n = 7$  oggetti ed uno zaino con capacità  $C = 100$ . I profitti e pesi degli oggetti sono riportati dai seguenti due vettori ( $p$ ) e ( $w$ ), ciascuno contenente 7 valori:

$$(p) = (40, 62, 14, 20, 98, 5, 73)$$

$$(w) = (18, 33, 12, 24, 21, 8, 28)$$

L'insieme di valori numerici per un problema di ottimizzazione costituisce una *istanza* del problema. Per l'istanza numerica sopra specificata, il corrispondente modello di PLI è il seguente

$$\begin{aligned} \max \quad & 40x_1 + 62x_2 + 14x_3 + 20x_4 + 98x_5 + 5x_6 + 73x_7 \\ & 18x_1 + 33x_2 + 12x_3 + 24x_4 + 21x_5 + 8x_6 + 28x_7 \leq 100 \\ & x_1, x_2, x_3, x_4, x_5, x_6, x_7 \in \{0, 1\} \end{aligned}$$

Risolvere questa istanza all'ottimo tramite il modello matematico sopra riportato significa determinare il valore (0 oppure 1) da attribuire a ciascuna variabile decisionale  $x_1, x_2, \dots, x_7$  in modo da ottimizzare la funzione obiettivo, rispettando tutti i vincoli presenti.

**Assunzioni** Senza perdita di generalità possiamo fare le seguenti assunzioni sui dati di ingresso di un problema knapsack:

1. La somma dei pesi di tutti gli oggetti eccede la capacità dello zaino, ovvero

$$\sum_{j=1}^n w_j > W$$

Infatti, se questa assunzione fosse violata, sarebbe possibile selezionare tutti gli oggetti in soluzione, e non sarebbe quindi richiesta alcuna ottimizzazione.

2. Ciascun oggetto può essere inserito da solo in soluzione, ovvero

$$w_j \leq W \quad \forall j = 1, \dots, n$$

Infatti, immaginiamo che esista un oggetto  $k$  il cui peso supera la capacità dello zaino, cioè tale che  $w_k > C$ . In nessuna soluzione ammissibile del problema è possibile inserire l'oggetto  $k$ . Quindi, la soluzione ottima dell'istanza sarà ottenibile risolvendo una nuova istanza di knapsack nella quale l'oggetto  $k$  è stato rimosso.

## Applicazioni

### 8.6.1 Varianti del problema

**Problema Knapsack con vincolo di uguaglianza** In questa variante del problema, bisogna selezionare un sottoinsieme di oggetti che utilizzi *tutta* la capacità a disposizione. Per modellare questo problema è possibile utilizzare le stesse variabili decisionali del problema knapsack, la stessa funzione obiettivo, e rimpiazzare il vincolo di capacità (8.35) con il seguente vincolo

$$\sum_{j=1}^n w_j x_j = C \tag{8.37}$$

Tale vincolo impone che la somma dei pesi degli oggetti selezionati sia *esattamente* uguale al valore della capacità dello zaino.

**Problema Knapsack in forma di minimo** Dati  $n$  oggetti, il  $j$ -esimo dei quali caratterizzato da un *costo* positivo  $c_j$  e da un *peso* positivo  $w_j$ , ed peso di riferimento  $B$ , il problema dello zaino in forma di minimo richiede di selezionare un sottoinsieme di oggetti tale che

- la somma dei pesi degli oggetti selezionati sia non inferiore a  $B$ ;
- la somma dei costi degli oggetti selezionati sia minima.

Questa variante del problema è speculare rispetto alla versione base del problema knapsack. Mentre nel caso precedente l'obiettivo era quello di selezionare “tanti” oggetti (in modo da massimizzare il profitto complessivo) senza eccedere un valore massimo di peso complessivo, in questo problema vogliamo selezionare “pochi” oggetti (in modo da minimizzare il costo complessivo), garantendo di occupare un valore minimo di peso complessivo. Nonostante questa differenza, il problema può essere modellato utilizzando ancora le variabili decisionali  $x_j$  descritte nella (8.33). Con questa scelta delle variabili, il corrispondente modello matematico risulta essere il seguente

$$(KP - \min) \quad \min \sum_{j=1}^n c_j x_j \quad (8.38)$$

$$\text{s.t.} \quad \sum_{j=1}^n w_j x_j \geq B \quad (8.39)$$

$$x_j \in \{0, 1\} \quad j = 1, \dots, n. \quad (8.40)$$

**Problema del Knapsack Multiple-Choice** Consideriamo ora una variante del problema knapsack nella quale gli oggetti hanno un secondo attributo che li caratterizza, ad esempio la dimensione, e che sia possibile classificare ciascun oggetto come grande, medio o piccolo. In base a questo secondo attributo, possiamo partizionare l'insieme di oggetti in 3 diversi sottoinsiemi disgiunti

$$N = N_G \cup N_M \cup N_P$$

dove  $N_G$ ,  $N_M$  e  $N_P$  rappresentano gli insiemi di oggetti grandi, medi e piccoli, rispettivamente.

Nel problema del Knapsack Multiple-Choice si vuole determinare una soluzione del problema knapsack che soddisfi questo vincolo aggiuntivo: è possibile selezionare al più un oggetto per ciascuna categoria. Questo vuol dire che, nel nostro esempio, qualunque soluzione ammissibile deve contenere al più un oggetto grande, al più un oggetto medio e al più un oggetto piccolo. Per modellare questo problema, bisognerà aggiungere queste condizioni al modello del problema knapsack (che non richiedeva tale requisito). Utilizzando ancora le variabili  $x_j$  descritte nella (8.33), i vincoli aggiuntivi possono essere modellati nel seguente modo

$$\sum_{j \in N_G} x_j \leq 1 \quad \sum_{j \in N_M} x_j \leq 1 \quad \sum_{j \in N_P} x_j \leq 1$$

Ciascuna sommatoria sui riferisce ad un particolare sottoinsieme di oggetti e coinvolge le variabili associate a tali oggetti. Dato che ciascuna variabile vale 1 (se il corrispondente oggetto è selezionato) oppure 0 (altrimenti), la somma delle variabili che compare nella parte sinistra del vincolo indica il numero di variabili che assumono il valore 1, ossia il numero di oggetti selezionati nel sottoinsieme. Ciascun vincolo impone quindi che questo numero sia al più pari a 1 per ciascun sottoinsieme di oggetti.

Più in generale, nel problema del Knapsack Multiple-Choice gli oggetti sono partizionati in  $k$  sottoinsiemi:  $N = N_1 \cup N_2 \dots \cup N_k$ , ed esiste un vincolo aggiuntivo che impone di selezionare al massimo un oggetto per ogni sottoinsieme  $N_\ell$  ( $\ell = 1, \dots, k$ ). Si osservi che il numero  $k$  di sottoinsiemi e la composizione di ciascun sottoinsieme sono un dato di input, noto a priori, e non dipende quindi dalle scelte da effettuare in fase di ottimizzazione. Per modellare il problema, possiamo utilizzare ancora le variabili  $x_j$  ed aggiungere al modello (8.34)–(8.36) i seguenti vincoli

$$\sum_{j \in N_\ell} x_j \leq 1 \quad \ell = 1, \dots, k \quad (8.41)$$

Il termine di sinistra di ciascun vincolo  $\ell$ -esimo corrisponde al numero di oggetti appartenenti al sottoinsieme  $\ell$  che sono selezionati in soluzione (cioè per i quali la variabile associata vale 1). Il vincolo impone che tale numero sia al massimo pari a 1.

Consideriamo ancora l'esempio numerico con  $n = 7$  oggetti, uno zaino con capacità  $C = 100$  ed i seguenti vettori di profitti e pesi:

$$(p) = (40, 62, 14, 20, 98, 5, 73) \quad (w) = (18, 33, 12, 24, 21, 8, 28)$$

Gli oggetti sono partizionati nei seguenti  $k = 3$  sottoinsiemi

$$N_1 = \{1, 2, 5\}, \quad N_2 = \{3, 6\}, \quad N_3 = \{4, 7\}$$

Il corrispondente modello di PLI per questa istanza è il seguente

$$\begin{array}{rclclclclclcl} \max & 40x_1 & +62x_2 & +14x_3 & +20x_4 & +98x_5 & +5x_6 & +73x_7 \\ & 18x_1 & +33x_2 & +12x_3 & +24x_4 & +21x_5 & +8x_6 & +28x_7 & \leq & 100 \\ & x_1 & + & x_2 & & & + & x_5 & \leq & 1 \\ & & & & x_3 & & & + & x_6 & \leq & 1 \\ & & & & & x_4 & & & + & x_7 & \leq & 1 \\ x_1 & , & x_2 & , & x_3 & , & x_4 & , & x_5 & , & x_6 & , & x_7 & \in \{0, 1\} \end{array}$$

**Problema del Knapsack Bounded** Consideriamo la variante del problema knapsack nella quale ogni oggetto  $j$  è disponibile in  $u_j$  esemplari identici; ciascun esemplare selezionato porta ad un profitto  $p_j$  ed occupa un peso  $w_j$ .

In questo problema, la soluzione non deve solo indicare il sottoinsieme di oggetti da selezionare, ma anche il numero di esemplari di ciascun oggetto che conviene selezionare. Per questo motivo, non è possibile scrivere un modello matematico di Programmazione Lineare Intera utilizzando solo  $n$  variabili binarie.

Per definire un modello corretto, introduciamo  $n$  variabili decisionali a valori interi

$$x_j = \text{numero di esemplari dell'oggetto } j \text{ in soluzione} \quad (j = 1, \dots, n) \quad (8.42)$$

Con questa scelta delle variabili decisionali, il profitto ottenuto selezionando  $x_j$  esemplari dell'oggetto  $j$  è pari a  $p_j x_j$  ed il corrispondente peso occupato è  $w_j x_j$ . Il modello matematico risultante è il seguente

$$(KP - b) \quad \max \sum_{j=1}^n p_j x_j \quad (8.43)$$

$$\text{s.t.} \quad \sum_{j=1}^n w_j x_j \leq C \quad (8.44)$$

$$x_j \leq u_j \quad j = 1, \dots, n \quad (8.45)$$

$$x_j \geq 0 \quad \text{intera} \quad j = 1, \dots, n \quad (8.46)$$

Consideriamo una istanza con  $n = 3$  oggetti, uno zaino con capacità  $C = 100$  ed i seguenti vettori di profitti, pesi e numero di esemplari:

$$(p) = (40, 62, 14) \quad (w) = (18, 33, 12) \quad (u) = (4, 2, 3)$$

Il corrispondente modello di PLI per questa istanza è il seguente

$$\begin{aligned} \max \quad & 40x_1 + 62x_2 + 14x_3 \\ \text{s.t.} \quad & 18x_1 + 33x_2 + 12x_3 \leq 100 \\ & x_1 \leq 4 \\ & x_2 \leq 2 \\ & x_3 \leq 3 \\ & x_1, x_2, x_3 \geq 0 \quad \text{intera} \end{aligned}$$

Si osservi che il KP01 è un caso particolare del knapsack bounded che si verifica quando  $u_j = 1$  per ogni  $j = 1, \dots, n$ . In altre parole, un qualunque strumento di soluzione (modello PLI, algoritmo, ...) per il knapsack bounded può essere utilizzato per risolvere una qualunque istanza di KP01 (cioè, se sappiamo risolvere knapsack bounded siamo in grado di risolvere il K01). È però vero anche il contrario (cioè se sappiamo risolvere KP01 siamo in grado di risolvere il knapsack bounded), ma questo richiede uno sforzo supplementare. Per ricondurre una istanza di knapsack bounded ad una istanza di KP01, bisogna introdurre, per ciascun oggetto  $j$ , un numero  $u_j$  di esemplari identici, tutti con lo stesso profitto e peso. Per l'esempio numerico descritto sopra, la corrispondente istanza di KP01 è definita nel seguente modo:

- 9 oggetti
- profitti  $(p) = (40, 40, 40, 40, 62, 52, 14, 14, 14)$
- pesi  $(w) = (18, 18, 18, 18, 33, 33, 12, 12, 12)$
- capacità pari a 100

**Problema del Knapsack Unbounded** In questa variante del problema, ciascun oggetto  $j$  è disponibile in un numero infinito di esemplari identici.

Per modellare il problema, si possono introdurre le variabili  $x_j$  definite in (8.42) e modeificare il modello del knapsack bounded rimuovendo il vincolo (8.45). Il modello risultante è il seguente

$$(KP - u) \quad \max \sum_{j=1}^n p_j x_j \quad (8.47)$$

$$\text{s.t.} \quad \sum_{j=1}^n w_j x_j \leq C \quad (8.48)$$

$$x_j \geq 0 \quad \text{intera} \quad j = 1, \dots, n \quad (8.49)$$

Si osservi che, anche se il numero di esemplari disponibili per ciascun oggetto  $j$  è illimitato, nella pratica esiste sempre un upper bound sul numero massimo di esemplari da considerare. Tale numero deriva dal fatto che la capacità del knapsack è finita e quindi, per ciascun oggetto  $j$ , il numero di esemplari che possono essere inseriti in qualunque soluzione ammissibile non può superare  $\lfloor \frac{C}{w_j} \rfloor$ . In altre parole, qualunque istanza di knapsack unbounded può essere trasformata in una istanza di knapsack bounded che ha la stessa soluzione ottima.

**Problema del Subset Sum** Questo problema è un caso particolare di KP01 nel quale ogni oggetto ha un profitto che coincide con il suo peso, ovvero  $p_j = w_j, \forall j = 1, \dots, n$ . Il modello matematico risultante è quindi il seguente

$$(SSP) \quad \max \sum_{j=1}^n w_j x_j \quad (8.50)$$

$$\text{s.t.} \quad \sum_{j=1}^n w_j x_j \leq C \quad (8.51)$$

$$x_j \in \{0, 1\} \quad j = 1, \dots, n. \quad (8.52)$$

Questo problema modella situazioni nelle quali è dato un insieme  $N$  di numeri positivi  $w_j$  ( $j \in N$ ) che deve essere partizionato in due sottoinsiemi il più bilanciati possibile. A tal fine si indichi con  $C = \frac{\sum_{j \in N} w_j}{2}$  il valore ideale di ciascun sottoinsieme, ed si assegni a ciascun numero  $j$  una variabile  $x_j$  che vale 1 se e solo se il numero appartiene al primo sottoinsieme. Con queste scelte, il modello (8.50)–(8.52) restituisce un sottoinsieme di massimo valore, senza che tale valore ecceda la metà del valore complessivo degli oggetti.

**Problema del Knapsack con due vincoli** Si consideri la variante del problema KP01 nella quale ciascun oggetto  $j$  è caratterizzato, oltre che dal profitto  $p_j$  e dal peso  $w_j$ , anche da un volume  $v_j > 0$ . Coerentemente, il knapsack

ha una capacità in peso  $C$  ed una capacità in volume pari a  $V$ . Utilizzando ancora le variabili  $x_j$ , il modello matematico corrispondente è il seguente

$$(2DKP) \quad \max \sum_{j=1}^n p_j x_j \quad (8.53)$$

$$\text{s.t.} \quad \sum_{j=1}^n w_j x_j \leq C \quad (8.54)$$

$$\sum_{j=1}^n v_j x_j \leq V \quad (8.55)$$

$$x_j \in \{0, 1\} \quad j = 1, \dots, n. \quad (8.56)$$

## 8.7 Problema del Knapsack Multiplo

**Definizione del problema** Consideriamo ora una variante più complessa del problema KP01 nella quale sono dati  $m$  knapsack, l' $i$ -esimo dei quali ha capacità pari a  $C_i$  ( $i = 1, \dots, m$ ). L'obiettivo del problema consiste nel determinare quali oggetti inserire in ciascun knapsack, in modo tale da evitare di eccedere la capacità di ciascun contenitore e da massimizzare il profitto complessivo degli oggetti selezionati.

**Modello PLI** Per modellare questo problema bisogna identificare le variabili decisionali; si noti che, a differenza del problema KP01, in questo caso non ci basta sapere se ciascun oggetto  $j$  è inserito in soluzione oppure no, ma ci serve sapere *in quale knapsack* viene inserito ciascun oggetto selezionato. In altre parole, non è più possibile utilizzare le  $n$  variabili  $x_j$ , in quanto queste non sono sufficienti a descrivere completamente una soluzione. Un possibile modo di codificare questa informazione è basato sull'utilizzo delle seguenti variabili binarie

$$x_{ij} = \begin{cases} 1 & \text{se l'oggetto } j \text{ viene inserito nel knapsack } i \\ 0 & \text{altrimenti} \end{cases} \quad (i = 1, \dots, m; j = 1, \dots, n) \quad (8.57)$$

In altre parole, per ciascun oggetto  $j$ , vengono introdotte  $m$  variabili binarie, una per ciascun knapsack (e non una sola, come avveniva in KP01).

Il corrispondente modello di PLI è il seguente

$$(MKP) \quad \max \sum_{j=1}^n \sum_{i=1}^m p_j x_{ij} \quad (8.58)$$

$$\text{s.t.} \quad \sum_{j=1}^n w_j x_{ij} \leq C_i \quad i = 1, \dots, m \quad (8.59)$$

$$\sum_{i=1}^m x_{ij} \leq 1 \quad j = 1, \dots, n \quad (8.60)$$

$$x_{ij} \in \{0, 1\} \quad i = 1, \dots, m; j = 1, \dots, n \quad (8.61)$$

I vincoli (8.59) sono  $m$  condizioni, una per ogni knapsack  $i$ . Il vincolo associato al generico knapsack  $i$  impone che la somma dei pesi degli oggetti inseriti nel knapsack sia non superiore alla capacità  $C_i$  del knapsack.

I vincoli (8.60) sono scritti uno per ogni oggetto  $j$ . Il vincolo associato al generico oggetto  $j$  impone che l'oggetto venga inserito in al più un knapsack, ovvero che esista al più una variabile che assume il valore 1 tra  $x_{1j}, x_{2j}, \dots, x_{mj}$ . Nel caso in cui vi sia una variabile che assume il valore 1, l'oggetto  $j$  sarà inserito nel knapsack corrispondente; viceversa, se  $x_{1j} = x_{2j} = \dots = x_{mj} = 0$ , allora l'oggetto  $j$  non è selezionato in soluzione in nessun knapsack.

Per interpretare la funzione obiettivo, consideriamo il generico oggetto  $j$ . Abbiamo visto che il termine di sinistra del corrispondente vincolo (8.60) indica se l'oggetto è stato selezionato in qualche knapsack oppure no. Di conseguenza, il profitto associato all'oggetto  $j$  in una soluzione è pari a

$$p_j \sum_{i=1}^m x_{ij} = \begin{cases} p_j & \text{se l'oggetto } j \text{ è selezionato in qualche knapsack} \\ 0 & \text{altrimenti} \end{cases} \quad (j = 1, \dots, n)$$

Sommmando il contributo di tutti gli oggetti si ottiene quindi la funzione obiettivo (8.58) da massimizzare.

**Esempio numerico** Consideriamo il seguente esempio numerico. Sono dati  $n = 5$  oggetti e due knapsack, con capacità pari a 50 e 70, rispettivamente. I profitti e pesi degli oggetti sono riportati dai seguenti vettori  $(p)$  e  $(w)$

$$(p) = (40, 62, 14, 20, 5)$$

$$(w) = (18, 33, 12, 21, 8)$$

Il modello di PLI associato a questa istanza prevede l'utilizzo di 10 variabili decisionali  $x_{11}, \dots, x_{15}, x_{21}, \dots, x_{25}$

$$\begin{array}{ccccccccccccc}
\max & 40x_{11} & +40x_{21} & +62x_{12} & +62x_{22} & +14x_{13} & +14x_{23} & +20x_{14} & +20x_{24} & +5x_{15} & +5x_{25} & \leq & 50 \\
& 18x_{11} & & +33x_{12} & & +12x_{13} & & +21x_{14} & & +8x_{15} & & & \\
& & 18x_{21} & & +33x_{22} & & +12x_{23} & & +21x_{24} & & +8x_{25} & & \leq & 70 \\
& & x_{11} & +x_{21} & & & & & & & & & & \leq & 1 \\
& & & & x_{12} & +x_{22} & & & & & & & & \leq & 1 \\
& & & & & & x_{13} & +x_{23} & & & & & & \leq & 1 \\
& & & & & & & & x_{14} & +x_{24} & & & & \leq & 1 \\
& & & & & & & & & & x_{15} & +x_{25} & \leq & 1 \\
& & x_{11} & , x_{21} & , x_{12} & , x_{22} & , x_{13} & , x_{23} & , x_{14} & , x_{24} & , x_{15} & , x_{25} & \in & \{0, 1\}
\end{array}$$

**Assunzioni** Come per il KP01, anche in questo caso possiamo fare delle assunzioni sui dati di ingresso. In particolare, senza perdita faremo le seguenti assunzioni:

1. La somma dei pesi di tutti gli oggetti eccede la capacità di ciascuno zaino, ovvero

$$\sum_{j=1}^n w_j > \max_{i=1,\dots,m} C_i$$

Infatti, se questa assunzione fosse violata, sarebbe possibile selezionare tutti gli oggetti in soluzione inserendoli nel contenitore con massima capacità, e non sarebbe quindi richiesta alcuna ottimizzazione.

2. Ciascun oggetto può essere inserito da solo in almeno un knapsack, ovvero

$$w_j \leq \max_{i=1,\dots,m} C_i \quad \forall j = 1, \dots, n$$

Infatti, immaginiamo che esista un oggetto  $k$  il cui peso supera la capacità del knapsack più grande. In nessuna soluzione ammissibile è possibile inserire l'oggetto  $k$ . Quindi, la soluzione ottima dell'istanza sarà ottenibile risolvendo una nuova istanza di MKP nella quale l'oggetto  $k$  è stato rimosso.

3. Ciascun knapsack è sufficientemente grande per contenere almeno un oggetto, ovvero

$$C_i \geq \min_{j=1,\dots,n} w_j \quad \forall i = 1, \dots, m$$

Infatti, immaginiamo che esista un knapsack  $h$  la cui capacità è minore del peso di qualunque oggetto. In nessuna soluzione ammissibile è possibile inserire un oggetto nel knapsack  $h$ . Quindi, la soluzione ottima dell'istanza sarà ottenibile risolvendo una nuova istanza di MKP nella quale il knapsack  $h$  è stato rimosso.

## 8.8 Problema del Bin Packing

**Definizione del problema** Sono dati

- un insieme  $M = \{1, \dots, m\}$  di contenitori (*bin*) identici, ciascuno con capacità  $C$ ;

- un insieme  $N = \{1, \dots, n\}$  di oggetti (*items*); ogni oggetto  $j$  è caratterizzato da un peso  $w_j > 0$ ;

Il problema del bin packing richiede di determinare un impaccamento degli oggetti nei contenitori in modo che

- ciascun oggetto sia inserito in un contenitore;
- la somma dei pesi degli oggetti inseriti in ciascun contenitore non ecceda la capacità  $C$ ;
- il numero dei contenitori utilizzati sia minimo

**Modello PLI** Per modellare il problema, occorre introdurre delle variabili decisionali che descrivono il riempimento di ciascun contenitore. Analogamente a quanto fatto nel problema del Multiple Knapsack, introduciamo le seguenti variabili decisionali

$$x_{ij} = \begin{cases} 1 & \text{se l'oggetto } j \text{ viene inserito nel bin } i \\ 0 & \text{altrimenti} \end{cases} \quad (i = 1, \dots, m; j = 1, \dots, n) \quad (8.62)$$

Per poter modellare la funzione obiettivo, introduciamo delle variabili decisionali aggiuntive, una per ciascun bin

$$y_i = \begin{cases} 1 & \text{se il bin } i \text{ viene utilizzato} \\ 0 & \text{altrimenti} \end{cases} \quad (i = 1, \dots, m) \quad (8.63)$$

Con questa scelta delle variabili decisionali, un possibile modello di PLI per il problema è il seguente

$$(BPP) \quad \min \sum_{i=1}^m y_i \quad (8.64)$$

$$\sum_{j=1}^n w_j x_{ij} \leq C y_i \quad i = 1, \dots, m \quad (8.65)$$

$$\sum_{i=1}^m x_{ij} = 1 \quad j = 1, \dots, n \quad (8.66)$$

$$x_{ij} \in \{0, 1\} \quad i = 1, \dots, m; j = 1, \dots, n \quad (8.67)$$

$$y_i \in \{0, 1\} \quad i = 1, \dots, m \quad (8.68)$$

La funzione obiettivo (8.64) minimizza la somma delle variabili  $y_i$ , ovvero il numero di bin utilizzati. I vincoli (8.65) impongono che, per ciascun bin  $i$ , la somma dei pesi degli oggetti inseriti nel bin  $i$  non ecceda la capacità  $C$ , mentre i vincoli (8.66) impongono che ciascun item  $j$  sia inserito in un contenitore. Infine, le condizioni (8.67)–(8.68) definiscono il dominio delle variabili.

Si osservi che l' $i$ -esimo vincolo di capacità (8.65) coinvolge la variabile  $y_i$  nel termine di destra. Questa cosa non accadeva nei vincoli (8.59) per il problema Multiple Knapsack in quanto, in quel caso, non era previsto nessun costo legato all'utilizzo di un bin, e quindi non serviva introdurre una variabile che indicasse se un certo bin era utilizzato oppure no. Nel caso del Bin Packing, invece, esiste un costo legato all'utilizzo di ciascun bin  $i$ , il che viene modellato tramite l'introduzione della variabile  $y_i$  associata. I vincili di capacità servono per "legare" le variabili  $x_{ij}$  con le variabili  $y_i$  in modo che la soluzione rispetti queste condizioni logiche:

- Si consideri un bin  $h$  tale che  $y_h = 1$ . Allora il vincolo di capacità associato impone che  $\sum_{j=1}^n w_j x_{hj} \leq Cy_h = C \cdot 1 = C$ ; in questo caso, il vincolo è del tutto analogo a quanto visto nel caso del Knapsack Multiplo.
- Viceversa, si consideri un bin  $h$  tale che  $y_h = 0$ . Allora il vincolo di capacità associato impone che  $\sum_{j=1}^n w_j x_{hj} \leq Cy_h = C \cdot 0 = 0$ , ovvero  $\sum_{j=1}^n w_j x_{hj} = 0$ . Dato che ciascun  $w_j > 0$  e  $x_{hj} \geq 0$ , allora deve essere  $w_j x_{hj} = 0$  per ogni  $j$ , ovvero  $x_{hj} = 0$  per ogni  $j$ . Quindi il vincolo impone la relazione logica  $y_h = 0 \Rightarrow x_{hj} = 0 \quad \forall j$ , ovvero "non utilizzo il bin  $h$ " allora "non posso inserire nessun oggetto dentro al bin  $h$ ".

In realtà i vincoli (8.65) descrivono anche le seguenti relazioni logiche aggiuntive:

- Si consideri un oggetto  $k$  ed bin  $h$  tali che  $x_{hk} = 1$ . Allora il vincolo di capacità associato al bin  $h$  impone che  $Cy_h \geq \sum_{j=1}^n w_j x_{hj} \geq w_k$ , dove l'ultima diseguaglianza deriva dal fatto che l'oggetto  $k$  è inserito nel bin  $h$ . Dividendo ambo i membri per  $C$  si ottiene la condizione  $y_h \geq \frac{w_k}{C} > 0$  dove ancora si è sfruttato il fatto che  $w_k > 0$ . Questo vincolo impone che la variabile  $y_h$  sia *strettamente* positiva;

Ci sono comunque delle relazioni logiche che non vengono modellate dai vincoli (8.65). Si consideri un bin  $h$  nel quale non viene inserito nessun item, cioè tale che  $x_{hj} = 0$  per ogni  $j$ . Allora il vincolo associato diventa  $0 = \sum_{j=1}^n w_j x_{hj} \leq C \cdot y_h$ , ovvero  $y_h \geq 0$ . Nonostante dal punto di vista logico ci si aspetti  $y_h = 0$ , dato che il bin  $h$  non è utilizzato, esistono delle soluzioni ammissibili anche con  $y_h = 1$ . A garantire la correttezza del modello contribuisce la funzione obiettivo che, volendo minimizzare la somma delle variabili  $y$ , metterà queste variabili a zero, ove possibile. In altre parole, in qualunque soluzione ottima la variabile  $y_h$  assumerà il valore zero, come atteso.

**Esempio numerico** Consideriamo il seguente esempio numerico. Sono dati  $n = 5$  oggetti e  $m = 5$  bin, ciascuno con capacità pari a 50. I pesi degli oggetti sono riportati dal seguente vettore  $(w) = (18, 33, 12, 21, 8)$

Il modello di PLI associato a questa istanza prevede l'utilizzo di 30 variabili decisionali

- 5 variabili  $y_1, y_2, \dots, y_5$ ;

- 25 variabili  $x_{11}, \dots, x_{15}, x_{21}, \dots, x_{25}, \dots, x_{51}, \dots, x_{55}$ .

$$\begin{array}{ccccccccc}
\min & y_1 & +y_2 & +y_3 & +y_4 & +y_5 & & \\
& 18x_{11} & +33x_{12} & +12x_{13} & +21x_{14} & +8x_{15} & \leq & 50y_1 \\
& 18x_{21} & +33x_{22} & +12x_{23} & +21x_{24} & +8x_{25} & \leq & 50y_2 \\
& 18x_{31} & +33x_{32} & +12x_{33} & +21x_{34} & +8x_{35} & \leq & 50y_3 \\
& 18x_{41} & +33x_{42} & +12x_{43} & +21x_{44} & +8x_{45} & \leq & 50y_4 \\
& 18x_{51} & +33x_{52} & +12x_{53} & +21x_{54} & +8x_{55} & \leq & 50y_5 \\
& x_{11} & +x_{21} & +x_{31} & +x_{41} & +x_{51} & = & 1 \\
& x_{12} & +x_{22} & +x_{32} & +x_{42} & +x_{52} & = & 1 \\
& x_{13} & +x_{23} & +x_{33} & +x_{43} & +x_{53} & = & 1 \\
& x_{14} & +x_{24} & +x_{34} & +x_{44} & +x_{54} & = & 1 \\
& x_{15} & +x_{25} & +x_{35} & +x_{45} & +x_{55} & = & 1 \\
& y_1 & , y_2 & , y_3 & , y_4 & , y_5 & \in & \{0, 1\} \\
& x_{11} & , x_{21} & , x_{31} & , x_{41} & , x_{51} & \in & \{0, 1\} \\
& x_{12} & , x_{22} & , x_{32} & , x_{42} & , x_{52} & \in & \{0, 1\} \\
& x_{13} & , x_{23} & , x_{33} & , x_{43} & , x_{53} & \in & \{0, 1\} \\
& x_{14} & , x_{24} & , x_{34} & , x_{44} & , x_{54} & \in & \{0, 1\} \\
& x_{15} & , x_{25} & , x_{35} & , x_{45} & , x_{55} & \in & \{0, 1\}
\end{array}$$

**Assunzioni** Nel caso del Bin Packing, è possibile fare le seguenti assunzioni sui dati di ingresso

1. Ciascun item può essere inserito singolarmente in un bin, ovvero

$$w_j \leq W \quad \forall j = 1, \dots, n$$

Infatti, se esistesse un oggetto  $k$  il cui peso supera la capacità del bin, non sarebbe possibile definire una soluzione ammissibile per il problema, dato che in qualunque soluzione verrebbe violato il vincolo di capacità di almeno un contenitore.

2. Il numero dei bin a disposizione è maggiore o uguale al numero degli oggetti, ovvero

$$m \geq n$$

Seppur questa assunzione sia con perdita di generalità, la sua presenza, assieme all'assunzione precedente, garantisce l'esistenza di almeno una soluzione ammissibile (ad esempio la soluzione che impacca ciascun oggetto  $j$  nel bin  $j$ -esimo per  $j = 1, \dots, n$ ).

### 8.8.1 Varianti del problema

**Problema del Vector Packing** In questa variante del problema, gli oggetti ed i contenitori sono caratterizzati da  $k$  dimensioni distinte (peso, volume, ...). Di conseguenza, ogni oggetto  $j \in N$  ha associato un valore  $w_j^t$  per ogni dimensione  $t = 1, \dots, k$ , ed ogni bin ha una capacità  $W^t$  per ogni dimensione  $t = 1, \dots, k$ .

Introducendo ancora le variabili binarie  $x_{ij}$  e  $y_i$ , si ottiene il seguente modello PLI

$$(VPP) \quad \min \sum_{i=1}^m y_i \quad (8.69)$$

$$\sum_{j=1}^n w_j^t x_{ij} \leq W^t y_i \quad i = 1, \dots, m; t = 1, \dots, k \quad (8.70)$$

$$\sum_{i=1}^m x_{ij} = 1 \quad j = 1, \dots, n \quad (8.71)$$

$$x_{ij} \in \{0, 1\} \quad i = 1, \dots, m; j = 1, \dots, n \quad (8.72)$$

$$y_i \in \{0, 1\} \quad i = 1, \dots, m \quad (8.73)$$

## 8.9 Problema del Set Covering

**Definizione del problema** Sono Dati

- una matrice  $A$  con  $m$  righe e  $n$  colonne e coefficienti binari  $[a_{ij}]$ ;
- un vettore di dimensione  $n$  di costi  $[c_j]$ ;

Il problema del set covering richiede di selezionare un sottoinsieme  $S \subseteq \{1, \dots, n\}$  di colonne in modo che:

- per ogni riga  $i = 1, \dots, m$ , esista almeno una colonna  $j \in \{1, \dots, n\}$  tale che
  - $a_{ij} = 1$  (la colonna  $j$  “copre” la riga  $i$ );
  - $j \in S$  (ossia  $j$  è selezionata)
- il costo complessivo delle colonne selezionate sia minimo.

**Modello PLI** Per modellare il problema occorre definire quali sono le colonne selezionate in soluzione. Pertanto la scelta più naturale è quella di introdurre, per ogni colonna  $j$ , una variabile binaria che indichi se la colonna è selezionata in soluzione oppure no, ossia

$$x_j = \begin{cases} 1 & \text{se la colonna } j \text{ è selezionata} \\ 0 & \text{altrimenti} \end{cases} \quad (j = 1, \dots, n) \quad (8.74)$$

Con questa scelta delle variabili decisionali, un possibile modello di PLI per il problema è il seguente

$$(SCP) \quad \min \sum_{j=1}^n c_j x_j \quad (8.75)$$

$$s.t. \quad \sum_{j=1}^n a_{ij} x_j \geq 1 \quad i = 1, \dots, m \quad (8.76)$$

$$x_j \in \{0, 1\} \quad j = 1, \dots, n. \quad (8.77)$$

I vincoli (8.76) impongono che, per ciascuna riga  $i$ , esista almeno una colonna  $j$  che (i) abbia  $a_{ij} = 1$  e (ii) sia selezionata (ossia abbia  $x_j = 1$ ). Si noti che il problema richiede di garantire che la proprietà di copertura di ciascuna riga sia soddisfatta, ma non di specificare quale colonna copre ciascuna riga.

**Esempio numerico** Consideriamo l'esempio numerico definito dalla seguente matrice A con  $m = 3$  righe e  $n = 6$  colonne

$$(a_{ij}) = \begin{matrix} & 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & [ & 1 & 0 & 1 & 0 & 1 & 0 \\ 2 & & 0 & 0 & 1 & 1 & 0 & 1 \\ 3 & & 0 & 1 & 0 & 0 & 1 & 1 \end{matrix}$$

e da un vettore dei costi delle colonne  $(c) = [9, 3, 2, 4, 6, 3]$ .

Per l'istanza numerica sopra specificata, si ottiene il seguente modello di PLI con 6 variabili e 3 vincoli

$$\begin{array}{lllllllllll} \min & 9x_1 & +3x_2 & +2x_3 & +4x_4 & +6x_5 & +3x_6 & & & & \\ & x_1 & & +x_3 & & +x_5 & & \geq & & 1 \\ & & & x_3 & +x_4 & & +x_6 & \geq & & 1 \\ & & & x_2 & & +x_5 & +x_6 & \geq & & 1 \\ x_1 & , & x_2 & , & x_3 & , & x_4 & , & x_5 & , & x_6 & \in \{0, 1\} \end{array}$$

Una possibile soluzione ammissibile consiste nel selezionare le colonne  $\{1, 3, 5\}$ , ed ha costo = 17, mentre una soluzione ottima è costituita dall'insieme  $S = \{2, 3\}$  ed ha costo = 5.

**Assunzioni** Senza perdita di generalità possiamo fare le seguenti assunzioni sui dati di ingresso:

1. Per ogni riga  $i$  esiste almeno una colonna  $j$  che può coprire la riga  $i$ , ossia tale che  $a_{ij} = 1$ .

È facile vedere che, se questa assunzione non è verificata, non è possibile coprire la riga  $i$  ed il problema non ammette nessuna soluzione ammissibile. Viceversa, se questa assunzione è soddisfatta, il problema ha almeno una soluzione ammissibile, ad esempio la soluzione che seleziona tutte le colonne.

2. Per ogni colonna  $j$ , esiste almeno una riga  $i$  che può essere coperta dalla colonna  $j$ , ossia tale che  $a_{ij} = 1$ .

Infatti, se esistesse una colonna  $j$  che non copre nessuna riga, questa colonna non farebbe mai parte di una soluzione ottima del problema (se  $c_j > 0$ ) oppure farebbe sempre parte di una soluzione ottima (se  $c_j \leq 0$ ). Quindi, la soluzione ottima dell'istanza sarebbe ricavabile risolvendo una nuova istanza di set covering nella quale la colonna  $j$  è stata rimossa.

3. Tutte le colonne hanno costi positivi, ossia

$$c_j > 0 \quad \forall j = 1, \dots, n$$

Immaginiamo infatti che questa assunzione sia violata ed esista una colonna  $j$  con costo negativo (o nullo). È facile vedere che esiste una soluzione ottima nella quale la colonna  $j$  è selezionata. Pertanto, la soluzione ottima dell'istanza sarebbe ottenibile risolvendo una nuova istanza di set covering nella quale la colonna  $j$  è stata selezionata.

**Applicazioni** Si consideri la seguente applicazione in ambito urbanistico, nella quale l'amministrazione locale di una città deve definire dove posizionare le nuove stazioni dei vigili del fuoco. La mappa della città è suddivisa in  $n$  aree (es. i quartieri) in ciascuna delle quali può essere localizzata una stazione; in tal caso, l'amministrazione incorre in un costo specifico dell'area, in funzione, ad esempio, del diverso valore del terreno edificabile nelle diverse aree. Se una stazione viene edificata in un'area essa può gestire le chiamate di emergenza nella propria area ed in quelle adiacenti. L'obiettivo dell'amministrazione è determinare un sottoinsieme di stazioni da costruire, in modo che

- in caso di emergenza, ogni zona possa essere servita da una stazione dei vigili del fuoco;
- il costo complessivo di costruzione sia il più basso possibile

Il requisito di adiacenza tra le zone può essere modellato definendo in modo opportuno i coefficienti di una matrice  $A$  di dimensione  $n \times n$ . In particolare  $a_{ij} = 1$  se la stazione localizzata nell'area  $j$  può servire le richieste dell'area  $i$ , e  $a_{ij} = 0$  in caso contrario. Un esempio di città suddivisa in aree è mostrato nella figura 8.2 e la relativa matrice  $A$  è illustrata nella figura 8.3.

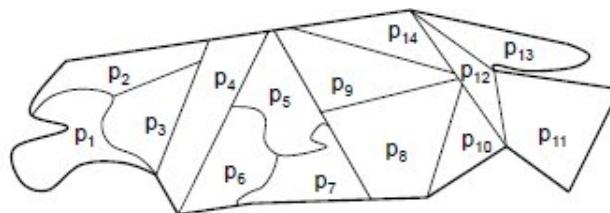


Figura 8.2: Esempio di applicazione del set covering per la localizzazione di stazioni dei pompieri in una regione suddivisa in aree.

	$S_1$	$S_2$	$S_3$	$S_4$	$S_5$	$S_6$	$S_7$	$S_8$	$S_9$	$S_{10}$	$S_{11}$	$S_{12}$	$S_{13}$	$S_{14}$
$p_1$	1	1	1	0	0	0	0	0	0	0	0	0	0	0
$p_2$	1	1	1	1	0	0	0	0	0	0	0	0	0	0
$p_3$	1	1	1	1	0	0	0	0	0	0	0	0	0	0
$p_4$	0	1	1	1	1	1	0	0	0	0	0	0	0	0
$p_5$	0	0	0	1	1	1	1	1	1	0	0	0	0	0
$p_6$	0	0	0	1	1	1	1	0	0	0	0	0	0	0
$p_7$	0	0	0	0	1	0	1	1	0	0	0	0	0	0
$p_8$	0	0	0	0	1	0	1	1	1	1	0	1	0	0
$p_9$	0	0	0	0	1	0	0	1	1	0	0	1	0	1
$p_{10}$	0	0	0	0	0	0	0	1	0	1	0	1	0	0
$p_{11}$	0	0	0	0	0	0	0	0	0	0	1	1	0	0
$p_{12}$	0	0	0	0	0	0	0	1	1	1	1	1	1	1
$p_{13}$	0	0	0	0	0	0	0	0	0	0	0	1	1	0
$p_{14}$	0	0	0	0	0	0	0	0	1	0	0	1	0	1

Figura 8.3: Matrice dei coefficienti del problema di set covering che modella il problema di localizzazione delle stazioni dei pompieri.

Per ogni area  $j = 1, \dots, n$  viene introdotta una variabile decisionale  $x_j$  che assume valore 1 se in tale area viene costruita una stazione e 0 altrimenti.

Si osservi noti che non è richiesto specificare, tramite variabili decisionali, quale stazione viene utilizzata per gestire le emergenze di una certa area, in quanto il problema richiede solamente di garantire che esista una stazione adeguata (cioè posizionata in una area adiacente a quella di interesse). Infine, si noti che la matrice  $A$  consente di definire qualsiasi tipo di compatibilità tra due aree  $i$  e  $j$ . Ad esempio, è possibile modellare la possibilità per una stazione posizionata nell'area  $j$  di coprire una emergenza nell'area  $i$  ponendo  $a_{ij} = 1$  indipendentemente dalla adiacenza delle due aree. Inoltre, la matrice non deve necessariamente essere simmetrica, ossia si può avere  $a_{ij} = 1$  e  $a_{ji} = 0$ .

## 8.10 Problema del Set Partitioning

**Definizione del problema** Sono Dati

- una matrice  $A$  con  $m$  righe e  $n$  colonne e coefficienti binari  $[a_{ij}]$ ;
- un vettore di dimensione  $n$  di costi  $[c_j]$ ;

Il problema del set partitioning richiede di selezionare un sottoinsieme  $S \subseteq \{1, \dots, n\}$  di colonne in modo che:

- per ogni riga  $i = 1, \dots, m$ , esista esattamente una colonna  $j \in \{1, \dots, n\}$  tale che

- $a_{ij} = 1$  (la colonna  $j$  “copre” la riga  $i$ );
- $j \in S$  (ossia  $j$  è selezionata)
- il costo complessivo delle colonne selezionate sia minimo.

Questo problema è analogo al problema del set covering, salvo per il fatto che ogni riga deve essere coperta da una ed una sola colonna selezionata (e non da *almeno* una colonna selezionata, come era nel set covering).

**Modello PLI** Il modello PLI del set partitioning utilizza le stesse variabili decisionali  $x_j$  definite per il modello del set covering, e differisce da quest’ultimo solo per i vincoli di copertura (8.79), che sono imposti come uguaglianza invece che come disuguaglianza.

$$(SPP) \quad \min \sum_{j=1}^n c_j x_j \quad (8.78)$$

$$\text{s.t.} \quad \sum_{j=1}^n a_{ij} x_j \geq 1 \quad i = 1, \dots, m \quad (8.79)$$

$$x_j \in \{0, 1\} \quad j = 1, \dots, n. \quad (8.80)$$

**Esempio numerico** Riprendendo l’esempio numerico della sezione 8.9, il modello corrispondente è il seguente:

$$\begin{array}{ccccccc} \min & 9x_1 & +3x_2 & +2x_3 & +4x_4 & +6x_5 & +3x_6 \\ & x_1 & & +x_3 & & +x_5 & = & 1 \\ & & & x_3 & +x_4 & & +x_6 & = & 1 \\ & & & x_2 & & +x_5 & +x_6 & = & 1 \\ & x_1 & , & x_2 & , & x_3 & , & x_4 & , & x_5 & , & x_6 & \in & \{0, 1\} \end{array}$$

**Assunzioni** Nonostante il set partitioning sia molto simile al set covering, non tutte le assunzioni che si possono fare sui dati di ingresso dei due problemi sono uguali.

1. Per ogni riga  $i$  esiste almeno una colonna  $j$  che può coprire la riga  $i$ , ossia tale che  $a_{ij} = 1$ .

Questa assunzione è identica a quella vista per il set covering. Nonostante questo, non è possibile affermare che, quando questa assunzione è verificata, il problema ha almeno una soluzione ammissibile, come avveniva nel caso del set covering. Si consideri ad esempio un problema con 3 righe e 3

colonne e matrice  $A$  definita in questo modo:  $(a_{ij}) = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$

È evidente che selezionando due delle tre colonne si ottiene una soluzione di set covering, mentre non esiste nessuna soluzione di set partitioning.

2. Per ogni colonna  $j$ , esiste almeno una riga  $i$  che può essere coperta dalla colonna  $j$ , ossia tale che  $a_{ij} = 1$ .

Questa assunzione è identica a quella vista per il set covering.

**Nota:** nel problema del Set Partitioning non è possibile fare l'assunzione

$$c_j > 0 \quad \forall j = 1, \dots, n$$

in quanto selezionare una colonna a costo negativo potrebbe coprire alcune righe, con conseguente perdita di alcune soluzioni ammissibili (eventualmente anche della soluzione ottima).

## 8.11 Problema del Set Packing

**Definizione del problema** Sono Dati

- una matrice  $A$  con  $m$  righe e  $n$  colonne e coefficienti binari  $[a_{ij}]$ ;
- un vettore di dimensione  $n$  di profitti  $[p_j]$ ;

Il problema del set packing richiede di selezionare un sottoinsieme  $S \subseteq \{1, \dots, n\}$  di colonne in modo che:

- per ogni riga  $i = 1, \dots, m$ , esista al massimo una colonna  $j \in \{1, \dots, n\}$  tale che
  - $a_{ij} = 1$  (la colonna  $j$  “copre” la riga  $i$ );
  - $j \in S$  (ossia  $j$  è selezionata)
- il profitto complessivo delle colonne selezionate sia massimo.

Questo problema è speculare rispetto al problema del set covering, nel senso che si vuole coprire ogni riga al massimo una volta (invece che almeno una volta) massimizzando il profitto delle colonne scelte (invece che minimizzando il costo).

**Modello PLI** Il modello PLI del set packing utilizza le stesse variabili decisionali  $x_j$  definite per il modello del set covering, e differisce da quest'ultimo solo per la funzione obiettivo e per i vincoli di copertura.

$$(SPkP) \quad \max \sum_{j=1}^n p_j x_j \quad (8.81)$$

$$s.t. \quad \sum_{j=1}^n a_{ij} x_j \leq 1 \quad i = 1, \dots, m \quad (8.82)$$

$$x_j \in \{0, 1\} \quad j = 1, \dots, n. \quad (8.83)$$

**Esempio numerico** Riprendendo l'esempio numerico della sezione 8.9, immaginando che il vettore  $[c]$  descriva i profitti delle colonne, il modello corrispondente è il seguente:

$$\begin{array}{ccccccccc} \max & 9x_1 & +3x_2 & +2x_3 & +4x_4 & +6x_5 & +3x_6 & & \\ & x_1 & & +x_3 & & +x_5 & & \leq & 1 \\ & & & x_3 & +x_4 & & +x_6 & \leq & 1 \\ & & & x_2 & & +x_5 & +x_6 & \leq & 1 \\ & x_1 & , & x_2 & , & x_3 & , & x_4 & , & x_5 & , & x_6 & \in & \{0, 1\} \end{array}$$

**Applicazioni** Un'amministrazione regionale deve decidere la localizzazione degli inceneritori nelle province della regione. Nelle 8 province della regione sono state individuate in totale 10 aree opportune in cui localizzare gli inceneritori e nella tabella di figura 8.4 per ciascuna area sono indicate le province adiacenti a tale area. Nella figura è anche indicata la capacità di trattamento (in milioni di tonnellate di rifiuti) ed il costo di costruzione (in milioni di Euro) di un inceneritore in ciascuna area.

Il problema di localizzazione consiste nell'individuare in quali aree costruire gli inceneritori in modo tale che la capacità di trattamento sia massimizzata e ciascuna provincia sia adiacente al più ad un inceneritore costruito. Eventualmente al problema può essere aggiunto un vincolo sul costo complessivo che impone che il costo totale degli inceneritori costruiti non superi il limite del budget disponibile.

	<b>p<sub>1</sub></b>	<b>p<sub>2</sub></b>	<b>p<sub>3</sub></b>	<b>p<sub>4</sub></b>	<b>p<sub>5</sub></b>	<b>p<sub>6</sub></b>	<b>p<sub>7</sub></b>	<b>p<sub>8</sub></b>	<b>p<sub>9</sub></b>	<b>p<sub>10</sub></b>
<b>C<sub>1</sub></b>	1			1		1				1
<b>C<sub>2</sub></b>		1			1		1		1	
<b>C<sub>3</sub></b>	1				1		1		1	
<b>C<sub>4</sub></b>	1		1			1				1
<b>C<sub>5</sub></b>		1				1	1		1	
<b>C<sub>6</sub></b>		1		1		1				
<b>C<sub>7</sub></b>	1				1			1	1	1
<b>C<sub>8</sub></b>			1	1		1		1		
<b>CAP</b>	460	774	129	499	345	997	321	159	721	640
<b>COST</b>	7	11	3	7.5	6.5	12.2	5.1	3.2	10.7	8.8

Figura 8.4: Esempio di applicazione del set packing per la localizzazione di inceneritori a servizio delle 8 province di una regione.

## 8.12 Problema dell'Assegnamento

**Definizione del problema** Data una matrice  $C$  con  $n$  righe e  $n$  colonne, e coefficienti di costo  $[c_{ij}]$ , il problema dell'assegnamento richiede di determinare un assegnamento delle colonne alle righe in modo che:

- ogni colonna  $j = 1, \dots, n$  sia assegnata ad una riga;
- ogni riga  $i = 1, \dots, n$  sia assegnata ad una colonna;
- il costo complessivo dell'assegnamento sia minimo.

**Modello PLI** Una possibile soluzione per modellare il problema consiste nell'introdurre le seguenti variabili decisionali  $x_{ij}$  per ogni riga  $i$  e colonna  $j$ .

$$x_{ij} = \begin{cases} 1 & \text{se la riga } i \text{ è assegnata alla colonna } j \\ 0 & \text{altrimenti} \end{cases} \quad (i = 1, \dots, n; j = 1, \dots, n) \quad (8.84)$$

Con tale scelta delle variabili, si ottiene il seguente modello di PLI

$$(AP) \quad \min \sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij} \quad (8.85)$$

$$\sum_{i=1}^n x_{ij} = 1 \quad j = 1, \dots, n \quad (8.86)$$

$$\sum_{j=1}^n x_{ij} = 1 \quad i = 1, \dots, n \quad (8.87)$$

$$x_{ij} \in \{0, 1\} \quad i = 1, \dots, n; j = 1, \dots, n \quad (8.88)$$

I vincoli (8.86) impongono che ciascuna colonna  $j$  sia assegnata ad una sola riga. Analogamente i vincoli (8.87) impongono di assegnare ciascuna riga  $i$  ad una colonna. Infine, la funzione obiettivo minimizza il costo degli assegnamenti selezionati.

Si noti che, nel formulare il problema, non è stata fatta nessuna asunzione per quel che riguarda il segno dei coefficienti  $c_{ij}$ .

### 8.12.1 Problema dell'Assegnamento Generalizzato

Sono dati

- un insieme  $N = \{1, \dots, n\}$  di oggetti;
- un insieme  $M = \{1, \dots, m\}$  di risorse;
- ogni risorsa  $i \in M$  ha una disponibilità  $b_i$

- per ogni oggetto  $j \in N$  e risorsa  $i \in M$ :
  - la richiesta  $r_{ij}$  dell'oggetto  $j$  se assegnato alla risorsa  $i$ ;
  - il costo  $c_{ij}$  di assegnamento dell'oggetto  $j$  alla risorsa  $i$

Il problema dell'assegnamento generalizzato richiede di assegnare gli oggetti alle risorse in modo che:

- ogni oggetto  $j \in N$  sia assegnato ad una risorsa;
- per ogni risorsa  $i \in M$ , la richiesta complessiva degli oggetti assegnati alla risorsa non ecceda la disponibilità  $b_i$ ;
- il costo complessivo dell'assegnamento sia minimo

Come suggerisce il nome, questo problema generalizza il problema dell'assegnamento al caso in cui il numero di righe da assegnare differisce dal numero di colonne. In questo contesto, identifichiamo le righe come *risorse*, che hanno una certa disponibilità e a cui devono essere assegnati degli *oggetti*, identificati dalle colonne. Utilizzando ancora le variabili di assegnamento  $x_{ij}$  ( $i = 1, \dots, m; j = 1, \dots, n$ ), il problema può essere modellato nel seguente modo

$$(GAP) \quad \min \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \quad (8.89)$$

$$\sum_{i=1}^m x_{ij} = 1 \quad j = 1, \dots, n \quad (8.90)$$

$$\sum_{j=1}^n r_{ij} x_{ij} \leq b_i \quad i = 1, \dots, m \quad (8.91)$$

$$x_{ij} \in \{0, 1\} \quad i = 1, \dots, m; j = 1, \dots, n \quad (8.92)$$

## 8.13 Problemi di facility location

**Definizione del problema** Dati

- un insieme  $N = \{1, \dots, n\}$  di risorse (*facilities*) che possono essere attivate; ciascuna facility  $j$  ha associato un costo fisso  $f_j$  di installazione;
- un insieme  $M = \{1, \dots, m\}$  di clienti da servire;
- una matrice  $C$  di dimensione  $m \times n$  il cui generico elemento  $c_{ij}$  rappresenta il costo richiesto per servire il cliente  $i$  dalla facility  $j$ ;

Problema: determinare un sottoinsieme di facilities  $S \subseteq N$  da attivare e l'assegnamento dei clienti alle facilities attivate in modo che

- ciascun cliente sia servito da una facility;
- il costo complessivo (=costo fisso + costo di servizio) sia il più piccolo possibile.

**Modello PLI** In questo problema bisogna operare due tipi di scelte: da un lato, occorre definire l'insieme di facilities da attivare. Dall'altro, occorre assegnare ciascun cliente ad una facility attivata. Il primo insieme di decisioni è analogo a quanto visto nei problemi di tipo knapsack (selezione), e quindi richiede l'utilizzo delle seguenti variabili decisionali

$$y_j = \begin{cases} 1 & \text{se la facility } j \text{ è attivata} \\ 0 & \text{altrimenti} \end{cases} \quad (j = 1, \dots, n) \quad (8.93)$$

Viceversa, l'assegnamento dei clienti alle facilities può essere modellato utilizzando le variabili di assegnamento

$$x_{ij} = \begin{cases} 1 & \text{se il cliente } i \text{ è assegnato alla facility } j \\ 0 & \text{altrimenti} \end{cases} \quad (i = 1, \dots, m; j = 1, \dots, n) \quad (8.94)$$

Il modello risultante è il seguente

$$(UFLP) \quad \min \sum_{j=1}^n f_j y_j + \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \quad (8.95)$$

$$\sum_{j=1}^n x_{ij} = 1 \quad i = 1, \dots, m \quad (8.96)$$

$$x_{ij} \leq y_j \quad i = 1, \dots, m; j = 1, \dots, n \quad (8.97)$$

$$y_j \in \{0, 1\} \quad j = 1, \dots, n \quad (8.98)$$

$$x_{ij} \in \{0, 1\} \quad i = 1, \dots, m; j = 1, \dots, n \quad (8.99)$$

La funzione obiettivo minimizza la somma dei costi di apertura delle facilities e dei costi di assegnamento. I vincoli (8.96) sono i classici vincoli di assegnamento per ciascun cliente  $i$ , mentre i vincoli (8.97) impongono che un cliente  $i$  possa essere assegnato ad una facility  $j$  solo se tale facility è stata attivata. Si noti che, in questo problema, ciascuna facility attivata può gestire un numero arbitrario di clienti, ovvero non esiste nessun limite alla capacità della facility di servire dei clienti. Per questo motivo, questa variante del problema è nota come Uncapacitated Facility Location Problem (UFLP).

**Facility Location capacitato** Consideriamo ora una variante del problema nella quale ciascuna facility  $j$  ha associato un costo fisso  $f_j$  di installazione ed una capacità  $b_j$ , e ciascun cliente  $i$  ha una richiesta pari a  $d_i$ . Il problema

richiede ancora di assegnare i clienti alle facilities attivate, minimizzando i costi e garantendo che ciascuna facility abbia capacità sufficiente per servire tutti i clienti che le sono stati assegnati.

Il problema in questione viene detto Capacitated Facility Location Problem (CFLP). Un modello PLI di questo problema che utilizza le stesse variabili viste prima è il seguente

$$(CFLP) \quad \min \sum_{j=1}^n f_j y_j + \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \quad (8.100)$$

$$\sum_{j=1}^n x_{ij} = 1 \quad i = 1, \dots, m \quad (8.101)$$

$$\sum_{i=1}^m d_i x_{ij} \leq b_j y_j \quad j = 1, \dots, n \quad (8.102)$$

$$y_j \in \{0, 1\} \quad j = 1, \dots, n \quad (8.103)$$

$$x_{ij} \in \{0, 1\} \quad i = 1, \dots, m; j = 1, \dots, n \quad (8.104)$$

**Facility Location capacitato con splitting** Infine, consideriamo una variante del probelma CFLP nella quale la richiesta di ogni cliente possa essere servita da più risorse (caso “splittable”). Questa possibilità permette, in generale, di ottenere soluzioni di costo inferiore e quindi potrebbe essere preferibile in alcune applicazioni reali.

Per modellare questa variante, utilizziamo ancora le variabili  $y_j$  di attivazione delle facilities. Quando alle variabili  $x_{ij}$ , immaginiamo che queste siano variabili continue nel dominio  $[0, 1]$  e che ciascuna variabile rappresenti la frazione della domanda del cliente  $i$  che viene servita dalla facility  $j$ . Con tale convenzione, il modello che si ottiene è il seguente

$$(CFLPs) \quad \min \sum_{j=1}^n f_j y_j + \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \quad (8.105)$$

$$\sum_{j=1}^n x_{ij} = 1 \quad i = 1, \dots, m \quad (8.106)$$

$$\sum_{i=1}^m d_i x_{ij} \leq b_j y_j \quad j = 1, \dots, n \quad (8.107)$$

$$y_j \in \{0, 1\} \quad j = 1, \dots, n \quad (8.108)$$

$$x_{ij} \in [0, 1] \quad i = 1, \dots, m; j = 1, \dots, n \quad (8.109)$$

Seppur il significato delle variabili  $x_{ij}$  sia diverso dal precedente, il modello ottenuto risulta essere molto simile a quello che descrive CFLP. In particolare, i vincoli (8.106) impongono che ciascun cliente  $i$  sia servito al 100% considerando l'insieme di facilities che lo servono.

## 8.14 Problemi di scheduling

**Definizione del problema** Dati

- un insieme  $N = \{1, \dots, n\}$  di jobs;
- un insieme  $M = \{1, \dots, m\}$  di macchine identiche, ciascuna delle quali può processare un solo lavoro alla volta;
- per ogni job  $j \in N$  e macchina  $i \in M$  è noto il tempo di processamento  $p_{ij}$  del lavoro  $j$  sulla macchina  $i$ .

Problema: assegnare ogni lavoro ad una macchina in modo che

- ogni lavoro sia assegnato ad una sola macchina;
- il tempo totale di processamento della macchina che termina per ultima (*makespan*) sia il più piccolo possibile.

**Esempio numerico** Si consideri il seguente esempio numerico, nel quale sono dati  $n = 6$  jobs e  $m = 2$  macchine. I tempi di processamento dei job sulle macchine sono i seguenti

$$p_{11} = 3; p_{12} = 8; p_{13} = 2; p_{14} = 4; p_{15} = 10; p_{16} = 5$$

$$p_{21} = 2; p_{22} = 6; p_{23} = 6; p_{24} = 5; p_{25} = 8; p_{26} = 6$$

Ad esempio, assegnando alla prima macchina i task 1,2 e 3, tale macchina richiede  $3+8+2 = 13$  unità di tempo, mentre assegnando alla seconda macchina i restanti task, questa impiegherebbe  $5+8+6 = 19$  unità di tempo. Il valore di questa soluzione sarebbe quindi  $\max\{13, 19\} = 19$ .

La soluzione che assegna alla prima macchina i task 3, 4, 5 e alla seconda macchina i task 1, 2, 6 ha invece valore  $\max\{2 + 4 + 10, 19, 2 + 6 + 6\} = \max\{16, 14\} = 14$ .

**Modello PLI** Per modellare questo problema occorre anzitutto osservare che il tempo di processamento di una macchina non dipende dall'ordine con il quale la macchina processa i suoi lavori, ma solo dall'insieme di lavori che le vengono assegnati. Pertanto, sembra naturale introdurre le seguenti variabili di assegnamento di lavori a macchine

$$x_{ij} = \begin{cases} 1 & \text{se il lavoro } j \text{ è assegnato alla macchina } i \\ 0 & \text{altrimenti} \end{cases} \quad (i = 1, \dots, m; j = 1, \dots, n) \quad (8.110)$$

Per comodità, possiamo introdurre le seguenti variabili decisionali

$$C_i = \text{tempo totale di processamento della macchina } i \quad (i = 1, \dots, m) \quad (8.111)$$

Con questa scelta delle variabili decisionali, non è però possibile esprimere la funzione obiettivo utilizzando vincoli lineari. Per ottenere un modello lineare (intero), occorre lavorare in uno spazio delle variabili a dimensionalità superiore, ottenuto aggiungendo una ulteriore variabile decisionale

$$z = \text{makespan} \quad (8.112)$$

Il corrispondente modello matematico è il seguente

$$\min z \quad (8.113)$$

$$\sum_{i=1}^m x_{ij} = 1 \quad j = 1, \dots, n \quad (8.114)$$

$$C_i = \sum_{j=1}^n p_{ij} x_{ij} \quad i = 1, \dots, m \quad (8.115)$$

$$z \geq C_i \quad i = 1, \dots, m \quad (8.116)$$

$$x_{ij} \in \{0, 1\} \quad i = 1, \dots, m; j = 1, \dots, n \quad (8.117)$$

I vincoli (8.114) impongono l'assegnamento di ciascun task  $j$  ad una macchina, mentre le equazioni (8.115) definiscono il tempo totale di processamento di ciascuna macchina. Infine, la funzione obiettivo minimizza il makespan (rappresentato dalla variabile  $z$ ) che, per via dei vincoli (8.116), risulta essere almeno pari al tempo totale di processamento di ciascuna macchina.

Il modello PLI associato all'esempio numerico introdotto in precedenza è il seguente

$$\begin{aligned} & \min z \\ & x_{11} + x_{21} = 1 \\ & x_{12} + x_{22} = 1 \\ & x_{13} + x_{23} = 1 \\ & x_{14} + x_{24} = 1 \\ & x_{15} + x_{25} = 1 \\ & x_{16} + x_{26} = 1 \\ & C_1 = 3x_{11} + 8x_{12} + 2x_{13} + 4x_{14} + 10x_{15} + 5x_{16} \\ & C_2 = 2x_{21} + 6x_{22} + 6x_{23} + 5x_{24} + 8x_{25} + 6x_{26} \\ & z \geq C_1 \\ & z \geq C_2 \\ & x_{11}, x_{12}, x_{13}, x_{14}, x_{15}, x_{16}, x_{21}, x_{22}, x_{23}, x_{24}, x_{25}, x_{26} \in \{0, 1\} \end{aligned}$$

Si noti che le variabili  $C_i$  non sono strettamente indispensabili, ed è possibile eliminarle sostituendo i vincoli (8.115)-(8.116) con i seguenti vincoli

$$z \geq \sum_{j=1}^n p_{ij} x_{ij} \quad i = 1, \dots, m \quad (8.118)$$

## 8.15 Problema del lot sizing

**Definizione del problema** Dati

- un orizzonte temporale suddiviso in  $n$  periodi; ogni periodo  $i$  ( $i = 1, \dots, n$ ) ha associate
  - il numero  $d_i$  di prodotti (identici) richiesti dal mercato nel periodo;
  - il costo unitario di produzione  $c_i$  nel periodo;
  - il costo di stoccaggio unitario  $r_i$  nel periodo;
- un magazzino, con capacità di stoccaggio pari a  $M$  unità di prodotto;
- la giacenza iniziale  $s_0$  presente all'interno del magazzino all'inizio dell'orizzonte temporale in esame.

Problema: determinare la quantità di prodotti da realizzare e da stoccare in ogni periodo in modo che

- in ogni periodo, il numero di prodotti disponibili sia sufficiente per soddisfare la richiesta del mercato;
- il costo complessivo sostenuto (=costi di produzione + costi di stoccaggio) sia il più basso possibile.

**Modello PLI** Per definire una soluzione del problema occorre determinare il numero di prodotti da realizzare ed il numero di prodotti da stoccare in ogni periodo. Questo suggerisce l'introduzione delle seguenti variabili decisionali  
 $x_i$  = numero di prodotti realizzati nell' $i$ -esimo periodo ( $i = 1, \dots, n$ )  
 $s_i$  = num. di prodotti nel magazzino alla fine dell' $i$ -esimo periodo ( $i = 1, \dots, n$ )

Il corrispondente modello matematico è quindi

$$\min \sum_{i=1}^n c_i x_i + \sum_{i=1}^n r_i s_i \quad (8.119)$$

$$s_i \leq M \quad i = 1, \dots, n \quad (8.120)$$

$$x_i + s_{i-1} - s_i = d_i \quad i = 1, \dots, n \quad (8.121)$$

$$x_i, s_i \geq 0 \quad i = 1, \dots, n \quad (8.122)$$

La funzione obiettivo minimizza il costo complessivo, dati dai costi di produzione e da quelli di stoccaggio. I vincoli (8.120) garantiscono che il numero di prodotti stoccati in ciascun periodo non ecceda la capacità del magazzino, mentre i vincoli (8.121) definiscono la quantità stoccati alla fine di ciascun periodo: tale quantità risulta essere pari alla quantità presente a inizio periodo (=presente alla fine del periodo precedente), cui viene sommata la quantità prodotta e detratta la quantità destinata al mercato.

Si noti infine che questo problema non coinvolge variabili intere ed è pertanto un modello di LP.

## 8.16 Problema del fixed charge

### Definizione del problema

- Produzione di un singolo prodotto

- esiste una variabile decisionale

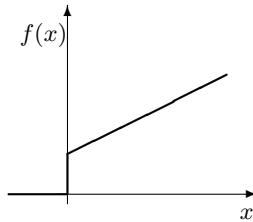
$x$  = numero di prodotti da realizzare

che permette di gestire facilmente il costo unitario;

- bisogna tener conto anche dell'eventuale costo fisso  $k$ ;

- la funzione di costo dipende dalla variabile  $x$  in modo *non lineare*

$$f(x) = \begin{cases} k + cx & \text{se } x > 0 \\ 0 & \text{se } x = 0 \end{cases}$$



### Modello PLI

- per derivare una formulazione MILP introduciamo una variabile aggiuntiva con il seguente significato

$$y := \begin{cases} 1 & \text{se } x > 0 \text{ (viene prodotta una quantità non nulla di prodotto)} \\ 0 & \text{altrimenti} \end{cases}$$

- **funzione obiettivo**  $\min k y + c x$

- **vincoli aggiuntivi**  $x \leq M y$   
 $y \in \{0, 1\}$   
dove  $M$  è un numero “grande a piacere”;
- il vincolo modella le relazioni

$$\begin{aligned}x > 0 &\Rightarrow y = 1 \\y = 0 &\Rightarrow x = 0\end{aligned}$$

### Esempio numerico

#### Considerazioni

- Il procedimento può essere esteso al caso in cui la stessa variabile binaria  $y$  sia associata alla produzione di  $n$  prodotti diversi;
- per ogni prodotto  $j$ , indichiamo con  $M_j$  il massimo numero di prodotti che possono essere realizzati ( $j = 1, \dots, n$ );
- **vincoli aggiuntivi**

$$\begin{aligned}x_j \leq M_j y &\quad j = 1, \dots, n \\y \in \{0, 1\}\end{aligned}$$

- il primo vincolo modella le relazioni

$$\begin{aligned}x_j > 0 \text{ per qualche } j \in \{1, \dots, n\} &\Rightarrow y = 1 \\y = 0 &\Rightarrow x_j = 0 \quad \forall j = 1, \dots, n\end{aligned}$$

## 8.17 Disattivazione di vincoli

- In alcune situazioni si vuole modellare la possibilità che un vincolo

$$\alpha^T x \geq b$$

sia attivo solo sotto certe condizioni;

- immaginiamo che questo debba accadere quando  $y = 0$ , dove  $y$  è una variabile binaria;
- in tal caso è possibile imporre l'implicazione logica

$$y = 0 \Rightarrow \alpha^T x \geq b$$

tramite i seguenti vincoli

$$\begin{aligned}\alpha^T x &\geq b - My \\y \in \{0, 1\}\end{aligned}$$

- Analogamente: l'implicazione  $y = 1 \Rightarrow \alpha^T x \geq b$  viene imposta dai vincoli  $\alpha^T x \geq b - M(1 - y)$  e  $y \in \{0, 1\}$
- Supponiamo di avere i due seguenti vincoli

$$\begin{aligned}\alpha_i^T x &\geq b_i \\ \alpha_k^T x &\geq b_k\end{aligned}$$

e di voler imporre che *almeno* uno dei due sia soddisfatto;

- introducendo due variabili binarie  $y_i$  e  $y_k$  è possibile riscrivere i vincoli nella seguente forma

$$\begin{aligned}\alpha_i^T x &\geq b_i - My_i \\ \alpha_k^T x &\geq b_k - My_k \\ y_i + y_k &\leq 1 \\ y_i, y_k &\in \{0, 1\}\end{aligned}$$

- imporre che la somma delle variabili  $y$  sia 1 corrisponde a permettere ad uno solo dei due vincoli di essere disattivato.

## Capitolo 9

# Esercizi: modelli di Programmazione Lineare Intera NEW: ((Ver. 6.0))

### Esercizio 9.15. Taglio di una barra.

Si consideri il problema di tagliare una barra di metallo in modo da ottenere dei pezzi più piccoli. La barra ha una lunghezza pari a  $W$ , mentre ogni pezzo richiesto ha una lunghezza pari a  $w_j$ . L'obiettivo è determinare un sottoinsieme di pezzi che possono essere ottenuti tagliando la barra, in modo da minimizzare la quantità di materiale che viene sprecata.

### Soluzione

**Variabili** Per determinare i pezzi che vogliamo ottenere, associamo una variabile binaria  $x_j$  ad ogni pezzo  $j$ , con seguente significato:

$$x_j := \begin{cases} 1 & \text{se il pezzo } j \text{ viene realizzato} \\ 0 & \text{altrimenti} \end{cases} \quad (j = 1, \dots, n)$$

**Modello di Programmazione Lineare Intera** Dato che minimizzare lo scarto è equivalente a massimizzare la parte utile, possiamo modellare il problema come un KP01 in cui ogni oggetto  $j$  ha un profitto pari alla sua lunghezza  $w_j$ , e massimizzare il profitto, ovvero la lunghezza dei pezzi ottenuti.

$$\max \sum_{j=1}^n w_j x_j \quad (9.1a)$$

$$\sum_{j=1}^n w_j x_j \leq W \quad (9.1b)$$

$$x_j \in \{0, 1\} \quad j = 1, \dots, n \quad (9.1c)$$

**Variante 1** Nel caso in cui i pezzi vengano tagliati da un insieme di lame parallele, modificare il modello in modo da gestire la limitazione derivante dal fatto che il numero di lame a disposizione è pari a  $R$ .

**Modifica al modello** Se ogni lama separa un pezzo dalla barra, è necessario aggiungere il vincolo:

$$\sum_{j=1}^n x_j \leq R$$

**Variante 2** Si consideri il caso in cui gli ordini dei clienti non specificano esattamente la lunghezza di ciascun pezzo. Ogni ordine  $j$  è associato ad un pezzo la cui lunghezza deve appartenere all'intervallo  $[\ell_j, u_j]$  (per ogni  $j = 1, \dots, n$ ). Per ciascun ordine  $j$  che viene accettato, la quantità prodotta deve stare nell'intervallo specificato.

**Modifica al modello** È necessario aggiungere le seguenti variabili continue che definiscono la lunghezza dei pezzi tagliati (e che valgono 0 per i pezzi non tagliati):

$$y_j := \text{lunghezza del pezzo } j \quad (j = 1, \dots, n)$$

e modificare il modello come segue:

$$\begin{aligned} \max \quad & \sum_{j=1}^n y_j \\ & \sum_{j=1}^n y_j \leq W \\ & \ell_j x_j \leq y_j \leq u_j x_j \quad j = 1, \dots, n \\ & x_j \in \{0, 1\} \quad j = 1, \dots, n \end{aligned}$$

## Esercizio 9.16. Assemblaggio elettrodomestici.

Un'azienda assembla  $n$  elettrodomestici  $\{1, \dots, n\}$ , il  $j$ -esimo con un prezzo di vendita pari a  $g_j$ , utilizzando un insieme di  $m$  componenti standard  $\{1, \dots, m\}$ , dell' $i$ -esimo dei quali sono disponibili  $b_i$  esemplari. Per assemblare un elettrodomestico di tipo  $j$  sono necessari  $a_{ij}$  esemplari di ciascun componente  $i$ .

Esigenze di mercato impongono che, per ciascun elettrodomestico  $j$ , il numero di unità prodotte sia almeno pari a  $r_j$  ed al massimo pari a  $t_j$ . Assumendo che tutti i dati del problema siano interi non negativi, si vuole determinare il massimo ricavo ottenibile assemblando i componenti disponibili.

### Soluzione

**Variabili** La scelta riguarda quanti elettrodomestici di ogni tipo realizzare (e non semplicemente se realizzare un elettrodomestico). Di conseguenza useremo delle variabili intere:

$$x_j := \text{numero di elettrodomestici di tipo } j \text{ realizzati} \quad (j = 1, \dots, n)$$

#### Modello di Programmazione Lineare Intera

$$\max \sum_j^n g_j x_j \tag{9.2a}$$

$$\sum_{j=1}^n a_{ij} x_j \leq b_i \quad i = 1, \dots, m \tag{9.2b}$$

$$r_j \leq x_j \leq t_j \quad \text{intera} \quad j = 1, \dots, n \tag{9.2c}$$

La funzione obiettivo (9.2a) moltiplica il prezzo di vendita di ogni elettrodomestico per la quantità realizzata, determinando il ricavo per elettrodomestico, e poi somma su tutti gli elettrodomestici, determinando il ricavo totale. I vincoli (9.2b) sono scritti per ogni componente  $i$ . Fissato  $i$ , sommano su tutti gli elettrodomestici il prodotto della numero di componenti  $i$  necessario per ottenere l'elettrodomestico  $j$  ( $a_{ij}$ ) per la quantità realizzata  $x_j$ , determinando il consumo totale del componente  $i$ , che deve essere minore o uguale alla quantità disponibile a magazzino. Infine, i vincoli (9.2c) impongono le quantità minime e massime da realizzare.

### Esercizio 9.17. Caricamento di camion.

Un'azienda di trasporto deve caricare i propri camion in modo da servire giornalmente un dato insieme di clienti. Nei camion devono essere caricati  $n$  pallet  $\{1, \dots, n\}$ , il  $j$ -esimo dei quali ha un peso  $p_j$  ed un volume  $v_j$ . Per ciascun camion la capacità in peso è pari a  $P$  e quella in volume pari a  $V$ ; inoltre vi è un costo fisso pari a  $c$  per l'utilizzo del camion.

L'azienda dispone di un numero  $m$  di camion sufficiente a caricare tutti i pallet e desidera minimizzare il costo complessivo dei camion utilizzati per caricare tutti i pallet. Si assuma che tutti i dati del problema siano interi non negativi.

### Soluzione

**Variabili** Definiamo due insiemi di variabili binarie. Il primo descrive come i pallet sono assegnati ai camion:

$$x_{ij} := \begin{cases} 1 & \text{se il pallet } j \text{ è caricato sul camion } i \\ 0 & \text{altrimenti} \end{cases} \quad (i = 1, \dots, m; j = 1, \dots, n)$$

mentre il secondo insieme di variabili indica quali camion sono utilizzati (e quindi pagati):

$$y_i := \begin{cases} 1 & \text{se il camion } i \text{ viene utilizzato} \\ 0 & \text{altrimenti} \end{cases} \quad (i = 1, \dots, m)$$

#### Modello di Programmazione Lineare Intera

$$\min c \sum_{i=1}^m y_i \tag{9.3a}$$

$$\sum_{i=1}^m x_{ij} = 1 \quad j = 1, \dots, n \tag{9.3b}$$

$$\sum_{j=1}^n p_j x_{ij} \leq P y_i \quad i = 1, \dots, m \tag{9.3c}$$

$$\sum_{j=1}^n v_j x_{ij} \leq V y_i \quad i = 1, \dots, m \tag{9.3d}$$

$$y_i \in \{0, 1\} \quad i = 1, \dots, m \tag{9.3e}$$

$$x_{ij} \in \{0, 1\} \quad i = 1, \dots, m; j = 1, \dots, n \tag{9.3f}$$

Differenze rispetto al Bin Packing:

- a ciascun camion è associato un costo  $c$  (uguale per tutti);
- vincoli di capacità sia in termini di volume che in termini di peso.

## Esercizio 9.18. Localizzazione ambulatori.

L’Azienda USL deve decidere riguardo all’apertura di nuovi ambulatori medici sul territorio per servire un insieme di  $m$  utenti  $i$ ,  $i = 1, \dots, m$ . Uno studio preliminare individua un insieme di  $n$  potenziali località. Ogni utente  $i$  può essere servito da un ambulatorio nella località  $j$ ,  $j = 1, \dots, n$  se questa non è troppo distante dal domicilio dell’utente; è quindi data una matrice binaria  $A$  il cui generico elemento  $a_{ij}$  ha valore 1 se l’utente  $i$  può essere servito nella località  $j$  e 0 altrimenti. Il costo di apertura di un ambulatorio nella località  $j$  è pari a  $c_j$  e tale ambulatorio è in grado di servire al più  $k_j$  utenti. Per ragioni di programmazione, l’Azienda USL vuole comunque aprire almeno  $P$  ambulatori, servendo tutti gli utenti e minimizzando il costo complessivo.

### Soluzione

**Variabili** Come nell’esercizio precedente, abbiamo un primo insieme di variabili binarie che assegnano gli utenti agli ambulatori:

$$x_{ij} := \begin{cases} 1 & \text{se l’utente } i \text{ viene assegnato all’ambulatorio } j \\ 0 & \text{altrimenti} \end{cases} \quad (i = 1, \dots, m; j = 1, \dots, n)$$

e un secondo insieme di variabili che definiscono quali ambulatori sono aperti:

$$y_j := \begin{cases} 1 & \text{se l’ambulatorio } j \text{ viene aperto} \\ 0 & \text{altrimenti} \end{cases} \quad (j = 1, \dots, n)$$

#### Modello di Programmazione Lineare Intera

$$\min \sum_{j=1}^n c_j y_j \tag{9.4a}$$

$$x_{ij} \leq a_{ij} \quad i = 1, \dots, m; j = 1, \dots, n \tag{9.4b}$$

$$\sum_{j=1}^n x_{ij} = 1 \quad i = 1, \dots, m \tag{9.4c}$$

$$\sum_{i=1}^m x_{ij} \leq k_j y_j \quad j = 1, \dots, n \tag{9.4d}$$

$$\sum_{j=1}^n y_j \geq P \tag{9.4e}$$

$$y_j \in \{0, 1\} \quad j = 1, \dots, n \tag{9.4f}$$

$$x_{ij} \in \{0, 1\} \quad i = 1, \dots, m; j = 1, \dots, n \tag{9.4g}$$

Differenze rispetto al Bin Packing:

- non tutte le potenziali località possono servire tutti gli utenti, i vincoli **(9.4b)** impediscono di assegnare un utente a una località che non lo può servire;
- devono essere aperti almeno  $P$  ambulatori;
- ogni ambulatorio ha una capacità specifica;
- il numero di clienti  $m$  è diverso dal numero di potenziali ambulatori  $n$ .

### Esercizio 9.19. Assegnamento di lavori.

Un'azienda ha a disposizione  $m$  macchinari con cui deve effettuare  $n$  lavorazioni. Se il macchinario  $i$  ( $i = 1, \dots, m$ ) effettua la lavorazione  $j$  ( $j = 1, \dots, n$ ) viene pagato un costo positivo  $c_{ij}$ , in particolare si ha  $c_{ij} = +\infty$  qualora il macchinario non possa effettuare la lavorazione. Ogni lavorazione  $j$  ha una durata positiva  $t_j$  ( $j = 1, \dots, n$ ). L'obiettivo dell'azienda è quello di minimizzare la spesa complessiva, facendo in modo che tutte le lavorazioni siano eseguite e che ogni macchinario sia impegnato per un tempo massimo complessivo TMAX.

### Soluzione

#### Variabili

$$x_{ij} := \begin{cases} 1 & \text{se la lavorazione } j \text{ viene assegnata al macchinario } i \\ 0 & \text{altrimenti} \end{cases} \quad (i = 1, \dots, m; j = 1, \dots, n)$$

#### Modello di Programmazione Lineare Intera

$$\min \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \quad (9.5a)$$

$$\sum_{j=1}^n t_j x_{ij} \leq TMAX \quad i = 1, \dots, m \quad (9.5b)$$

$$\sum_{i=1}^m x_{ij} = 1 \quad j = 1, \dots, n \quad (9.5c)$$

$$x_{ij} \in \{0, 1\} \quad i = 1, \dots, m; j = 1, \dots, n \quad (9.5d)$$

Si noti che una lavorazione potrebbe essere assegnata ad una macchina che non può farla, ma questa scelta avrebbe un costo  $+\infty$  e quindi non sarà mai effettuata se esiste una alternativa *ammissibile* (cioè di costo finito).

Differenze rispetto al Generalized Assignment Problem:

- il tempo di lavorazione non dipende dalla macchina;
- il tempo massimo complessivo non dipende dalla macchina.

## Esercizio 9.20. Assegnamento di lavori 2.

Un'industria ha a disposizione  $m$  macchinari con cui deve lavorare  $n$  pezzi meccanici. È nota una matrice binaria  $A$  tale che  $a_{ij} = 1$  se il macchinario  $i$  può lavorare il pezzo  $j$  e  $a_{ij} = 0$  altrimenti ( $i = 1, \dots, m; j = 1, \dots, n$ ). Se il macchinario  $i$  ( $i = 1, \dots, m$ ) lavora il pezzo  $j$  ( $j = 1, \dots, n$ ), viene imputato un costo industriale positivo  $c_{ij}$ . Ogni pezzo  $j$  ha un tempo di lavorazione positivo  $l_j$  ( $j = 1, \dots, n$ ) e ogni macchinario  $i$ , a causa di vincoli sindacali legati all'operatore del macchinario stesso, può lavorare al massimo per un tempo (dato dalla somma delle durate delle lavorazioni dei singoli pezzi ad esso assegnati)  $M_i$  ( $i = 1, \dots, m$ ). L'obiettivo dell'azienda è quello di minimizzare il costo industriale complessivo, facendo in modo che ogni pezzo meccanico sia lavorato esattamente da un macchinario e che vengano rispettati i vincoli sulle durate massime dei tempi di lavorazione dei macchinari.

### Soluzione

#### Variabili

$$x_{ij} := \begin{cases} 1 & \text{se il pezzo } j \text{ viene assegnato al macchinario } i \\ 0 & \text{altrimenti} \end{cases} \quad (i = 1, \dots, m; j = 1, \dots, n)$$

#### Modello di Programmazione Lineare Intera

$$\min \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \quad (9.6a)$$

$$\sum_{j=1}^n l_j x_{ij} \leq M_i \quad i = 1, \dots, m \quad (9.6b)$$

$$x_{ij} \leq a_{ij} \quad i = 1, \dots, m; j = 1, \dots, n \quad (9.6c)$$

$$\sum_{i=1}^m x_{ij} = 1 \quad j = 1, \dots, n \quad (9.6d)$$

$$x_{ij} \in \{0, 1\} \quad i = 1, \dots, m; j = 1, \dots, n \quad (9.6e)$$

Differenze rispetto al Generalized Assignment Problem:

- il tempo di lavorazione non dipende dalla macchina;
- non tutti i macchinari possono lavorare tutti i tipi di prodotto (vincoli (9.6c)).

## Esercizio 9.21. Localizzazione magazzini.

Un'azienda di trasporti vuole decidere dove costruire i propri magazzini al fine di servire un certo insieme  $1, \dots, m$  di clienti. A tale scopo, è stato individuato un insieme  $\{1, \dots, n\}$  di postazioni nelle quali sarebbe possibile posizionare un magazzino. Ciascuna postazione  $j$  ( $j = 1, \dots, n$ ) ha un costo fisso di attivazione pari a  $f_j$ , può servire al massimo  $k_j$  clienti e permette di servire il generico cliente  $i$  ( $i = 1, \dots, m$ ) con un costo di servizio pari a  $c_{ij}$ . Obiettivo dell'azienda è stabilire dove posizionare i propri magazzini e determinare a quale magazzino assegnare ciascun cliente in modo da minimizzare il costo complessivo.

## Soluzione

### Variabili

$$x_{ij} := \begin{cases} 1 & \text{se il cliente } i \text{ viene assegnato alla postazione } j \\ 0 & \text{altrimenti} \end{cases} \quad (i = 1, \dots, m; j = 1, \dots, n)$$

$$y_j := \begin{cases} 1 & \text{se la postazione } j \text{ viene utilizzata per un magazzino} \\ 0 & \text{altrimenti} \end{cases} \quad (j = 1, \dots, n)$$

### Modello di Programmazione Lineare Intera

$$\min \sum_{j=1}^n f_j y_j + \sum_{j=1}^n \sum_{i=1}^m c_{ij} x_{ij} \quad (9.7a)$$

$$\sum_{i=1}^m x_{ij} \leq k_j y_j \quad j = 1, \dots, n \quad (9.7b)$$

$$\sum_{j=1}^n x_{ij} = 1 \quad i = 1, \dots, m \quad (9.7c)$$

$$x_{ij} \in \{0, 1\} \quad i = 1, \dots, m; j = 1, \dots, n \quad (9.7d)$$

$$y_j \in \{0, 1\} \quad j = 1, \dots, n \quad (9.7e)$$

Differenze rispetto al Facility Location:

- la “quantità di servizio” richiesta dai clienti è sempre uguale a 1; quindi i coefficienti delle variabili  $x_{ij}$  nei vincoli di capacità (9.7b) sono tutti uguali a 1.

## Esercizio 9.22. Test software.

Una azienda di informatica deve effettuare dei test prima di rilasciare le applicazioni software sviluppate. In particolare, sono date  $n$  applicazioni software  $\{1, \dots, n\}$  e  $k$  computer  $\{1, \dots, k\}$ . Per avere garanzie di qualità, ogni applicazione  $j$  deve essere testata con diversi sistemi operativi e dunque deve essere eseguita su  $s_j$  computer distinti. Il tempo di esecuzione dell'applicazione  $j$  sul computer  $c$  è pari a  $t_{jc}$  ed ogni computer  $c$  può effettuare test per un tempo massimo  $T_c$ . Inoltre, è noto un profitto  $p_{jc}$  che si ottiene se l'applicazione software  $j$  viene testata sul computer  $c$ . L'obiettivo dell'azienda è di assegnare le applicazioni software ai computer, in modo tale che ogni applicazione  $j$  sia testata su  $s_j$  computer, ogni computer esegua test per un tempo massimo  $T_c$  e il profitto complessivo derivante dall'assegnazione sia massimizzato.

## Soluzione

### Variabili

$$x_{jc} := \begin{cases} 1 & \text{se l'applicazione } j \text{ viene eseguita sul computer } c \\ 0 & \text{altrimenti} \end{cases} \quad (j = 1, \dots, n; c = 1, \dots, k)$$

### Modello di Programmazione Lineare Intera

$$\max \sum_{c=1}^k \sum_{j=1}^n p_{jc} x_{jc} \quad (9.8a)$$

$$\sum_{j=1}^n t_{jc} x_{jc} \leq T_c \quad c = 1, \dots, k \quad (9.8b)$$

$$\sum_{c=1}^k x_{jc} = s_j \quad j = 1, \dots, n \quad (9.8c)$$

$$x_{jc} \in \{0, 1\} \quad c = 1, \dots, k; j = 1, \dots, n \quad (9.8d)$$

Differenze rispetto all'Assegnamento Generalizzato:

- ogni applicazione  $j$  deve essere eseguita  $s_j$  volte;
- ad ogni assegnamento viene associato un profitto e la funzione obiettivo è da massimizzare.

### Esercizio 9.23. Mix di produzione.

Il management di un impianto produttivo deve definire il mix di produzione per il prossimo anno. L'impianto dispone di  $m$  macchinari del tipo FMS (flexible manufacturing system) che, grazie alla loro flessibilità, possono eseguire lavorazioni su ognuno degli  $n$  prodotti del portafoglio aziendale. I macchinari in questione sono eterogenei, per cui il tempo necessario per eseguire la lavorazione di un'unità di prodotto del tipo  $j$  sulla macchina  $i$  è pari a  $t_{ij}$ . Ogni macchinario ha un orario di lavoro regolamentare pari a  $T_i$  cui può essere sommato un massimo di  $Over_i$  unità di tempo che verranno considerate come straordinario. La programmazione deve indicare quanti prodotti realizzare per ciascun tipo di prodotto  $j$  e come assegnare le unità di prodotto ai macchinari. Si consideri che

- è consentito produrre lo stesso tipo di prodotto su macchinari differenti;
- per ragioni di marketing, è opportuno che siano prodotte almeno  $k_j$  unità di ciascun tipo di prodotto  $j$ ;
- ogni unità di prodotto  $j$  ha un profitto pari a  $p_j$ .

Ipotizzando che tutti i prodotti realizzati saranno venduti l'obiettivo del management è la massimizzazione del profitto complessivo opportunamente nettato dei costi dovuti allo straordinario.

### Soluzione

#### Variabili

$x_{ij} :=$  numero di prodotti di tipo  $j$  assegnati alla macchina  $i$  ( $i = 1, \dots, m; j = 1, \dots, n$ )

$y_i :=$  unità di tempo di lavoro straordinario sulla macchina  $i$  ( $i = 1, \dots, m$ )

#### Modello di Programmazione Lineare Intera

$$\max \sum_{j=1}^n p_j \sum_{i=1}^m x_{ij} - \sum_{i=1}^m c_i y_i \quad (9.9a)$$

$$\sum_{j=1}^n t_{ij} x_{ij} \leq T_i + y_i \quad i = 1, \dots, m \quad (9.9b)$$

$$y_i \leq Over_i \quad i = 1, \dots, m \quad (9.9c)$$

$$\sum_{i=1}^m x_{ij} \geq k_j \quad j = 1, \dots, n \quad (9.9d)$$

$$x_{ij} \geq 0 \quad \text{intera} \quad i = 1, \dots, m; j = 1, \dots, n \quad (9.9e)$$

$$y_i \geq 0 \quad \text{intera} \quad i = 1, \dots, m \quad (9.9f)$$

Differenze rispetto all'Assegnamento Generalizzato:

- ad ogni prodotto è associato un profitto;
- ogni prodotto deve essere lavorato in  $k_j$  unità → variabili intere e non binarie;
- possibilità di lavorare in straordinario.

### Esercizio 9.24. Posizionamento prodotti.

In un supermercato si vogliono disporre dei prodotti (scelti tra un insieme di  $n$  prodotti) su  $m$  scaffali.

Ogni prodotto  $j$  ( $j = 1, \dots, n$ ) ha un peso  $w_j$  e può esser disposto su al massimo uno scaffale  $i$  ( $i = 1, \dots, m$ ).

Ogni scaffale  $i$  ha una capacità in peso  $W_i$ . Se si utilizza uno scaffale  $i$  si paga un costo  $c_i$ . Se il prodotto  $j$  viene disposto sullo scaffale  $i$  si ottiene un profitto  $p_{ij}$ .

Si deve determinare quali prodotti disporre sugli scaffali, in modo da massimizzare i guadagni (dati dalla differenza tra profitti e costi), rispettando i vincoli di capacità degli scaffali.

### Soluzione

#### Variabili

$$x_{ij} := \begin{cases} 1 & \text{se il prodotto } j \text{ viene collocato sullo scaffale } i \\ 0 & \text{altrimenti} \end{cases} \quad (i = 1, \dots, m; j = 1, \dots, n)$$

$$y_i := \begin{cases} 1 & \text{se lo scaffale } i \text{ viene utilizzato} \\ 0 & \text{altrimenti} \end{cases} \quad (i = 1, \dots, m)$$

#### Modello di Programmazione Lineare Intera

$$\max \sum_{i=1}^m \sum_{j=1}^n p_{ij} x_{ij} - \sum_{i=1}^m c_i y_i \quad (9.10a)$$

$$\sum_{j=1}^n w_j x_{ij} \leq W_i y_i \quad i = 1, \dots, m \quad (9.10b)$$

$$\sum_{i=1}^m x_{ij} \leq 1 \quad j = 1, \dots, n \quad (9.10c)$$

$$x_{ij} \in \{0, 1\} \quad i = 1, \dots, m; j = 1, \dots, n \quad (9.10d)$$

$$y_j \in \{0, 1\} \quad j = 1, \dots, n \quad (9.10e)$$

### Esercizio 9.25. Assegnamento posti barca.

Il direttore di un porto turistico italiano deve definire come assegnare i posti barca in base alle richieste di prenotazione pervenutegli per la stagione estiva. Il porto è strutturato in  $m$  aree distinte, ogni area  $i$  ( $i = 1, \dots, m$ ) è caratterizzata dalla lunghezza massima delle barche che possono attraccarvi (ogni barca attraccata non può essere lunga più di  $L_i$ ) e dal numero di posti barca disponibili  $D_i$ . L'attivazione di un'area per la stagione estiva comporta un costo pari a  $c_i$  dovuto ai costi di manutenzione ed aggiornamento.

Le richieste pervenute al porto sono relative ad  $n$  imbarcazioni. Per ogni imbarcazione  $j$  è nota la rispettiva lunghezza  $l_j$  ed il prezzo che deve essere pagato per l'affitto del posto barca  $p_j$ , definito dal direttore in base alle caratteristiche della stessa (lunghezza, fabbisogno energetico, prestigio). Ogni imbarcazione può essere assegnata al più ad un'area.

L'obiettivo della pianificazione è definire quali aree del porto attivare e quali richieste assegnare a ciascuna di esse al fine di massimizzare il ricavo del porto, opportunamente nettato dai costi di attivazione, nel rispetto dei vincoli connessi alle aree.

## Soluzione

### Variabili

$$x_{ij} := \begin{cases} 1 & \text{se l'area } i \text{ è assegnata alla barca } j \\ 0 & \text{altrimenti} \end{cases} \quad (i = 1, \dots, m; j = 1, \dots, n)$$

$$y_i := \begin{cases} 1 & \text{se l'area } i \text{ è attivata} \\ 0 & \text{altrimenti} \end{cases} \quad (i = 1, \dots, m)$$

### Modello di Programmazione Lineare Intera

$$\max \sum_{j=1}^n p_j \sum_{i=1}^m x_{ij} - \sum_{i=1}^m c_i y_i \quad (9.11a)$$

$$\sum_{i=1}^m x_{ij} \leq 1 \quad j = 1, \dots, n \quad (9.11b)$$

$$\sum_{j=1}^n x_{ij} \leq D_i \quad i = 1, \dots, m \quad (9.11c)$$

$$l_j x_{ij} \leq L_i y_i \quad i = 1, \dots, m; j = 1, \dots, n \quad (9.11d)$$

$$x_{ij} \in \{0, 1\} \quad i = 1, \dots, m; j = 1, \dots, n \quad (9.11e)$$

$$y_i \in \{0, 1\} \quad i = 1, \dots, m \quad (9.11f)$$

Differenze rispetto al Facility Location:

- non tutte le richieste devono essere necessariamente accettate;
- vincolo sul numero massimo di barche assegnate ad un'area;
- vincolo sulla lunghezza massima delle barche attraccate in un'area.

### Esercizio 9.26. Localizzazione vigili del fuoco.

Il Ministero dell'Interno deve decidere riguardo all'apertura di nuove stazioni dei Vigili del Fuoco al fine di servire un insieme di  $q$  quartieri. Uno studio preliminare ha individuato un insieme di  $s$  potenziali strade dove localizzare una stazione. Ogni quartiere  $i$  ( $i = 1, \dots, q$ ) può essere servito da una stazione nella strada  $j$  ( $j = 1, \dots, s$ ) se questa non è troppo distante dal quartiere stesso: è quindi data una matrice binaria  $A$  il cui generico elemento  $a_{ij}$  ha valore 1 se il quartiere  $i$  può essere servito da una stazione posta nella strada  $j$  e 0 altrimenti. Il costo di apertura di una stazione nella strada  $j$  è pari a  $r_j$ . Per ragioni di sicurezza, ogni quartiere deve poter essere servito da almeno 2 stazioni distinte; inoltre devono essere aperte complessivamente almeno  $B$  stazioni.

### Soluzione

#### Variabili

$$y_j := \begin{cases} 1 & \text{se la stazione } j \text{ viene aperta} \\ 0 & \text{altrimenti} \end{cases} \quad (j = 1, \dots, s)$$

#### Modello di Programmazione Lineare Intera

$$\min \sum_{j=1}^s r_j y_j \quad (9.12a)$$

$$\sum_{j=1}^s a_{ij} y_j \geq 2 \quad i = 1, \dots, q \quad (9.12b)$$

$$\sum_{j=1}^s y_j \geq B \quad (9.12c)$$

$$y_j \in \{0, 1\} \quad j = 1, \dots, s \quad (9.12d)$$

## Esercizio 9.27.

(In blu le variazioni rispetto all'esercizio 12)

Il Ministero dell'Interno deve decidere riguardo all'apertura all'assegnamento di nuove stazioni dei Vigili del Fuoco al fine di servire a un insieme di  $q$  quartieri. Uno studio preliminare ha individuato un insieme di  $s$  potenziali strade dove localizzare una stazione. Ogni quartiere  $i$  ( $i = 1, \dots, q$ ) può essere servito da una stazione nella strada  $j$  ( $j = 1, \dots, s$ ) se questa non è troppo distante dal quartiere stesso: è quindi data una matrice binaria  $A$  il cui generico elemento  $a_{ij}$  ha valore 1 se il quartiere  $i$  può essere servito da una stazione posta nella strada  $j$  e 0 altrimenti.

Il costo di apertura di una stazione nella strada  $j$  è pari a  $r_j$ .

Il costo relativo al servizio del quartiere  $i$  da una stazione aperta nella strada  $j$  è pari a  $c_{ij}$ .

Per ragioni di sicurezza, ogni quartiere deve poter essere servito da almeno 2 stazioni distinte; inoltre devono essere aperte complessivamente almeno  $B$  stazioni. una stazione aperta nella generica strada  $j$  può servire al massimo  $b_j$  quartieri.

## Soluzione

### Variabili

$$x_{ij} := \begin{cases} 1 & \text{se il quartiere } i \text{ viene servito dalla stazione in } j \\ 0 & \text{altrimenti} \end{cases} \quad (i = 1, \dots, q; j = 1, \dots, s)$$

### Modello di Programmazione Lineare Intera

$$\min \sum_{j=1}^s \sum_{i=1}^q c_{ij} x_{ij} \quad (9.13a)$$

$$\sum_{i=1}^q x_{ij} \leq b_j \quad j = 1, \dots, s \quad (9.13b)$$

$$\sum_{j=1}^s a_{ij} x_{ij} \geq 2 \quad i = 1, \dots, q \quad (9.13c)$$

$$x_{ij} \in \{0, 1\} \quad i = 1, \dots, q; j = 1, \dots, s \quad (9.13d)$$

## Esercizio 9.28.

(In rosso le variazioni rispetto all'esercizio 13)

Il Ministero dell'Interno deve decidere riguardo all'apertura di nuove stazioni dei Vigili del Fuoco ed al loro assegnamento ad un insieme di  $q$  quartieri. Uno studio preliminare ha individuato un insieme di  $s$  potenziali strade dove localizzare una stazione. Ogni quartiere  $i$  ( $i = 1, \dots, q$ ) può essere servito da una stazione nella strada  $j$  ( $j = 1, \dots, s$ ) se questa non è troppo distante dal quartiere stesso: è quindi data una matrice binaria  $A$  il cui generico elemento  $a_{ij}$  ha valore 1 se il quartiere  $i$  può essere servito da una stazione posta nella strada  $j$  e 0 altrimenti. Il costo di apertura di una stazione nella strada  $j$  è pari a  $r_j$  ed il costo relativo al servizio del quartiere  $i$  da una stazione aperta nella strada  $j$  è pari a  $c_{ij}$ . Per ragioni di sicurezza, ogni quartiere deve essere servito da almeno 2 stazioni distinte; inoltre devono essere aperte complessivamente almeno  $B$  stazioni ed una stazione aperta nella generica strada  $j$  può servire al massimo  $b_j$  quartieri.

## Soluzione

### Variabili

$$x_{ij} := \begin{cases} 1 & \text{se il quartiere } i \text{ viene servito dalla stazione in } j \\ 0 & \text{altrimenti} \end{cases} \quad (i = 1, \dots, q; j = 1, \dots, s)$$

$$y_j := \begin{cases} 1 & \text{se la stazione } j \text{ viene aperta} \\ 0 & \text{altrimenti} \end{cases} \quad (j = 1, \dots, s)$$

### Modello di Programmazione Lineare Intera

$$\min \sum_{j=1}^s \sum_{i=1}^q c_{ij} x_{ij} + \sum_{j=1}^s r_j y_j \quad (9.14a)$$

$$\sum_{i=1}^q x_{ij} \leq b_j y_j \quad j = 1, \dots, s \quad (9.14b)$$

$$\sum_{j=1}^s a_{ij} x_{ij} \geq 2 \quad i = 1, \dots, q \quad (9.14c)$$

$$\sum_{j=1}^s y_j \geq B \quad (9.14d)$$

$$x_{ij} \in \{0, 1\} \quad i = 1, \dots, q; j = 1, \dots, s \quad (9.14e)$$

$$y_j \in \{0, 1\} \quad j = 1, \dots, s \quad (9.14f)$$

### Esercizio 9.29. Assegnamento lavori.

Una azienda deve lavorare  $n$  diversi prodotti su  $m$  macchine. La lavorazione di un generico prodotto  $j$  in una generica macchina  $i$  richiede un tempo di lavorazione pari a  $p_{ij}$ , se eseguito in modalità “standard”. Ogni lavoro (di un prodotto su una macchina) può però essere eseguito in modalità “crash”: in questo caso si stima un costo pari a  $c_j$  e il tempo di lavorazione è ridotto della metà rispetto a quello “standard”.

Le macchine sono raggruppate in  $K$  cluster contenenti insiemi disgiunti di macchine; nel generico cluster  $l$  è contenuto l’insieme di macchine  $C_l$ . Viene fornita una matrice  $A$  in cui  $a_{jl} = 1$  se il prodotto  $j$  deve essere lavorato in esattamente una delle macchine contenute nel cluster  $l$  e  $a_{jl} = 0$  se non è necessario eseguire nessuna lavorazione per il prodotto  $j$  in nessuna delle macchine in  $C_l$ . Di conseguenza, ogni prodotto potrà essere lavorato su più macchine appartenenti a cluster diversi.

L’obiettivo è quello di definire in quali macchine conviene lavorare ciascun prodotto in modo da minimizzare il *makespan* complessivo, tenendo conto del fatto che è possibile eseguire le lavorazioni in modalità “crash” fino ad un budget di costo massimo pari a  $B$ .

## Soluzione

### Variabili

$$x_{ij} := \begin{cases} 1 & \text{se il lavoro } j \text{ viene eseguito dalla macchina } i \text{ in modalità standard} \\ 0 & \text{altrimenti} \end{cases} \quad (i = 1, \dots, m; j = 1, \dots, n)$$

$$y_{ij} := \begin{cases} 1 & \text{se il lavoro } j \text{ viene eseguito dalla macchina } i \text{ in modalità “crash”} \\ 0 & \text{altrimenti} \end{cases} \quad (i = 1, \dots, m; j = 1, \dots, n)$$

$$z := \text{makespan}$$

### Modello di Programmazione Lineare Intera

$$\min z \quad (9.15a)$$

$$\sum_{j=1}^n p_{ij} x_{ij} + \sum_{j=1}^n \frac{p_{ij}}{2} y_{ij} \leq z \quad i = 1, \dots, m \quad (9.15b)$$

$$\sum_{i \in C_l} (x_{ij} + y_{ij}) = a_{jl} \quad l = 1, \dots, K; j = 1, \dots, n \quad (9.15c)$$

$$\sum_{j=1}^n \sum_{i=1}^m c_j y_{ij} \leq B \quad (9.15d)$$

$$x_{ij} \in \{0, 1\} \quad i = 1, \dots, m; j = 1, \dots, n \quad (9.15e)$$

$$y_{ij} \in \{0, 1\} \quad i = 1, \dots, m; j = 1, \dots, n \quad (9.15f)$$

### Esercizio 9.30. Piano lavorazione.

Una azienda deve definire il piano di produzione per i prossimi  $m$  mesi relativamente a un determinato prodotto. Per un generico mese  $i$  si ha un domanda pari a  $d_i$ , un costo di stoccaggio in magazzino pari a  $r_i$ , ed un costo unitario di produzione pari a  $c_i$ .

Inoltre, se durante il mese in questione viene lavorata almeno una unità di prodotto, si applicano dei costi fissi pari a  $K_i$ .

Considerando che il magazzino può stoccare al massimo  $D$  unità di prodotto, si vuole individuare il piano di produzione che minimizzi il costo complessivo.

## Soluzione

### Variabili

$$x_i := \text{quantità lavorata nel mese } i (i = 1, \dots, m)$$

$$s_i := \text{quantità stoccati nel mese } i (i = 1, \dots, m)$$

$$y_i := \begin{cases} 1 & \text{se il prodotto viene lavorato nel mese } i \\ 0 & \text{altrimenti} \end{cases} \quad (i = 1, \dots, m)$$

### Modello di Programmazione Lineare Intera

$$\min \sum_{i=1}^m K_i y_i + \sum_{i=1}^m c_i x_i + \sum_{i=1}^m r_i s_i \quad (9.16a)$$

$$s_i \leq D \quad i = 1, \dots, m \quad (9.16b)$$

$$x_i + s_{i-1} - s_i = d_i \quad i = 1, \dots, m \quad (9.16c)$$

$$x_i \leq M y_i \quad i = 1, \dots, m \quad (9.16d)$$

$$x_i \geq 0 \quad \text{intera} \quad i = 1, \dots, m \quad (9.16e)$$

$$s_i \geq 0 \quad \text{intera} \quad i = 1, \dots, m \quad (9.16f)$$

$$y_i \in \{0, 1\} \quad i = 1, \dots, m \quad (9.16g)$$

### Esercizio 9.31. Flusso massimo.

Si vuole stabilire il flusso massimo di gas erogabile tramite una certa rete di condutture, rappresentata da un grafo  $G = (V, A)$ , a partire da una sorgente  $s \in V$  e con destinazione  $t \in V$ . Ogni arco  $(i, j) \in A$  rappresenta un tratto della condutture ed ha una capacità massima pari a  $q_{ij}$ . Si consideri anche la limitazione derivante dal fatto che ogni nodo  $i \in V$  è in grado di gestire una quantità massima di flusso pari a  $b_i$ .

### Soluzione

#### Variabili

$x_{ij} :=$  quantità di flusso lungo l'arco  $(i, j) ((i, j) \in A)$

$z :=$  valore del flusso

#### Modello di Programmazione Lineare Intera

$$\max z \quad (9.17a)$$

$$\sum_{j:(i,j) \in A} x_{ij} - \sum_{k:(k,i) \in A} x_{ki} = \begin{cases} +z & \text{if } i = s \\ -z & \text{if } i = t \\ 0 & \text{if } i \in V \setminus \{s, t\} \end{cases} \quad (9.17b)$$

$$\sum_{j:(i,j) \in A} x_{ij} \leq b_i \quad i \in V \quad (9.17c)$$

$$x_{ij} \in \{0, 1\} \quad (i, j) \in A \quad (9.17d)$$

### Esercizio 9.32. Assegnamento con conflitti.

Un negozio di animali domestici ha a disposizione  $n$  vasche d'acqua per esporre in vetrina un insieme di  $m$  pesci tropicali. Ogni vasca  $j$  ha un costo pari a  $c_j$  ed un volume  $V_j$ .

È noto un grafo  $G = (V, A)$  in cui ciascun vertice corrisponde ad un pesce, ed il generico arco  $a_{ih}$  rappresenta l'impossibilità per il pesce  $i$  di essere nella stessa vasca del pesce  $h$  (per ragioni legate alla catena alimentare...).

E' inoltre nota una matrice binaria  $B$  in cui il generico elemento  $b_{ij}$  è uguale a 1 se il pesce  $i$  può essere assegnato alla vasca  $j$  e 0 altrimenti.

Infine, ogni pesce  $i$  necessita per vivere uno spazio stimato pari a  $v_i$ ; la somma dello spazio necessario ai pesci contenuti in ciascuna vasca  $j$  non deve superare la capacità  $V_j$  della vasca.

Si vuole minimizzare il costo totale delle vasche utilizzate per l'esposizione considerando che, per ragioni di spazio, il negozio può al massimo esporre  $K$  vasche.

### Soluzione

#### Variabili

$$x_{ij} := \begin{cases} 1 & \text{se il pesce } i \text{ è assegnato alla vasca } j \\ 0 & \text{altrimenti} \end{cases} \quad (i = 1, \dots, m; j = 1, \dots, n)$$

$$y_j := \begin{cases} 1 & \text{se la vasca } j \text{ è utilizzata} \\ 0 & \text{altrimenti} \end{cases} \quad (j = 1, \dots, n)$$

#### Modello di Programmazione Lineare Intera

$$\min \sum_{j=1}^n c_j y_j \tag{9.18a}$$

$$\sum_{j=1}^n y_j \leq K \tag{9.18b}$$

$$\sum_{j=1}^n b_{ij} x_{ij} = 1 \quad i = 1, \dots, m \tag{9.18c}$$

$$\sum_{i=1}^m v_i x_{ij} \leq V_j y_j \quad j = 1, \dots, n \tag{9.18d}$$

$$x_{ij} + x_{hj} \leq y_j \quad (i, h) \in A; j = 1, \dots, n \tag{9.18e}$$

$$x_{ij} \in \{0, 1\} \quad i = 1, \dots, m; j = 1, \dots, n \tag{9.18f}$$

$$y_j \in \{0, 1\} \quad j = 1, \dots, n \tag{9.18g}$$

# Capitolo 10

## Algoritmi per PLI

### 10.1 Introduzione

Per la soluzione esatta dei problemi PLI sono stati messi a punto numerosi algoritmi. In questo capitolo saranno esaminati i due approcci storicamente più importanti lasciando a corsi più avanzati la descrizione di tecniche più moderne. In particolare saranno esaminati l'algoritmo basato sui piani di taglio (noto come algoritmo cutting plane) proposto da Ralph Gomory e l'algoritmo branch-and-bound proposto da Alisa Land and Alison Doig.

### 10.2 Algoritmo Cutting Plane NEW: (Ver. 6.0)

L'algoritmo cutting plane, proposto da Gomory [Gom58CP], è una tecnica generale per la risoluzione di problemi PLI. L'algoritmo si basa sulla risoluzione di una serie di rilassamenti continui ciascuno associato ad una formulazione valida del problema PLI ottenuta aggiungendo un vincolo alla formulazione originaria che la renda migliore rispetto alla precedente.

Sia  $P = \{\min c^T x : Ax = d, x \geq 0, x \text{ intere}\}$  il problema PLI da risolvere e sia  $C(P)$  il corrispondente rilassamento continuo di  $P$ , ottenuto rimuovendo il vincolo di interezza, la cui soluzione ottima è  $x^C$ . Se  $x^C$  è intera allora è la soluzione ottima di  $P$  e l'algoritmo termina, altrimenti tale soluzione deve essere eliminata dalla regione ammissibile di  $P$  mediante l'aggiunta di un opportuno vincolo.

**Definizione 10.1** (cutting plane). *Dato un problema PLI  $P$  e la soluzione  $x^C$  del corrispondente rilassamento continuo  $C(P)$ , una disequazione  $\alpha x \geq \alpha_0$  è un piano di taglio (cutting plane) valido se:*

- a)  $x^C$  viola tale disequazione, ossia  $\alpha x^C < \alpha_0$ , e
- b) è soddisfatta da tutti i punti interi in  $P$ , ossia  $\alpha x \geq \alpha_0 \forall x \in P$

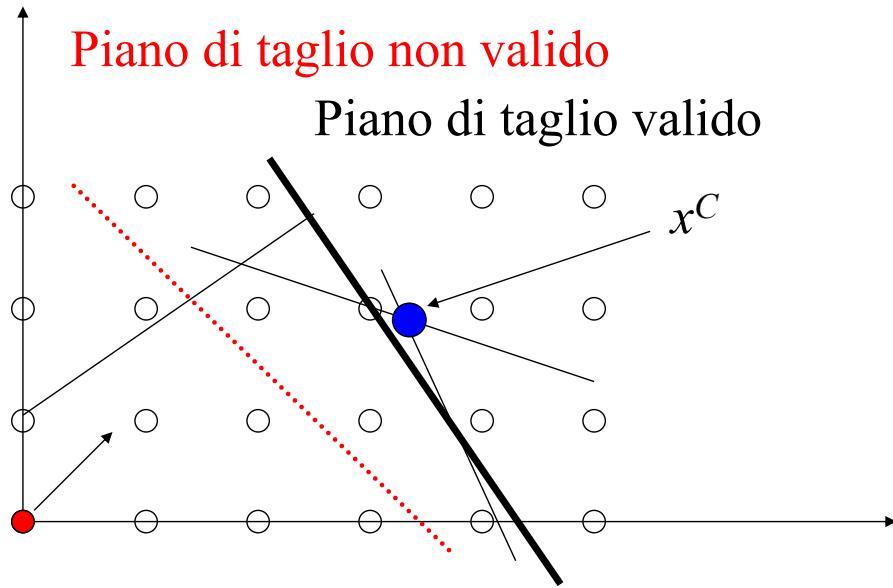


Figura 10.1: Esempio di piano di taglio.

La figura 10.1 illustra un piano di taglio valido ed uno non valido per un problema  $P$ .

Aggiungendo alla formulazione corrente un piano di taglio si ottiene una nuova formulazione valida per  $P$  che domina quella corrente dato che la regione ammissibile della nuova formulazione è contenuta nella precedente. L'algoritmo procede nell'aggiunta di piani di taglio alla formulazione finché non si ottiene una formulazione il cui rilassamento continuo ha soluzione intera.

---

**Algorithm 4** Algoritmo Cutting Plane.

---

```

1: procedure CUTTINGPLANE( $(c, A, d)$ )
2:   risolvi il rilassamento continuo  $C(P)$  ottenendo  $x^C$ ;
3:   while  $x^C$  non è intero do
4:     determina un cutting plane  $\alpha x \geq \alpha_0$  ed aggiungilo a  $P$ ;
5:     risolvi il rilassamento continuo  $C(P)$  ottenendo  $x^C$ ;
6:     if  $C(P)$  impossibile then
7:       stop
8:     end if
9:   end while
10:  return  $x^* = x^C$  (soluzione ottima)
11: end procedure

```

---

L'algoritmo cutting plane determina la soluzione ottima del problema  $P$  dato che ad ogni iterazione:

- $C(P)$  contiene tutte e sole le soluzioni intere di  $P$  (più altre non intere);
- I tagli aggiunti non eliminano soluzioni intere;
- $C(P)$  e  $P$  hanno la stessa funzione obiettivo.

Il procedimento è molto efficace quando la soluzione ottima del problema  $C(P)$  non è troppo diversa da quella del problema  $P$ . Gli svantaggi dell'algoritmo sono però rappresentati dal fatto che il numero di iterazioni del ciclo while necessarie alla determinazione della soluzione intera nel caso peggiore non è polinomiale nella dimensione del problema  $P$ . Inoltre, ad ogni iterazione il rilassamento continuo da risolvere diventa sempre più grande per effetto dell'aggiunta del piano di taglio e questo può portare a tempi di calcolo elevati per la soluzione.

Il problema principale nell'implementazione dell'algoritmo cutting plane è rappresentato dalla definizione in modo automatico del piano di taglio da aggiungere a ciascuna iterazione. A questo scopo può essere utilizzata il procedimento proposto da Gomory per definire, direttamente dalla soluzione del rilassamento continuo corrente, un piano di taglio basato sull'arrotondamento dei coefficienti ottenibili dal tableau ottimo.

Sia  $Y$  il tableau ottimo del rilassamento continuo corrente, corrispondente alla base ottima  $\mathcal{B}$  che comprende le variabili  $x_{\beta(1)}, \dots, x_{\beta(m)}$ . Sia  $i$  una riga in cui sia in base una variabile a valore frazionario. Dall'espressione del sistema di vincoli in forma canonica rispetto alla base si ha che

$$y_{i0} = x_{\beta(i)} + \sum_{A_j \notin \mathcal{B}} y_{ij} x_j. \quad (10.1)$$

Poiché in qualunque soluzione ammissibile di  $P$  si ha che  $x \geq 0$  sostituendo in (10.1) a  $y_{ij}$  il corrispondente arrotondamento verso il basso  $\lfloor y_{ij} \rfloor$  si ha che

$$x_{\beta(i)} + \sum_{A_j \notin \mathcal{B}} \lfloor y_{ij} \rfloor x_j \leq y_{i0} \quad (10.2)$$

Tale disequazione è valida per ogni soluzione frazionaria in  $P$ . Se consideriamo però le sole soluzioni intere in  $P$  il primo membro della disequazione avrà valore intero e possiamo quindi arrotondare verso il basso anche il secondo membro, ottenendo l'espressione del taglio di Gomory:

$$x_{\beta(i)} + \sum_{A_j \notin \mathcal{B}} \lfloor y_{ij} \rfloor x_j \leq \lfloor y_{i0} \rfloor \quad (10.3)$$

È facile verificare che l'espressione (10.3) è violata dalla soluzione frazionaria corrente dato che  $x_{\beta(i)} = y_{i0} < \lfloor y_{i0} \rfloor$  ed è quindi un piano di taglio valido che può essere aggiunto al problema corrente.

### 10.3 L'Algoritmo Branch and Bound

L'algoritmo branch-and-bound, dovuto a Land e Doig [**LandD60BB**], è una tecnica generale per la risoluzione di problemi di ottimizzazione combinatoria,

ossia problemi di ottimizzazione la cui regione ammissibile sia un insieme  $F$  finito. Il metodo si basa sulla scomposizione del problema da risolvere in sottoproblemi più semplici adottoando quindi il principio “Divide and Conquer”.

L'algoritmo è in grado di risolvere problemi molto generali in cui sia la regione ammissibile sia la funzione obiettivo sono nonlineari ma al fine di descriverlo ci limiteremo al caso di problemi PLI. A tal fine indichiamo il problema PLI da risolvere con  $P^0 = (z(\cdot), F(P^0))$  dove  $z(\cdot)$  indica la funzione obiettivo lineare del problema e  $F(P^0)$  indica la regione ammissibile del problema, nel nostro caso definita da vincoli lineari. La soluzione ottima del problema è indicata come  $z^* = z(P^0) = \min\{z(x) : x \in F(P^0)\}$ . Inoltre durante l'esecuzione dell'algoritmo indichiamo con  $z^{Best}$  la miglior soluzione ammissibile nota per il problema, osservando che al termine dell'esecuzione si avrà  $z^* = z^{Best}$ .

Dato il problema  $P^0$  da risolvere, che per semplicità considereremo come problema di minimizzazione, il metodo branch-and-boundlo suddivide in  $K$  sottoproblemi  $P^1, \dots, P^K$  tali che la loro totalità rappresenti  $P^0$ . Tale suddivisione può essere ad esempio operata suddividendo la regione ammissibile  $F(P^0)$  in sottoinsiemi  $F(P^1), \dots, F(P^K)$  tali che

$$\bigcup_{i=1}^K F(P^i) = F(P^0).$$

Normalmente tali sottoinsiemi costituiscono una partizione di  $F(P^0)$ , ossia  $F(P^i) \cap F(P^j) = \emptyset \forall i \neq j, i, j = 1, \dots, K$ , ma ciò non è strettamente necessario.

Il processo di suddivisione è spesso denominato di “ramificazione” (branching) e può essere iterato ricorsivamente suddividendo nello stesso modo i problemi via via generati. L'insieme dei problemi generati dal processo di suddivisione può essere efficacemente rappresentato graficamente da un albero decisionale (branch-decision tree), di cui un esempio è dato in figura 10.2. Nell'albero decisionale i nodi rappresentano i sottoproblemi generati e gli archi rappresentano le relazioni di discendenza tra i sottoproblemi. Il primo nodo dell'albero rappresenta il problema originale e viene generalmente chiamato *nodo radice* mentre tenendo conto delle relazioni di discendenza i nodi possono essere nodi *padre* (o genitore) da cui discendono nodi *figli*.

Data la suddivisione di  $P^0$  nei  $K$  sottoproblemi è evidente che la risoluzione di  $P^0$  può essere ottenuta dalla risoluzione di tutti i sottoproblemi generati. In particolare definendo come  $z(P^i) = \min\{z(x) : x \in F(P^i)\}$  si ha che

$$z(P^0) = \min\{z(P^1), \dots, z(P^K)\}.$$

Un generico sottoproblema  $P^i$  è *risolto* se:

- a) si può dimostrare che il sottoproblema è *impossibile*, ossia  $F(P^i) = \emptyset$ ;
- b) si è ottenuta la soluzione ottima  $z(P^i)$ , ad esempio nel caso di un problema PLI risolvendo il rilassamento continuo  $C(P^i)$  si ha che  $z(C^i)$  è intera;
- c) si può dimostrare che risolvendo  $P^i$  non si migliorerebbe la miglior soluzione nota  $z^{Best}$ . Ad esempio nel caso di un problema PLI se  $z(C^i) \geq z^{Best}$  dato che in questo caso anche  $z(P^i) \geq z^{Best}$ .

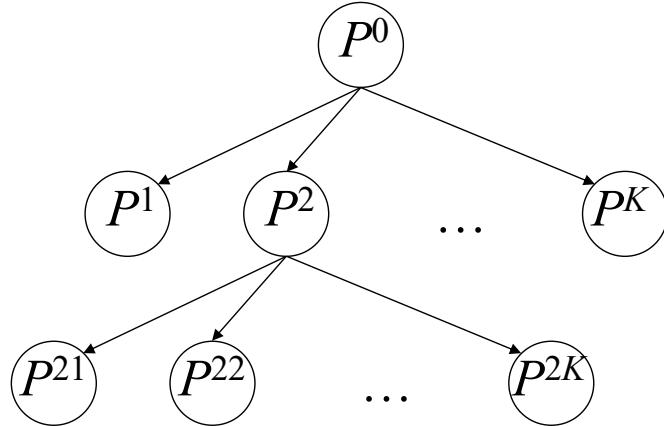


Figura 10.2: Esempio di albero decisionale

Se un problema viene risolto, la ramificazione di tale problema viene interrotta e nel caso b) viene eventualmente anche aggiornato il valore di  $z^{Best}$ . Se il problema non è risolto invece deve essere ulteriormente suddiviso in sottoproblemi. Il problema  $P^0$  è risolto quando tutti i sottoproblemi via via generati sono stati risolti ricadendo in uno dei tre casi sopraelencati. Al termine dell'esecuzione il valore della soluzione ottima è  $z^{Best}$ .

Vediamo ora in dettaglio una semplice implementazione dell'algoritmo branch-and-bound nel caso dei problemi PLI. In particolare facciamo riferimento ad un problema  $P^0 = \min\{c^T x \in \mathbb{R}^n : Ax = d, x \geq 0, \text{ intere}\}$ , ed indichiamo con  $x^k$  la soluzione ottima di un generico sottoproblema  $P^k$  di valore  $z^k$ . Inoltre,  $x^{Ck}$  indicherà la soluzione ottima del rilassamento continuo del problema  $P^k$ , di valore  $z^{Ck}$ . Si noti che per le proprietà dei rilassamenti continui descritte nel capitolo precedente si ha che

$$z^{Ck} = c^T x^{Ck} \leq z^k = c^T x^k.$$

Dato il problema originale  $P^0$  se ne risolve il rilassamento continuo  $C(P^0)$ , se la soluzione di tale rilassamento è intera allora è la soluzione ottima di  $P^0$  e ci si può fermare. Altrimenti tale soluzione contiene almeno una componente  $x_j^{C0}$  frazionaria che può essere utilizzata per la suddivisione di  $P^0$  in sottoproblemi.

In generale la suddivisione di un problema in sottoproblemi può essere effettuata suddividendone la regione ammissibile in sottoinsiemi mediante l'aggiunta di opportuni vincoli al problema stesso come illustrato in figura 10.3. In tale figura un sottoproblema è ottenuto aggiungendo al problema  $P^0$  la cui regione ammissibile è un triangolo, un vincolo del tipo  $\alpha^k x \leq \delta^k$ .

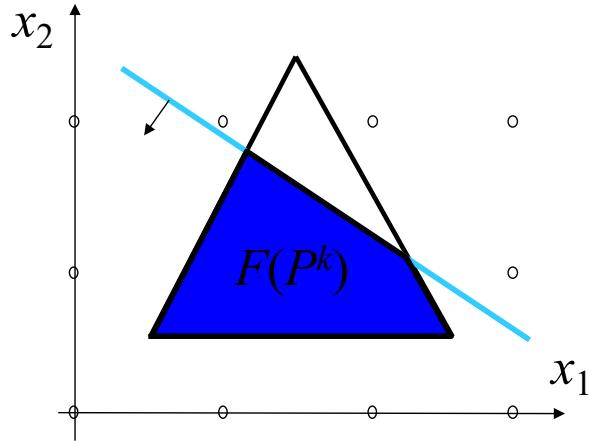


Figura 10.3: Esempio di suddivisione della regione ammissibile di un problema mediante l'aggiunta di un vincolo.

In una semplice implementazione del branch-and-bound per PLI si può ottenere una suddivisione di un problema in due sottoproblemi, detta branching binario, operando nel seguente modo. Scelta una componente  $x_j^{C_0}$  frazionaria, imponiamo due condizioni mutuamente esclusive ed esaustive, valide per ogni soluzione intera di  $P^0$ :

$$x_j \leq \lfloor x_j^{C_0} \rfloor \quad \text{e} \quad x_j \geq \lfloor x_j^{C_0} \rfloor + 1.$$

È facile verificare che tutti i punti interi contenuti in  $F(P^0)$  soddisfano una di queste disequazioni mentre la soluzione del rilassamento continuo non soddisfa nessuna di tali relazioni. Il branching binario descritto è illustrato graficamente nella figura 10.4 in cui sono indicate le regioni ammissibili dei due problemi associati al branching sulla variabile frazionaria  $x_j^{C_0}$ . Si noti che la soluzione del rilassamento continuo non soddisfa nessuno dei due vincoli imposti.

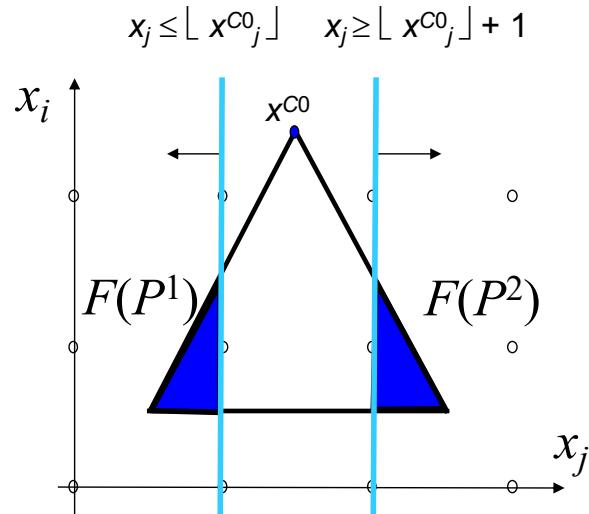


Figura 10.4: Esempio di branching binario basato una componente frazionaria della soluzione del rilassamento continuo.

Dalla suddivisione operata nel modo precedentemente descritto si originano due problemi:

- $P^1 = \min\{c^T x \in \mathbb{R}^n : Ax = d, x_j \leq \lfloor x_j^{C0} \rfloor, x \geq 0, \text{ intere}\}$ , e
- $P^2 = \min\{c^T x \in \mathbb{R}^n : Ax = d, x_j \geq \lfloor x_j^{C0} \rfloor + 1, x \geq 0, \text{ intere}\}$ .

per i quali sarà possibile calcolare il rilassamento continuo e decidere se questi devono essere eliminati o ulteriormente suddivisi.

Per illustrare meglio il procedimento consideriamo il seguente esempio:

$$-\min -z = -x_1 - x_2 \quad (10.4)$$

$$\text{s.t.} \quad 5x_1 + 3x_2 \leq 15 \quad (10.5)$$

$$5x_1 - 3x_2 \geq 0 \quad (10.6)$$

$$x_2 \geq \frac{1}{2} \quad (10.7)$$

$$x_1, x_2 \geq 0 \quad (10.8)$$

la cui regione ammissibile è illustrata in figura 10.5 nella quale è anche indicata la soluzione ottima del rilassamento continuo del nodo radice corrispondente al problema  $P^0$  in cui le variabili decisionali valgono  $x^{C0} = (\frac{3}{2}, \frac{5}{2})$ . Il valore di tale rilassamento è  $-z = -4$  e rappresenta un lower bound sul valore della soluzione ottima  $z^* = z^0$  di  $P^0$ .

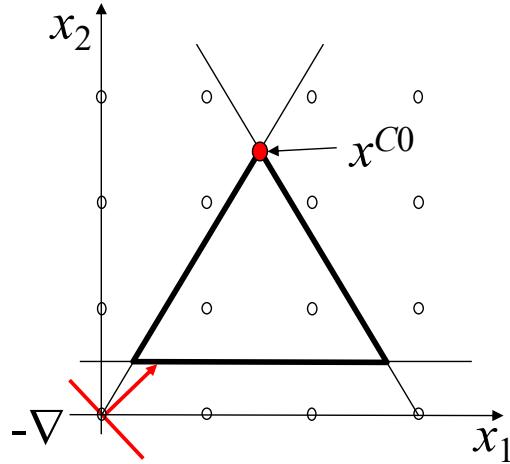


Figura 10.5: Regione ammissibile e soluzione del rilassamento continuo del nodo radice.

**Branching da  $P^0$ :** All'inizio dell'algoritmo non è nota nessuna soluzione intera del problema per cui il valore di  $-z^{Best}$  può essere inizializzato ad un valore convenzionale che garantisca di aggiornarlo non appena si trovi una soluzione intera. In questo caso poniamo  $-z^{Best} = +\infty$ .

La soluzione del rilassamento continuo del nodo radice è frazionaria  $x^{C0} = (\frac{3}{2}, \frac{5}{2})$  e vale  $-z = -4$ , quindi nella migliore delle ipotesi  $-z^* = -4$ . Poichè nessuna delle tre condizioni a)-c) elencate precedentemente ci permette di considerare il problema come risolto dobbiamo effettuare una suddivisione del problema (branching).

**Scelta della variabile per il branching.** Esistono molti criteri per la scelta della variabile frazionaria da utilizzare per il branching. Ad esempio, questa può essere scelta casualmente o può essere selezionata quella la cui parte frazionaria è maggiormente prossima ad  $\frac{1}{2}$ . Nel seguito useremo una semplice regola deterministica in base alla quale sceglieremo la variabile frazionaria di indice minimo. Nel nostro caso sceglieremo quindi la variabile  $x_1 = \frac{3}{2}$  e generiamo i due sottoproblemi:

- $P^1 = P^0 + x_1 \leq 1,$
- $P^2 = P^0 + x_1 \geq 2,$

In figura 10.6 sono rappresentate le regioni ammissibili dei due problemi generati e sono indicate le soluzioni ottime dei rilassamenti continui corrispondenti. I valori dei bound calcolati dai rilassamenti continui sono riportati in figura 10.7.

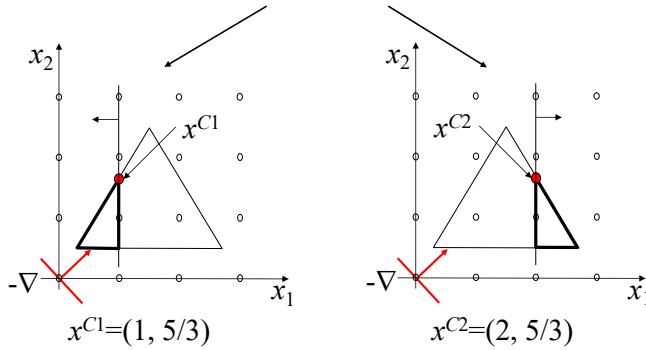


Figura 10.6: Regione ammissibile e soluzione del rilassamento continuo dei due sottoproblemi generati.

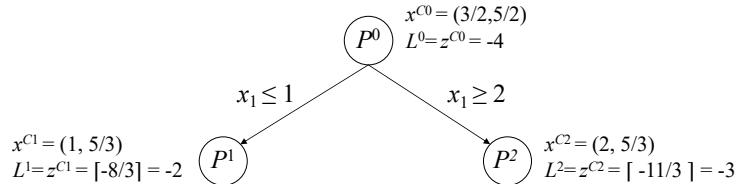


Figura 10.7: Albero decisionale dopo il primo branching.

**Possibile arrotondamento dei bound:** Qualora la funzione obiettivo sia caratterizzata da coefficienti interi, è evidente che il valore di qualsiasi soluzione intera avrà a sua volta valore intero. In generale la soluzione del rilassamento continuo  $x^C$  è frazionaria e di conseguenza anche il valore  $z^C$  di tale rilassamento potrà avere valore frazionario. È noto che, nel caso di problemi di minimo,  $z^C$  approssima per difetto il valore della soluzione ottima del problema intero  $z^*$ , ma se il valore di  $z^*$  è sicuramente intero allora anche l'arrotondamento all'intero superiore di  $z^C$  è un lower bound valido. Analogamente per problemi di massimo gli upper bound frazionari possono essere arrotondati all'intero inferiore. Pertanto se i coefficienti della funzione obiettivo sono tutti interi:

- per problemi di minimo i lower bound possono essere arrotondati per eccesso:  $L = \lceil z^C \rceil$ ;
- per problemi di massimo gli upper bound possono essere arrotondati per difetto:  $U = \lfloor z^C \rfloor$ .

Al termine del primo branching abbiamo quindi generato due sottoproblemi:

- $P^1$  la cui soluzione del rilassamento continuo è frazionaria  $x^{C1} = (1, \frac{5}{3})$  a cui corrisponde un lower bound  $L^1 = \lceil -\frac{8}{3} \rceil = -2$ ;

- $P^2$  la cui soluzione del rilassamento continuo è frazionaria  $x^{C2} = (2, \frac{5}{3})$  a cui corrisponde un lower bound  $L^2 = \lceil -\frac{11}{3} \rceil = -3$ .

Entrambi questi sottoproblemi non sono risolti e quindi dovremo effettuare il branching da uno di essi.

**Scelta del sottoproblema (strategia di esplorazione):** Ad una iterazione generica dell'algoritmo branch-and-bound possono essere “attivi” diversi sottoproblemi che sono stati generati e che non sono stati risolti e quindi devono essere ulteriormente suddivisi. Anche per la strategia di esplorazione esistono diverse possibilità ma ci limiteremo qui ad esporre una regola piuttosto efficace nota come “best bound first”.

Se consideriamo la situazione attuale dell'esempio i due problemi attivi sono  $P^1$  a cui è associato un lower bound pari a  $-2$  e  $P^2$  a cui è associato un lower bound pari a  $-3$ . È quindi chiaro che se risolvessimo all'ottimo  $P^1$ , sviluppando l'intero sottoalbero originato da tale nodo, la miglior soluzione intera che potremmo trovare varrebbe al più  $-2$ . Viceversa, risolvendo all'ottimo  $P^2$  potremmo trovare una soluzione che vale  $-3$  e quindi migliore. La regola da adottare nel scegliere il prossimo problema da cui effettuare il branching è

**Definizione 10.2** ((Regola di Branching)). *Dato l'insieme dei sottoproblemi attivi  $\mathcal{A}$  si sceglie per il branching il sottoproblema con il bound migliore. Nel caso di problemi di minimizzazione questo è il sottoproblema con il lower bound di valore minimo. Nel caso di problemi di massimizzazione è il sottoproblema con l'upper bound di valore massimo.*

**Secondo branching.** Seguendo la regola best bound first viene scelto per il branching il sottoproblema  $P^2$ . La soluzione del rilassamento continuo  $x^{C2} = (2, \frac{5}{3})$  ha una sola variabile frazionaria che quindi verrà usata per il branching generando due ulteriori sottoproblemi:

- $P^3 = P^2 + x_2 \leq 1, = P^0 + x_1 \geq 2 + x_2 \leq 1,$
- $P^4 = P^2 + x_2 \geq 2, = P^0 + x_1 \geq 2 + x_2 \leq 2,$

In figura 10.8 sono rappresentate le regioni ammissibili dei due problemi generati e sono indicate le soluzioni ottime dei rilassamenti continui corrispondenti. I valori dei bound calcolati dai rilassamenti continui sono riportati in figura 10.9.

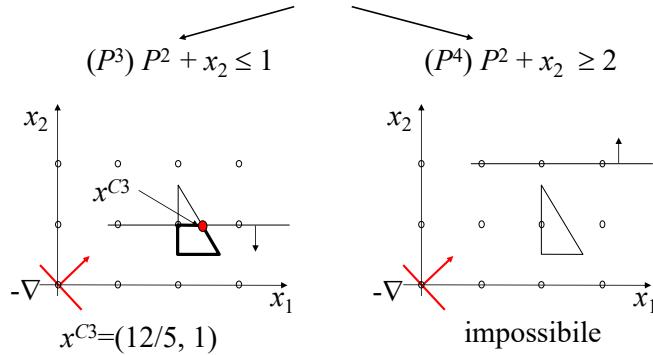


Figura 10.8: Regione ammissibile e soluzione del rilassamento continuo dei due sottoproblemi generati.

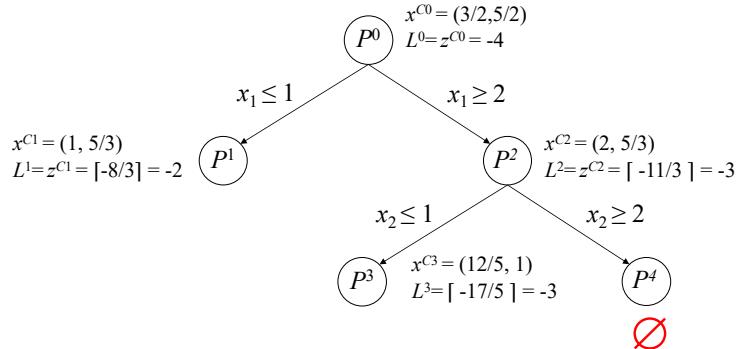


Figura 10.9: Albero decisionale dopo il secondo branching.

Al termine del secondo branching abbiamo quindi generato due sottoproblemi:

- $P^3$  la cui soluzione del rilassamento continuo è frazionaria  $x^{C3} = (\frac{12}{5}, 1)$  a cui corrisponde un lower bound  $L^3 = \lceil -\frac{17}{5} \rceil = -3$ ;
- $P^4$  la cui soluzione del rilassamento continuo è impossibile e quindi tale problema non andrà ulteriormente sviluppato.

Al termine del secondo branching sono ancora attivi i problemi  $P^1$  e  $P^3$  e quindi dovremo effettuare il branching da uno di essi.

**Terzo branching.** Seguendo la regola best bound first viene scelto per il branching il sottoproblema  $P^3$ . La soluzione del rilassamento continuo  $x^{C3} = (\frac{12}{5}, 1)$  ha una sola variabile frazionaria che quindi verrà usata per il branching generando due ulteriori sottoproblemi:

- $P^5 = P^3 + x_1 \leq 2, = P^0 + x_1 \geq 2 + x_2 \leq 1 + x_1 \leq 2,$

- $P^6 = P^3 + x_1 \geq 3, = P^0 + x_1 \geq 2 + x_2 \leq 1 + x_1 \geq 3,$

In figura 10.10 sono rappresentate le regioni ammissibili dei due problemi generati e sono indicate le soluzioni ottime dei rilassamenti continui corrispondenti. I valori dei bound calcolati dai rilassamenti continui sono riportati in figura 10.11.

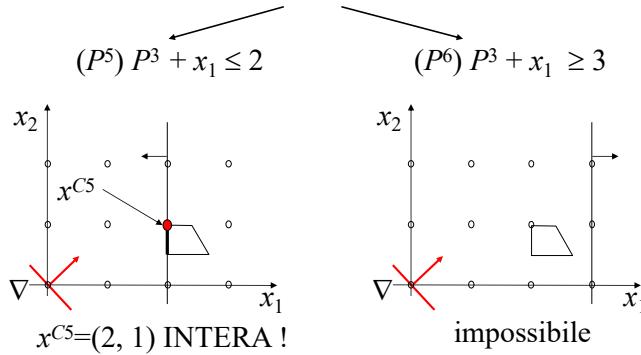


Figura 10.10: Regione ammissibile e soluzione del rilassamento continuo dei due sottoproblemi generati.

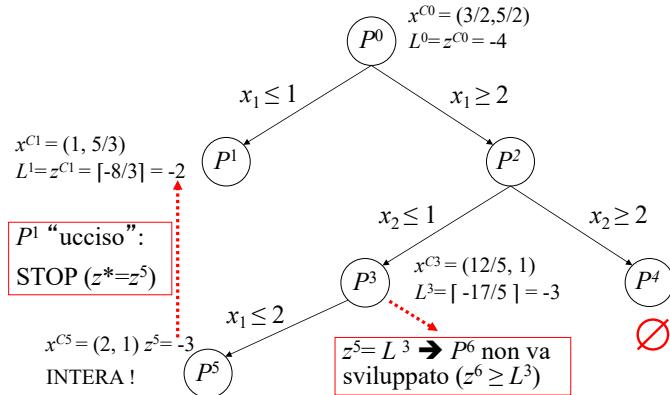


Figura 10.11: Albero decisionale dopo il terzo branching.

Al termine del terzo branching abbiamo quindi generato due sottoproblemi:

- $P^5$  la cui soluzione del rilassamento continuo è intera  $x^{C5} = (2, 1)$  a cui corrisponde una soluzione  $z^5 = -3$  che ci permette di aggiornare  $z^{Best} = z^5 = -3$  (e salvare la soluzione corrente  $x^{Best} = x^5 = (2, 1)$ );
- $P^6$  la cui soluzione del rilassamento continuo è impossibile e quindi tale problema non andrà ulteriormente sviluppato.

Al termine del terzo branching è ancora attivo il solo sottoproblema  $P^1$  che però può essere eliminato poiché  $L^1 \geq z^{Best}$  ed è quindi “ucciso” dal bound. Non essendovi più nodi attivi il problema è risolto e la soluzione ottima è la soluzione  $x^{Best} = (2, 1)$  di valore  $-3$  corrispondente a  $z^{Best}$ .

A commento dell'ultimo branching si può infine notare che il problema  $P^6$  poteva anche non essere esaminato dato che il bound del problema padre (ossia  $P^3$ ) era pari a  $-3$  e sviluppando il nodo fratello  $P^5$  abbiamo trovato una soluzione intera di tale valore. Pertanto anche sviluppando  $P^6$  non avremmo potuto trovare una soluzione migliore di quella già trovata.



## Capitolo 11

# Esercizi di Programmazione Lineare Intera

### Esercizio 11.1.

Si consideri il problema di programmazione lineare intera

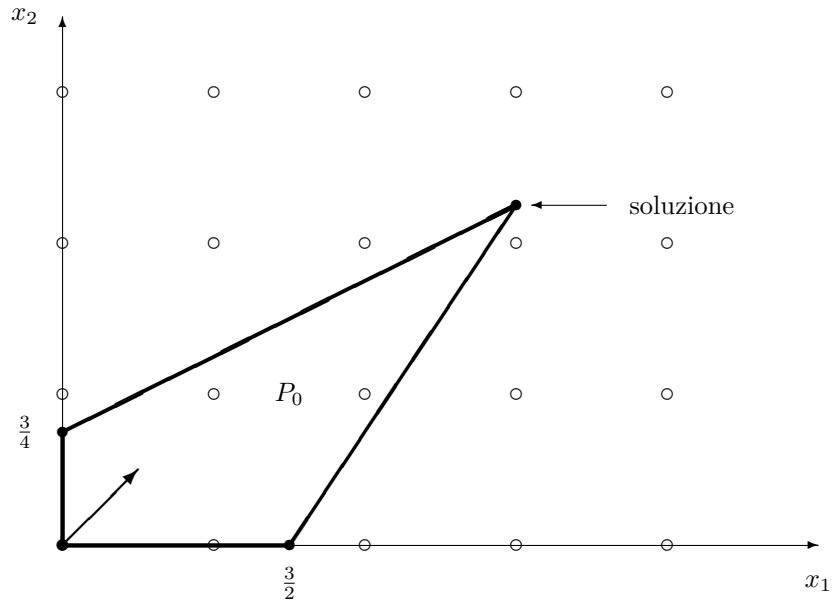
$$\begin{aligned} \max z = & \quad x_1 + x_2 \\ & - 6x_1 + 12x_2 \leq 9 \\ & 6x_1 - 4x_2 \leq 9 \\ & x_1, x_2 \geq 0, \text{ interi} \end{aligned}$$

- Risolvere mediante branch-and-bound, determinando per via grafica le varie soluzioni dei rilassamenti continui. Si dia l'albero decisionale, disegnando il politopo associato a ciascun nodo.
- Eseguire la prima iterazione dell'algoritmo di Gomory, determinando e disegnando almeno un piano di taglio.

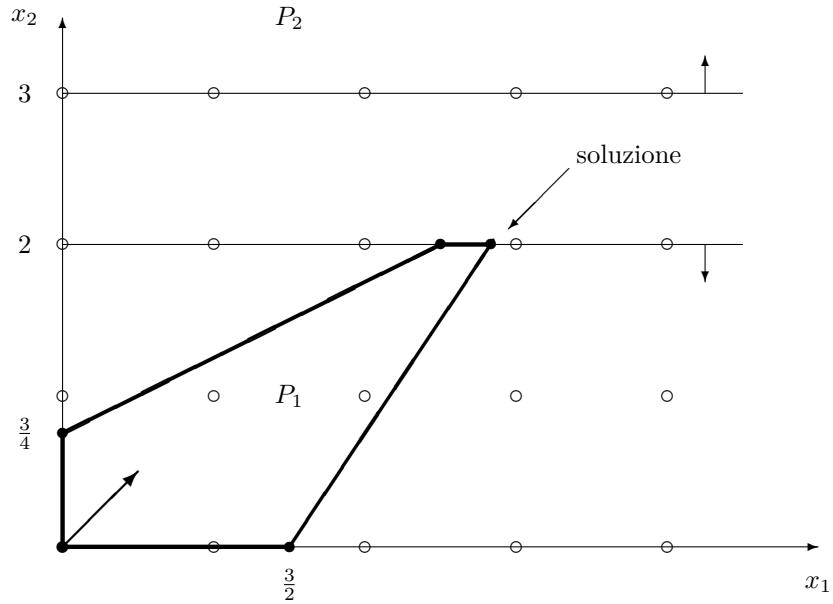
### Soluzione

- La regione ammissibile del problema è la seguente:

230 CAPITOLO 11. ESERCIZI DI PROGRAMMAZIONE LINEARE INTERA

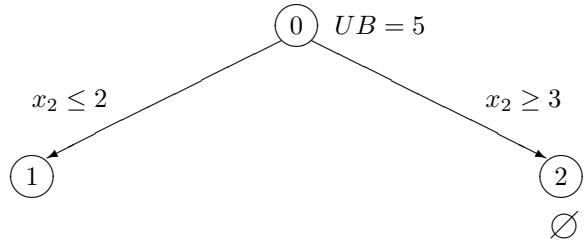


La soluzione ottima del rilassamento continuo è  $x_1 = 3$ ,  $x_2 = \frac{9}{4}$  ed ha valore  $\frac{21}{4}$ . Pertanto l'upper bound corrispondente vale  $\lfloor \frac{21}{4} \rfloor = 5$ . La variabile  $x_2$  ha valore non intero compreso tra 2 e 3. La ramificazione corrispondente divide il politopo mediante intersezione con i semipiani  $x_2 \leq 2$  ed  $x_2 \geq 3$ . Si ottengono quindi i politopi  $P_1$  e  $P_2$  di figura.

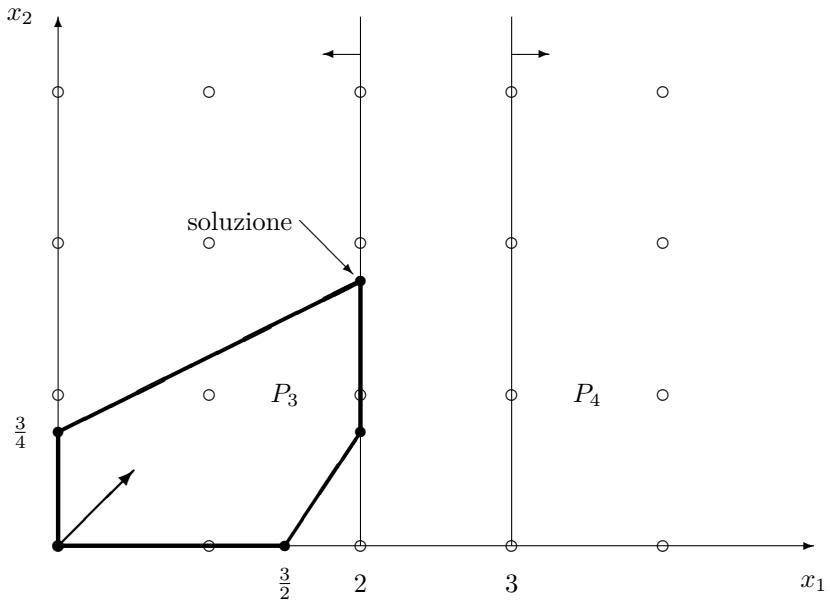


L'albero decisionale attuale è il seguente (al nodo  $i$  corrisponde il politopo

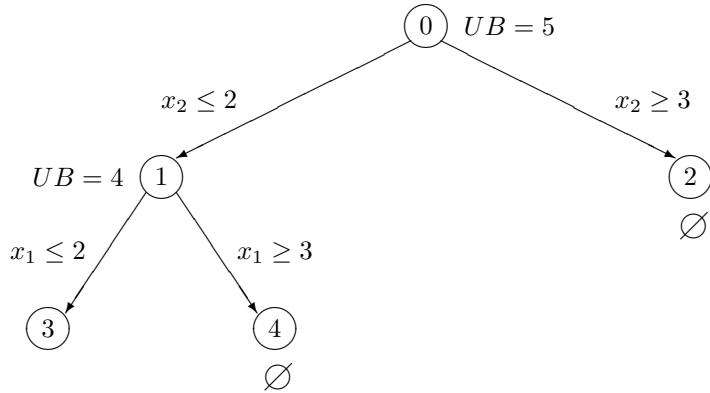
$P_i)$ :



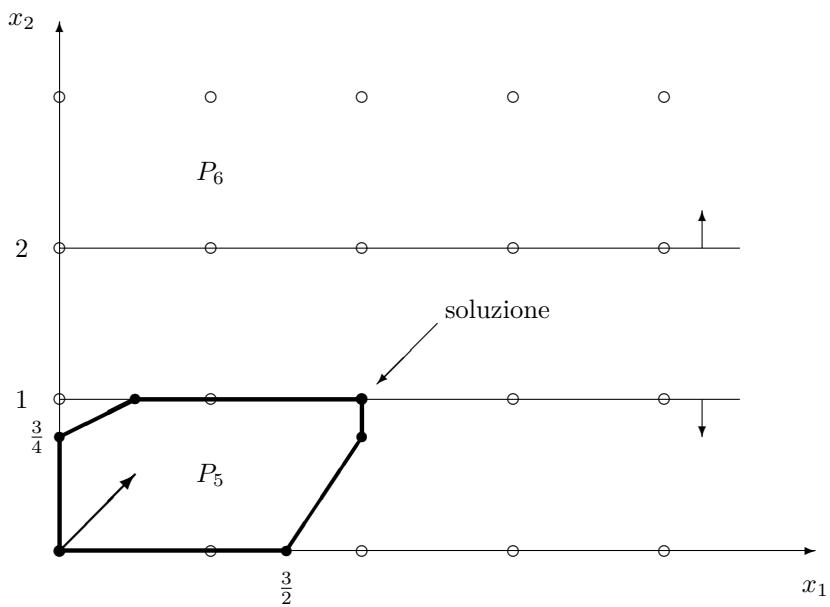
La soluzione ottima del rilassamento continuo del sottoproblema 1 è  $x_1 = \frac{17}{6}$ ,  $x_2 = 2$  e vale  $\frac{29}{6}$ ; l'upper bound corrispondente vale dunque  $\lfloor \frac{29}{6} \rfloor = 4$ . La variabile  $x_1$  ha valore frazionario compreso tra 2 e 3. Ramificando mediante i semipiani  $x_1 \leq 2$  ed  $x_1 \geq 3$  si ottengono i politopi  $P_3$  e  $P_4$  di figura.

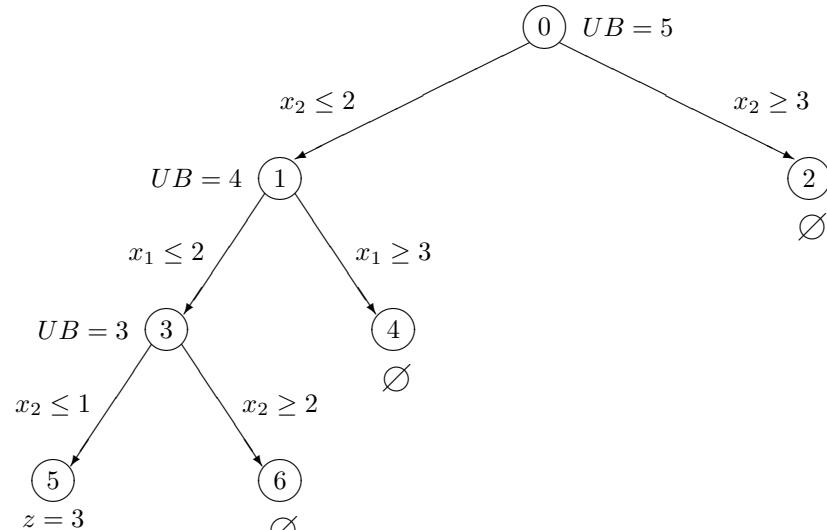


e l'albero decisionale



La soluzione ottima del rilassamento continuo del sottoproblema 3 è  $x_1 = 2$ ,  $x_2 = \frac{7}{4}$  e vale  $\frac{15}{4}$ ; l'upper bound corrispondente vale dunque  $\lfloor \frac{15}{4} \rfloor = 3$ . La variabile  $x_2$  ha valore frazionario compreso tra 1 e 2. Ramificando mediante i semipiani  $x_2 \leq 1$  ed  $x_2 \geq 2$  si ottengono i politopi  $P_5$  e  $P_6$  e l'albero decisionale mostrato in figura.





La soluzione ottima del rilassamento continuo del sottoproblema 5 ( $x_1 = 2$ ,  $x_2 = 1$ ) è intera e vale 3. Poiché non si ha nessun nodo attivo da cui ramificare, tale soluzione è quella ottima.

## Esercizio 11.2.

Un agricoltore deve preparare il mangime per i suoi polli miscelando due tipi di semi, A e B, disponibili in dosi preconfezionate. Il contenuto energetico dei semi A è doppio di quello dei semi B. Una dose di semi A contiene 4 unità di vitamina Q, mentre una dose di semi B ne contiene 5. La dieta non può contenere più di 20 unità in totale di tale vitamina. L'agricoltore, inoltre, vuole somministrare almeno  $4/5$  di dose di semi B e non più di 4 dosi di semi di tipo A.

- a) Scrivere il modello di programmazione lineare intera che permette di determinare la miscelazione che massimizza il contenuto energetico del mangime.
- b) Supponendo inizialmente accettabili soluzioni frazionarie si risolva il problema con l'algoritmo del simplesso, utilizzando il metodo delle due fasi e la regola di Bland.
- c) Si disegni con cura la regione ammissibile e vi si indichino con chiarezza le soluzioni corrispondenti a ciascun tableau esaminato al punto precedente.
- d) Si commenti la soluzione ottima determinata e se ne giustifichino le proprietà rilevate.
- e) Si supponga ora necessario determinare una soluzione intera. Risolvere il problema mediante il metodo branch and bound. A tal fine:
  - e.1) Si risolvano gli eventuali rilassamenti continui necessari PER VIA GRAFICA, utilizzando un disegno diverso per ogni problema ed indicandovi con chiarezza la regione ammissibile del sottoproblema (tagli generati dal Branch and bound) e la relativa soluzione ottima.
  - e.2) Per il Branching si selezioni la variabile frazionaria di indice MINIMO e si sviluppi il nodo relativo al problema più promettente.
  - e.3) Si disegni l'albero decisionale sviluppato indicando per ciascun nodo il valore del bound e la soluzione ottenuta.
  - e.4) Si indichi la soluzione ottima complessiva trovata.

## Soluzione

- a) Si utilizzano due variabili decisionali  $x_1$  e  $x_2$  che rappresentano il numero di dosi di semi A e B acquistate, rispettivamente. Il modello PLI del problema è

$$\begin{aligned}
 \max z = & \quad 2x_1 + x_2 \\
 & 4x_1 + 5x_2 \leq 20, \\
 & x_1 \leq 4, \\
 & x_2 \geq \frac{4}{5}, \\
 & x_1, x_2 \geq 0, \text{ intere.}
 \end{aligned}$$

b) Rimuovendo il vincolo di interezza e ponendo il modello in forma standard si ottiene

$$\begin{array}{lllll} -\min -z = & -2x_1 & -x_2 & & \\ & 4x_1 & +5x_2 & +x_3 & = 20, \\ & 4x_1 & & & +x_4 = 4, \\ & & x_2 & & -x_5 = \frac{5}{4}, \\ & x_1, & x_2, & x_3, & x_4, & x_5 \geq 0. \end{array}$$

Soluzione del rilassamento continuo

E' necessaria la Fase 1, aggiungendo una sola variabile artificiale (associata al secondo vincolo)

	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$y_1$
$-w$	0	0	0	0	0	1
$x_3$	20	4	5	1	0	0
$x_4$	4	4	0	0	1	0
$y_1$	$\frac{5}{4}$	0	1	0	0	-1

Poniamo il tableau in forma canonica rispetto alla base formata dalle colonne associate alle variabili  $x_3$ ,  $x_4$  ed  $y_1$  sottraendo la riga 3 dalla funzione obiettivo.

	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$y_1$
$-w$	$-\frac{5}{4}$	-1	0	0	1	0
$x_3$	20	4	5	1	0	0
$x_4$	4	4	0	0	1	0
$y_1$	$\frac{5}{4}$	0	1*	0	0	-1

Eseguiamo il pivot sulla colonna associata alla variabile  $x_2$  e di conseguenza sulla riga 3 che corrisponde al minimo dei rapporti considerando le sole righe con elementi positivi in tale colonna. Esce quindi dalla base la variabile  $y_1$ .

Poniamo il tableau in forma canonica rispetto alla base formata dalle colonne associate alle variabili  $x_2$ ,  $x_3$  ed  $x_4$ . A tal fine:

- sommiamo la riga 3 alla funzione obiettivo;
- sottraiamo dalla riga 1 la riga 3 moltiplicata per 5.

Il nuovo tableau è

236 CAPITOLO 11. ESERCIZI DI PROGRAMMAZIONE LINEARE INTERA

		$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$y_1$
$-w$	0	0	0	0	0	0	1
$x_3$	16	4	0	1	0	5	-5
$x_4$	4	4	0	0	1	0	0
$x_2$	$\frac{4}{5}$	0	1	0	0	-1	1

La soluzione ottima di Fase 1 vale 0 ed è relativa alla base associata alle colonne  $x_2$ ,  $x_3$  e  $x_4$  possiamo quindi passare alla Fase 2 rimuovendo le variabili artificiali e ripristinando la funzione obiettivo originale del problema.

Il nuovo tableau è

		$x_1$	$x_2$	$x_3$	$x_4$	$x_5$
$-z$	0	-2	-1	0	0	0
$x_3$	16	4	0	1	0	5
$x_4$	4	1	0	0	1	0
$x_2$	$\frac{4}{5}$	0	1	0	0	-1

Poniamo il tableau in forma canonica rispetto alla base sommando la riga 3 alla funzione obiettivo

		$x_1$	$x_2$	$x_3$	$x_4$	$x_5$
$z$	0	-2	0	0	0	-1
$x_3$	16	$4^*$	0	1	0	5
$x_4$	4	1	0	0	1	0
$x_2$	$\frac{4}{5}$	0	1	0	0	-1

Eseguiamo il pivot sulla colonna associata alla variabile  $x_1$ . Nel calcolo dei rapporti relativi alle righe con elementi positivi in tale colonna si ottiene una parità tra i valori nella riga 1, ossia  $\frac{16}{4}$ , e nella riga 2, ossia  $\frac{4}{1}$ . Di conseguenza, seguendo la regola di Bland si esegue il pivot scegliendo di far uscire dalla base la variabile di indice minimo. In questo caso eseguendo il pivot sulla riga 1 uscirebbe la variabile  $x_3$  mentre sulla riga 2 uscirebbe la variabile  $x_4$ . Si esegue pertanto il pivot sulla riga 1 ed esce la variabile  $x_3$ .

Poniamo il tableau in forma canonica rispetto alla base formata dalle colonne associate alle variabili  $x_2$ ,  $x_3$  ed  $x_4$ . A tal fine:

- dividiamo la riga 1 per 4;
- sommiamo la nuova riga 1 moltiplicata per 2 alla funzione obiettivo;
- sottraiamo dalla riga 2 la nuova riga 1.

Il nuovo tableau è

	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$
$z$	$\frac{44}{5}$	0	0	$\frac{1}{2}$	0
$x_1$	4	1	0	$\frac{1}{4}$	0
$x_4$	0	0	0	$-\frac{1}{4}$	1
$x_2$	$\frac{4}{5}$	0	1	0	0
					-1

Poiché tutte le variabili fuori base hanno costi ridotti non negativi la base corrente è ottima. La soluzione ottima è pertanto  $x_1 = 4$ ,  $x_2 = \frac{4}{5}$ ,  $x_3 = x_4 = x_5 = 0$  ed il suo valore è  $-z = -\frac{44}{5}$ . Rispetto alla funzione obiettivo originale del problema in forma di massimizzazione il valore della soluzione è pertanto  $z = \frac{44}{5} = 8.8$ .

- c) Il disegno della regione ammissibile del problema originale in due variabili decisionali è riportata in figura 11.1 in cui i punti interi sono rappresentati da pallini vuoti.

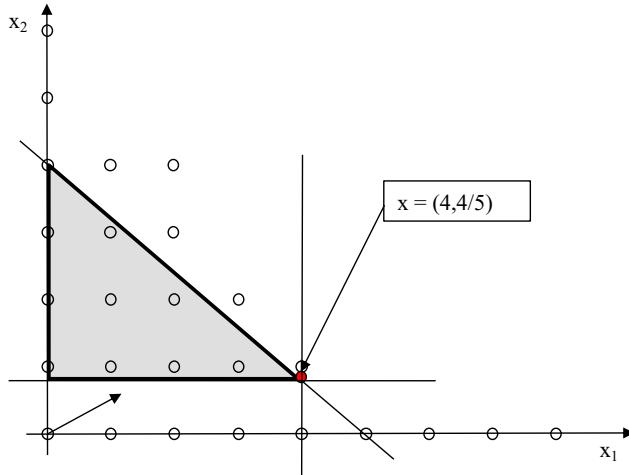


Figura 11.1: Regione ammissibile del problema.

La soluzione corrispondente al tableau iniziale di Fase 1 è l'origine. La soluzione corrispondente al tableau finale di Fase 1 è il punto di coordinate  $(0, \frac{4}{5})$ . La soluzione corrispondente al tableau finale di Fase 2 è la soluzione ottima di coordinate  $(4, \frac{4}{5})$ .

- d) Esaminando la regione ammissibile si può notare che il vertice ottimo del rilassamento continuo è sovradefinito. Infatti il vincolo  $x_1 \leq 4$  è chiaramente ridondante e passa per tale vertice. Come discusso nel paragrafo 4.6 la conseguenza di tale sovra definizione di un vertice è la degenerazione della base a

questo associata. Infatti la base ottima determinata al termine della Fase 2 è degenere dato che la variabile base  $x_4$  è in base sulla riga 2 a valore zero.

e) Soluzione del problema intero mediante Branch-and-Bound.

**Branching da  $P^0$ :** Consideriamo il problema originale in forma di massimizzazione la cui funzione obiettivo è  $\max z = 2x_1 + x_2$ . All'inizio dell'algoritmo non è nota nessuna soluzione intera del problema per cui si pone  $z^{Best} = -\infty$ .

La soluzione del rilassamento continuo del nodo radice è frazionaria  $x^{C0} = (4, \frac{4}{5})$  e vale  $z^{C0} = \frac{44}{5} = 8.8$ , che può essere arrotondato all'intero inferiore dato che i coefficienti della funzione obiettivo sono interi. Per cui  $UB^0 = \lfloor 8.8 \rfloor = 8$ . Poiché nessuna delle tre condizioni a)-c) elencate nel capitolo ?? ci permette di considerare il problema come risolto dobbiamo effettuare una suddivisione del problema (branching).

Scegliendo la variabile frazionaria di indice minimo effettuiamo il branching sulla variabile  $x_2 = \frac{4}{5}$  e generiamo i due sottoproblemi:

- $P^1 = P^0 + x_2 \leq 0$ ,
- $P^2 = P^0 + x_2 \geq 1$ ,

In figura 11.2 sono rappresentate le regioni ammissibili dei due problemi generati e sono indicate le soluzioni ottime dei rilassamenti continui corrispondenti. I valori dei bound calcolati dai rilassamenti continui sono riportati in figura 11.3.

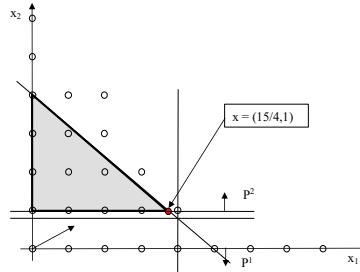


Figura 11.2: Regione ammissibile e soluzione del rilassamento continuo dei due sottoproblemi generati.

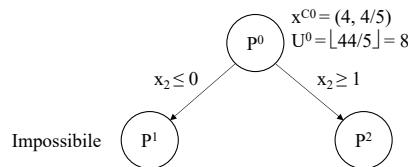


Figura 11.3: Albero decisionale dopo il primo branching.

Al termine del primo branching abbiamo quindi generato due sottoproblemi:

- $P^1$  la cui soluzione del rilassamento continuo è impossibile e quindi tale problema non andrà ulteriormente sviluppato;
- $P^2$  la cui soluzione del rilassamento continuo è frazionaria  $x^{C2} = (\frac{15}{4}, 1)$  a cui corrisponde un upper bound  $U^2 = \lfloor \frac{17}{2} \rfloor = 8$ .

Il sottoproblema  $P^1$  è chiuso ma il sottoproblema  $P^2$  non è risolto e quindi dovremo effettuare un ulteriore branching da questo.

**Secondo branching.** Seguendo la regola best bound first viene scelto per il branching il sottoproblema  $P^2$  che è comunque l'unico attivo. La soluzione del rilassamento continuo  $x^{C2} = (\frac{15}{4}, 1)$  ha una sola variabile frazionaria che quindi verrà usata per il branching generando due ulteriori sottoproblemi:

- $P^3 = P^2 + x_1 \leq 3, = P^0 + x_2 \geq 1 + x_1 \leq 3,$
- $P^4 = P^2 + x_1 \geq 4, = P^0 + x_2 \geq 1 + x_1 \geq 4,$

In figura 11.4 sono rappresentate le regioni ammissibili dei due problemi generati e sono indicate le soluzioni ottime dei rilassamenti continui corrispondenti. I valori dei bound calcolati dai rilassamenti continui sono riportati in figura 11.5.

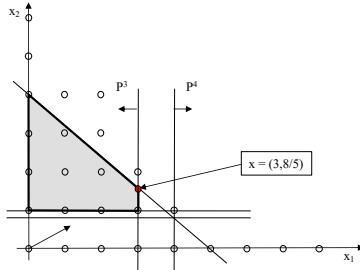


Figura 11.4: Regione ammissibile e soluzione del rilassamento continuo dei due sottoproblemi generati.

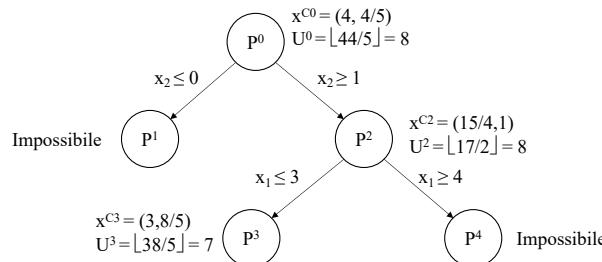


Figura 11.5: Albero decisionale dopo il secondo branching.

Al termine del secondo branching abbiamo quindi generato due sottoproblemi:

- $P^3$  la cui soluzione del rilassamento continuo è frazionaria  $x^{C3} = (3, \frac{8}{5})$  a cui corrisponde un upper bound  $U^3 = \lfloor \frac{38}{5} \rfloor = 7$ ;
- $P^4$  la cui soluzione del rilassamento continuo è impossibile e quindi tale problema non andrà ulteriormente sviluppato.

Al termine del secondo branching è ancora attivo il solo problema  $P^3$  e quindi dovremo effettuare il branching da questo.

**Terzo branching.** Seguendo la regola best bound first viene scelto per il branching il sottoproblema  $P^3$ . La soluzione del rilassamento continuo  $x^{C3} = (3, \frac{8}{5})$  ha una sola variabile frazionaria che quindi verrà usata per il branching generando due ulteriori sottoproblemi:

- $P^5 = P^3 + x_2 \leq 1, = P^0 + x_2 \geq 1 + x_1 \leq 3 + x_2 \leq 1,$
- $P^6 = P^3 + x_2 \geq 2, = P^0 + x_2 \geq 1 + x_1 \leq 3 + x_2 \geq 2,$

In figura 11.6 sono rappresentate le regioni ammissibili dei due problemi generati e sono indicate le soluzioni ottime dei rilassamenti continui corrispondenti. I valori dei bound calcolati dai rilassamenti continui sono riportati in figura 11.7.

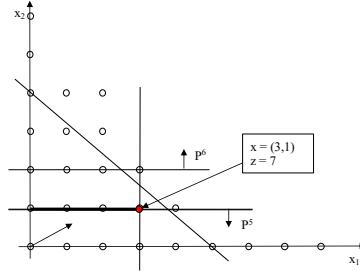


Figura 11.6: Regione ammissibile e soluzione del rilassamento continuo dei due sottoproblemi generati.

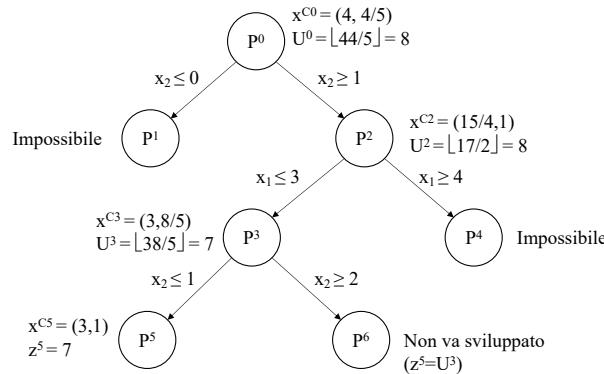


Figura 11.7: Albero decisionale dopo il terzo branching.

Al termine del terzo branching abbiamo quindi generato due sottoproblemi:

- $P^5$  la cui soluzione del rilassamento continuo è intera  $x^{C5} = (3, 1)$  a cui corrisponde una soluzione  $z^5 = 7$  che ci permette di aggiornare  $z^{Best} = z^5 = 7$  (e salvare la soluzione corrente  $x^{Best} = x^5 = (3, 1)$ );
- $P^6$  che non andrà sviluppato dato che il valore della soluzione intera di  $P^5$  è uguale al bound del nodo padre  $P^3$ .

Al termine del terzo branching nessun ulteriore sottoproblema è attivo e quindi il problema è risolto e la soluzione ottima è la soluzione  $x^{Best} = (3, 1)$  di valore 7 corrispondente a  $z^{Best}$ .



## Capitolo 12

# Risoluzione di problemi con Excel

I fogli elettronici, quali ad esempio Microsoft Excel, sono uno strumento di lavoro molto diffuso e si prestano in modo naturale alla rappresentazione dei problemi di PL e PLI. Infatti, molti modelli per foglio elettronico hanno una struttura simile a quella dei modelli matematici, consentendo di implementare la trasformazione di valori di input ( $x_1, \dots, x_n$ ) in valori di output ( $y$ ) mediante l'applicazione di funzioni matematiche  $y = f(x_1, \dots, x_n)$ . Inoltre in molti casi sono disponibili delle funzioni aggiuntive che consentono la risoluzione di tali problemi mediante implementazioni dell'algoritmo del simplex e dell'algoritmo Branch and Bound descritti nei capitoli precedenti. Nel seguito illustreremo brevemente come utilizzare il risolutore di PL/PLI disponibile in Microsoft Excel, facendo principalmente riferimento alla versione per Windows. Maggiori dettagli su tale risolutore, realizzato da Frontline Systems, sono disponibili online tra le risorse di Microsoft Excel (si veda, ad esempio, <https://appsource.microsoft.com/it-it/product/office/wa104100404?tab=overview>).

Il risolutore è un componente aggiuntivo che non è necessariamente attivo in Excel. Per attivarlo è sufficiente selezionare i componenti aggiuntivi di Excel (disponibili nel menu File > Opzioni) ed attivare la casella per il Componente aggiuntivo Risolutore. Dopo l'attivazione nel menu Dati comparirà il pulsante Risolutore che permette di accedere alla finestra di definizione del problema di ottimizzazione di cui discuteremo in seguito.

Al fine di illustrare in via intuitiva in cosa consista l'analisi di sensitività per un problema PL consideriamo il problema della produzione di vasche per idromassaggio descritto nel paragrafo 3.1.2. Il modello matematico del problema

è:

$$\begin{aligned}
 \max z = & 350 x_1 + 300 x_2 \\
 x_1 + x_2 &\leq 200 \\
 9x_1 + 6x_2 &\leq 1566 \\
 12x_1 + 16x_2 &\leq 2880 \\
 x_1, x_2 &\geq 0.
 \end{aligned}$$

Per implementare un modello PL/PLI sul foglio Excel è necessario seguire alcuni passi preliminari. In particolare, come illustrato in Figura 12.1:

1. Inserire i dati del problema in apposite celle del foglio, una per ogni singolo dato:
  - coefficienti della funzione obiettivo ( $c_i, i = 1, \dots, n$ );
  - termini noti dei vincoli ( $b_j, j = 1, \dots, m$ );
  - coefficienti dei vincoli ( $a_{i,j}, i = 1, \dots, n, j = 1, \dots, m$ ).
2. Riservare un insieme appropriato di celle per contenere i valori delle variabili decisionali ( $x_i, i = 1, \dots, n$ ).
3. Inserire in una apposita cella la funzione che permette di calcolare il valore della funzione obiettivo a partire dai coefficienti di costo e dai valori delle variabili decisionali ( $z = \sum_{i=1}^n c_i x_i$ ).
4. Per ogni vincolo inserire in una apposita cella la funzione che permette di calcolare il valore primo membro del vincolo a partire dai relativi coefficienti e dai valori delle variabili decisionali ( $l_j = \sum_{i=1}^n a_{i,j} x_i, j = 1, \dots, m$ ).

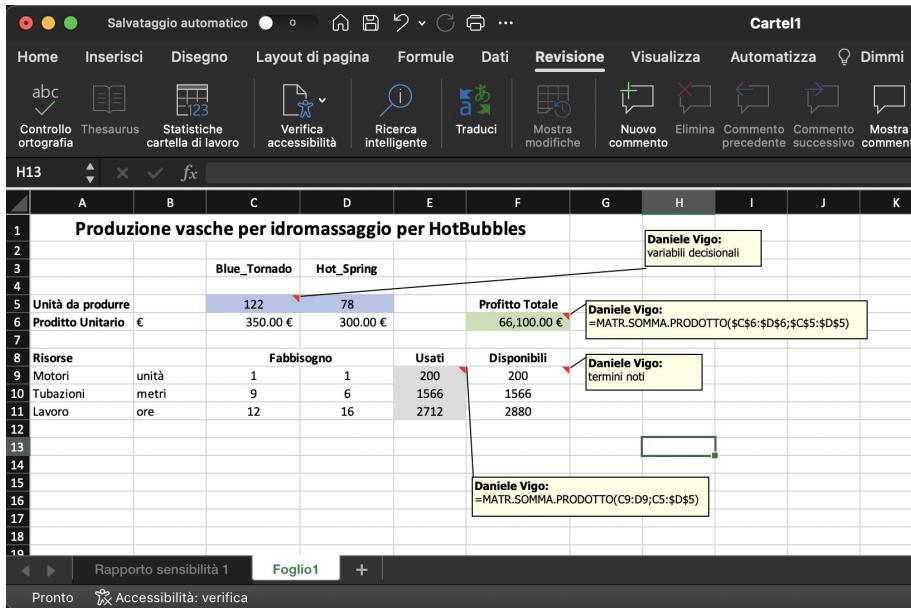


Figura 12.1: Impostazione su Excel del modello PL di produzione delle vasche per idromassaggio.

Si noti che per l'inserimento della funzione obiettivo e dei vincoli è opportuno servirsi della funzione Excel MATR.SOMMAPRODOTTO() che restituisce la somma dei prodotti di elementi corrispondenti di due insiemi di celle. Si ricordi inoltre l'utilità del simbolo \$ per bloccare in un range di valori uno o più indici quando una formula viene copiata in celle diverse. Infine è possibile indicare in celle opportune delle didascalie che specificano le varie componenti del modello ed il significato di variabili e vincoli. Tali didascalie hanno lo scopo di aumentare la leggibilità del modello ed in alcuni casi sono utilizzati automaticamente dal risolutore nella produzione dei report.

Una volta predisposto il foglio con i dati, le variabili e le funzioni è possibile accedere alla finestra di configurazione del modello disponibile nel menu Dati premendo il tasto Risolutore. Compare a questo punto la finestra illustrata in Figura 12.2 in cui è possibile definire:

- L'indirizzo della cella che contiene la formula che calcola il valore della funzione obiettivo (Cella Obiettivo).
- Il tipo di ottimizzazione desiderata (max o min).
- Gli indirizzi delle celle che contengono le variabili decisionali (XXXX).
- La definizione dei vincoli e dei tipi di variabili del problema (YYYY).

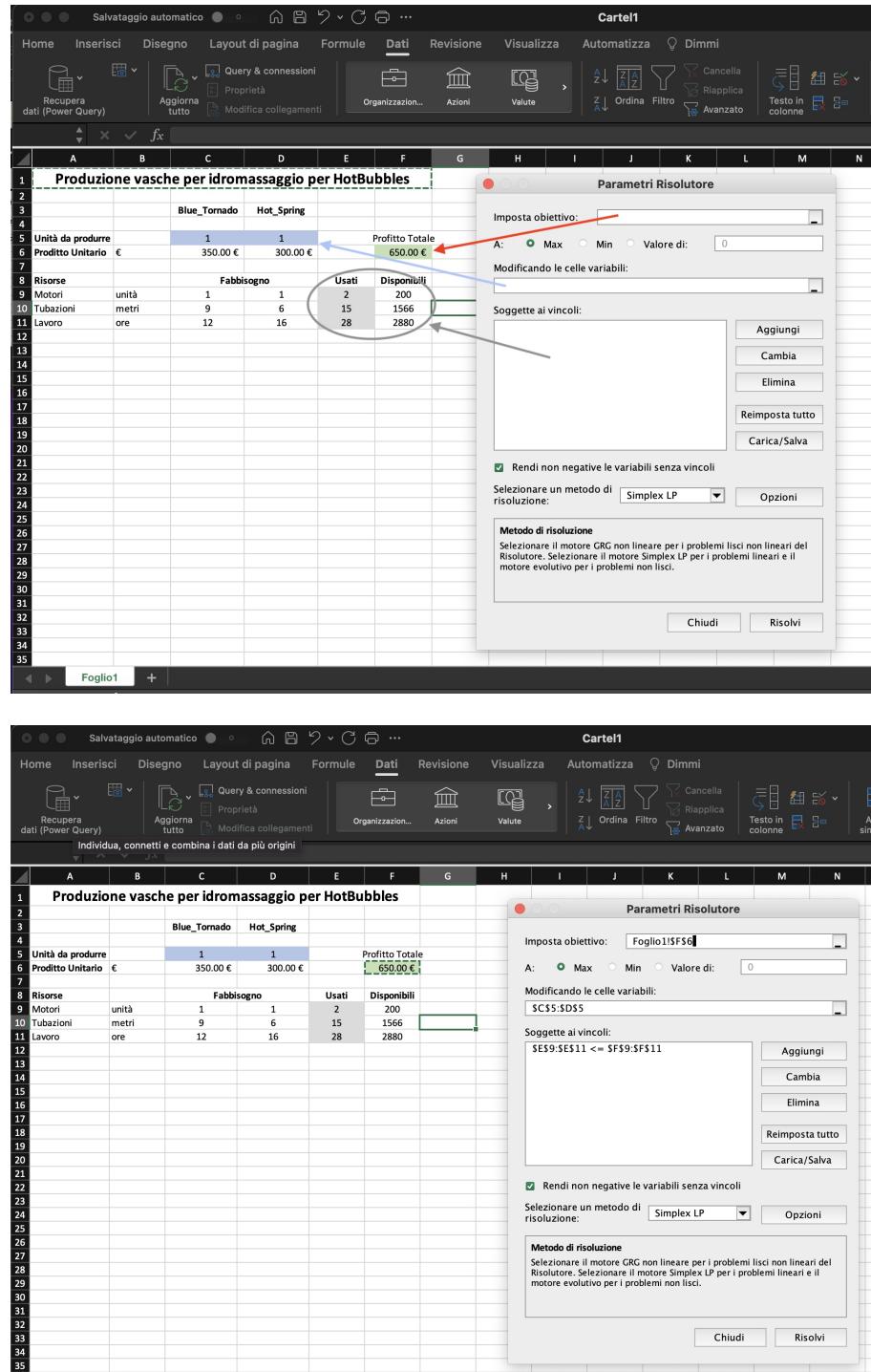


Figura 12.2: Finestra di impostazione delle componenti del del modello PL/PLI  
(da rifare in italiano)

La definizione dei vincoli e delle variabili avviene attraverso un'apposita ulteriore finestra in cui per ogni vincolo è necessario inserire:

- L'indirizzo della cella che contiene la formula che calcola il valore del primo membro del vincolo.
- Il tipo di vincolo ( $\leq$ ,  $=$  o  $\geq$ ).
- L'indirizzo della cella che contiene il valore del termine noto del vincolo.

Si noti che vincoli dello stesso tipo possono essere inseriti assieme specificando i range opportuni dei primi membri e dei termini noti di tali vincoli.

La stessa finestra permette inoltre di definire il tipo di variabili del problema. In tal caso nella prima tendina è possibile impostare il range delle variabili interessate e nella seconda è possibile specificare se si tratti di variabili intere (int), binarie (bin) o libere in segno (free). Si noti che non è necessario inserire esplicitamente i vincoli di non negatività delle variabili dato che è possibile attivare la spunta alla casella “Assumere tutte le variabili non negative”.

L'ultima impostazione necessaria è la scelta del risolutore da utilizzare e nel nostro caso questo è costituito da “Simplex LP”.

Una volta terminata l'impostazione del problema è possibile richiederne la risoluzione premendo il pulsante Risovi. Se la terminazione dell'algoritmo è positiva compare una finestra (illustrata in Figura 12.3) che segnala il buon fine della risoluzione e chiede se depositare nelle celle delle variabili i valori ottimi determinati. In caso il problema sia impossibile, illimitato o la convergenza dell'algoritmo non sia verificata per problemi di instabilità numerica tali condizioni vengono segnalate in apposite finestre che permettono di individuare l'esito della risoluzione.

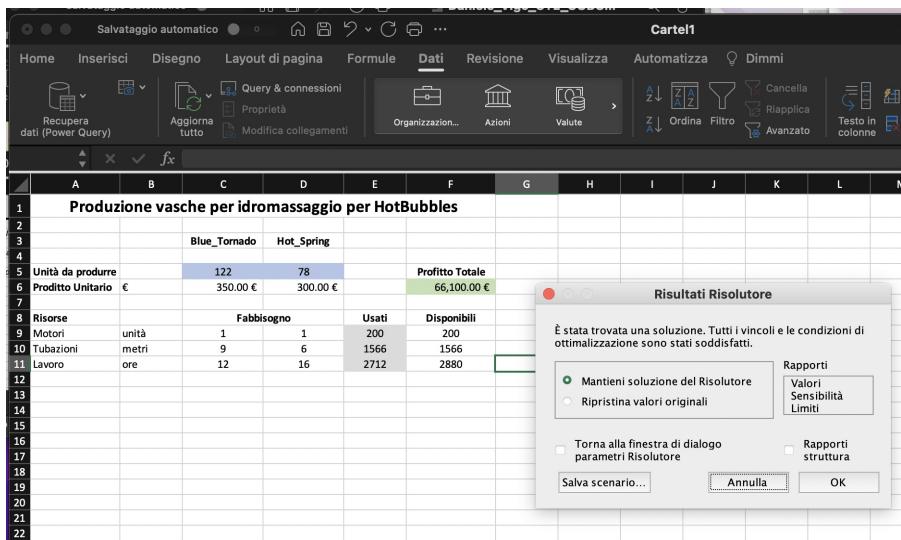


Figura 12.3: Esito positivo della soluzione di un problema PL/PLI.

Nella finestra che comunica l'esito della risoluzione è possibile richiedere la produzione del “Report Sensitività” che attiva la costruzione automatica di un foglio che riporta l'analisi di sensitività per i coefficienti della funzione obiettivo e dei termini noti dei vincoli del problema. Si noti che le traduzioni italiane di alcune componenti sono approssimative o scorrette. L'esito dell'analisi di sensitività del modello è illustrato nella Figura 12.4.

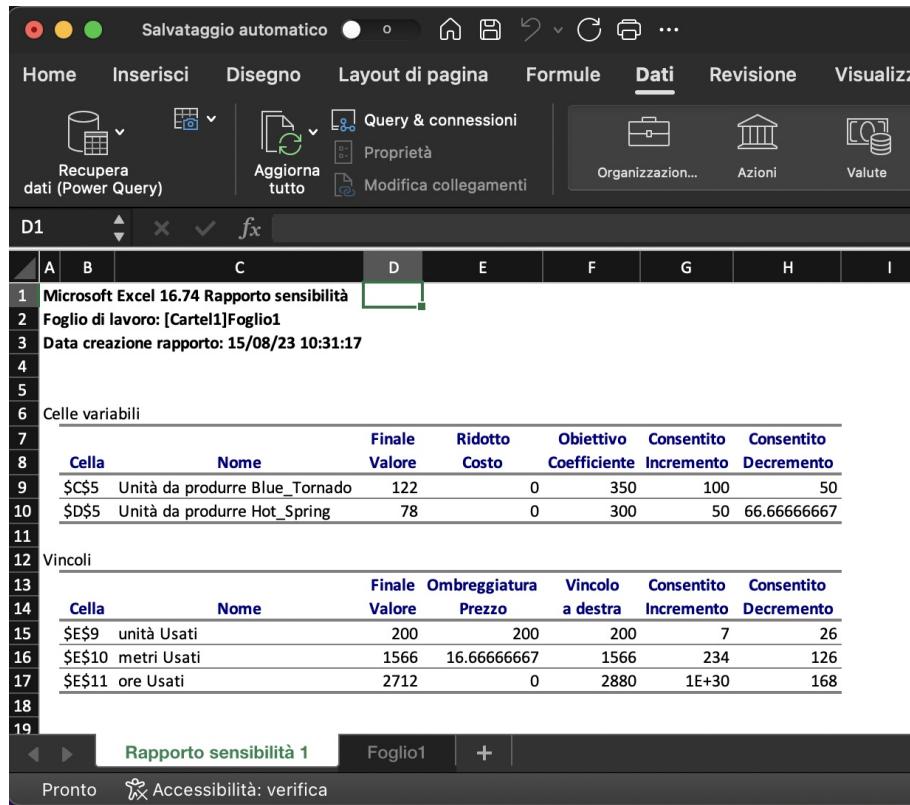


Figura 12.4: Report di sensitività per un modello PL.

Aggiungere spiegazione delle opzioni

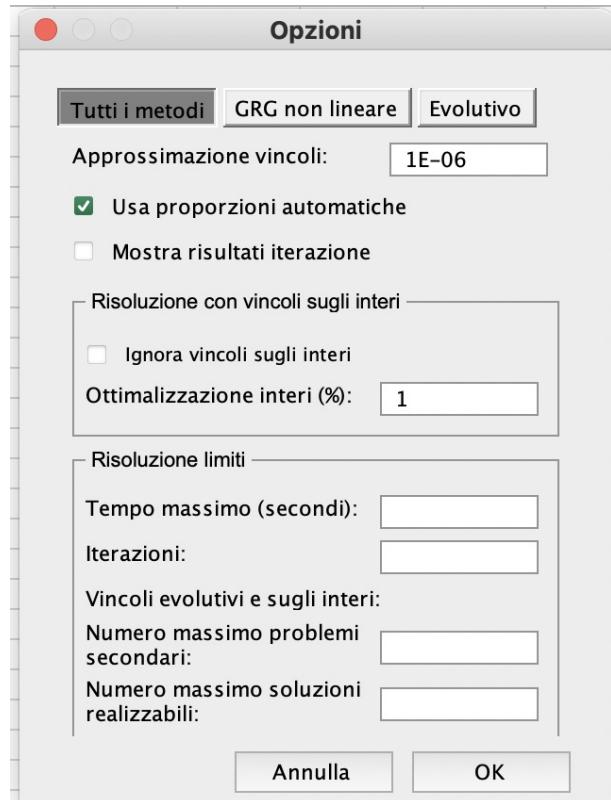


Figura 12.5: Opzioni del solver Excel per un modello PL.



# Capitolo 13

## Teoria dei Grafi

### 13.1 Introduzione

Un *grafo non orientato*, rappresentato come  $G = (V, E)$ , è definito dall'insieme dei *vertici* (o *nodi*) e dall'insieme dei *lati* che congiungono coppie non ordinate di vertici. In particolare:

- $V = \{1, 2, \dots, n\}$ : insieme dei vertici (o nodi);
- $E = \{e_1, e_2, \dots, e_m\}$ : insieme dei lati, che corrispondono a coppie *non ordinate* di vertici di  $V$  che sono *collegati*, i.e., un lato  $e_k = \{i, j\}$  collega i vertici  $i$  e  $j$ .

Due vertici sono *adiacenti* se esiste il lato che li collega, mentre due lati sono *consecutivi* se hanno un vertice in comune. Risulterà molto utile denotare con  $E(S)$  l'insieme dei lati con entrambi gli estremi nel sottoinsieme di vertici  $S \subseteq V$  e  $\Gamma(i)$  insieme dei vertici collegati a  $i$ .

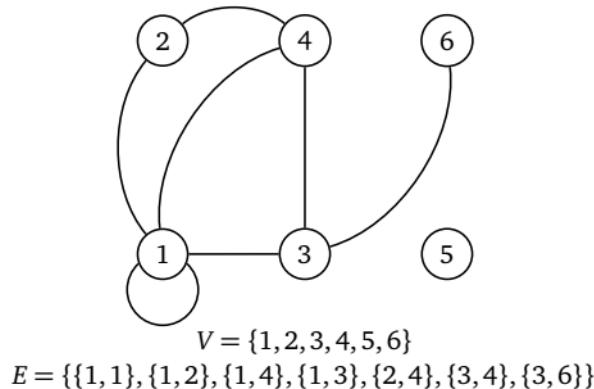


Figura 13.1: Esempio di grafo non orientato.

In figura 13.1 è riportato un grafo non orientato, in cui, per esempio, i vertici 3 e 4 sono collegati con il lato  $e = \{3, 4\}$ , e quindi sono adiacenti, mentre il vertice 5 non è collegato con nessun altro vertice. In quest'ultimo caso si dice che il vertice 5 *non è connesso*. In figura 13.1 si può osservare che, per esempio, i lati  $e = \{3, 4\}$  e  $e' = \{3, 6\}$  sono consecutivi. Inoltre, il grafo ha un *loop* (anche detto *autoanello* o *cappio*) in corrispondenza del vertice 1, in quanto esiste il lato  $\{1, 1\}$ . Se si considera il sottoinsieme di vertici  $S = \{1, 2, 4\}$  allora  $E(S) = \{\{1, 1\}, \{1, 2\}, \{1, 4\}, \{2, 4\}\}$ . Inoltre, per esempio,  $\Gamma(2) = \{1, 4\}$ ,  $\Gamma(4) = \{1, 2, 3\}$ ,  $\Gamma(5) = \emptyset$ .

Un grafo è detto *multiplo* se ha più di un lato per la stessa coppia di vertici. Un grafo non orientato è detto *semplice* se non comprende loop e archi multipli. Generalmente considereremo solo grafi semplici. Un grafo è *completo* se per ogni coppia di vertici esiste un lato. In Figura 13.2 sono proposti degli esempi di grafo multiplo, semplice e completo.

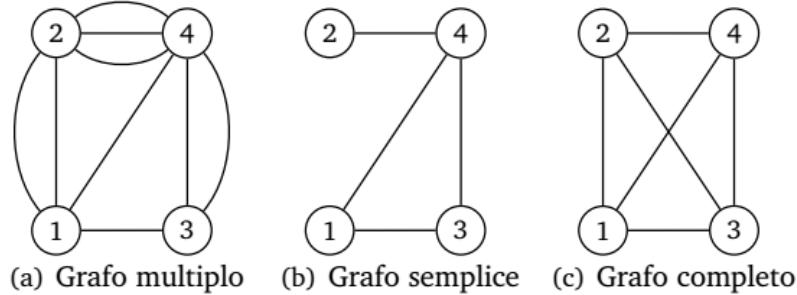


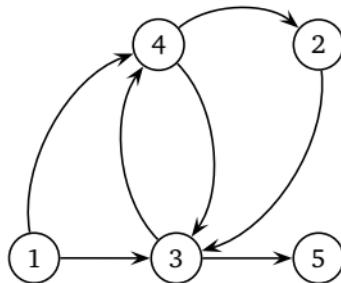
Figura 13.2: Esempi di grafo multiplo, semplice e completo.

Un *grafo orientato* (o *grafo diretto*) si differenzia da un grafo non orientato per la sostituzione dell'insieme dei lati con l'insieme degli *archi*, che sono coppie *ordinate* di vertici. Quindi, per un grafo orientato  $G = (V, A)$  avremo che:

- $V = \{1, 2, \dots, n\}$ : insieme dei vertici (o nodi);
- $A = \{a_1, a_2, \dots, a_m\}$ : insieme degli archi, che corrispondono a coppie *ordinate* di vertici di  $V$ , i.e., l'arco  $a_k = (i, j)$  indica che il vertice  $i$  è collegato al vertice  $j$ , ma il vertice  $j$  non è collegato al vertice  $i$  a meno che non esista un altro arco  $a_h = (j, i)$ .

Dato l'arco  $(i, j)$  il vertice  $i$  è detto *vertice iniziale* (o *testa*) e  $j$  è detto *vertice terminale* (o *coda*). Inoltre, il vertice  $j$  è anche detto *successore* di  $i$  mentre  $i$  è detto *predecessore* di  $j$ .

Per i grafi orientati sarà utile denotare con  $A(S)$  l'insieme degli archi con entrambi gli estremi (vertice iniziale e finale) nel sottoinsieme di vertici  $S \subseteq V$  e con  $\Gamma^+(i)$  e  $\Gamma^-(i)$  gli insiemi dei successori e dei predecessori di  $i$ , rispettivamente.



$$V = \{1, 2, 3, 4, 5\} \quad A = \{(1, 4), (1, 3), (3, 4), (4, 3), (4, 2), (2, 3), (3, 5)\}$$

Figura 13.3: Esempio di grafo orientato.

In figura 13.3 è riportato un esempio di grafo in cui, per esempio, l'arco  $(1, 4)$  *esce* dal vertice 1 ed *entra* nel vertice 4. Quindi il vertice 4 può essere raggiunto dal vertice 1, ma non può accadere il contrario, ossia dal vertice 4 non si può tornare al vertice 1. Se consideriamo il sottoinsieme di vertici  $S = \{1, 3, 4\}$ , allora  $A(S) = \{(1, 4), (1, 3), (3, 4), (4, 3)\}$ . Inoltre, per esempio,  $\Gamma^+(1) = \{3, 4\}$ ,  $\Gamma^+(4) = \{2, 3\}$ ,  $\Gamma^+(5) = \emptyset$ , mentre  $\Gamma^-(1) = \emptyset$ ,  $\Gamma^-(4) = \{1, 3\}$ ,  $\Gamma^-(5) = \{3\}$ .

Un grafo  $G$  non orientato (orientato) è pesato sui lati (archi) se esiste una funzione  $c : E \rightarrow \mathbb{R}$  ( $c : A \rightarrow \mathbb{R}$ ) che associa un *peso* ad ogni lato (arco). Il peso può rappresentare un costo, una distanza, un tempo di percorrenza, etc. Invece, il grafo  $G$  è pesato sui vertici se esiste una funzione  $w : V \rightarrow \mathbb{R}$  che associa un *peso* ad ogni vertice. In figura 13.4 è riportato un esempio di grafo non orientato pesato.

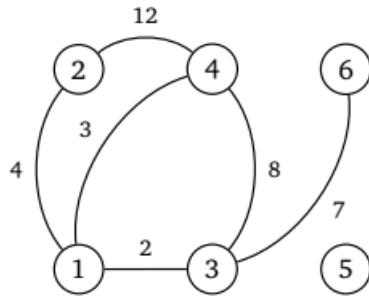


Figura 13.4: Esempio di grafo non orientato pesato.

### 13.1.1 Applicazioni

Le applicazioni della teoria dei grafi possono spaziare dalla teoria alla pratica.

Tra gli argomenti più noti nell'ambito della teoria dei grafi possiamo citare ad esempio:

- Cammini Euleriani: originato dal problema posto da Eulero, per determinare un percorso che, partendo da una qualsiasi delle quattro zone della città di Könisberg, attraversasse tutti i sette ponti una ed una sola volta ritornando al punto di partenza.
- Colorazione dei grafi: dove un esempio di applicazione e la colorazione delle mappe per garantire di non usare lo stesso colore per nazioni confinanti.
- Problema della clique (cricca): per esempio per calcolare la clique (i.e., sottografo completo) di cardinalità massima.

Per quanto riguarda le applicazioni nel mondo reale, in figura 13.5 sono riportati alcuni esempi proposti dal *MIT Center for Transportation & Logistics* in cui viene fornita anche qualche informazione su come il problema può essere modellato con i grafi. Come si può osservare, le applicazioni possono spaziare dalla progettazione e ottimizzazione di reti di comunicazione fino alle reti di trasporto.

Applications	Physical Analog of Nodes	Physical Analog of Arcs	Flow
Communication systems	Telephone exchanges, computers, transmission facilities, satellites	Cables, fiber optic links, microwave relay links	Voice messages, data, video transmissions
Hydraulic systems	Pumping stations, reservoirs, lakes	Pipelines	Water, gas, oil, hydraulic fluids
Integrated computer circuits	Gates, registers, processors	Wires	Electrical current
Mechanical systems	Joints	Rods, beams, springs	Heat, energy
Transportation systems	Intersections, airports, rail yards	Highways, railbeds, airline routes	Passengers, freight, vehicles, operators

Figura 13.5: Esempi di alcune applicazioni (tabella proposta dal *MIT Center for Transportation & Logistics*).

### 13.1.2 Taglio di un grafo

Dato un sottoinsieme  $S$  di vertici, si dice *taglio* l'insieme dei lati (o archi) che congiungono i vertici in  $S$  con quelli in  $V \setminus S$ .

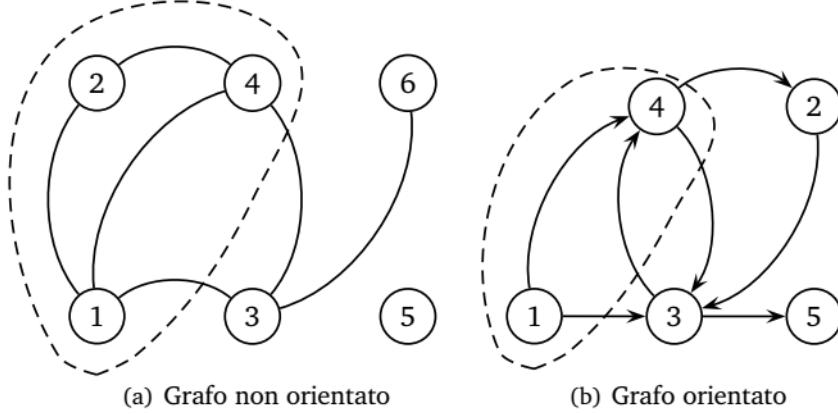


Figura 13.6: Esempio di taglio in un grafo non orientato e in uno orientato.

Nella figura 13.6 sono riportati due esempi di taglio in un grafo non orientato e in un grafo orientato.

Per i grafi non orientati si può rappresentare il taglio come:

$$\delta_G(S) = \{\{i, j\} \in E : i \in S, j \in V \setminus S \text{ oppure } j \in S, i \in V \setminus S\}$$

e nell'esempio di figura 13.6.a sarebbe  $\delta_G(\{1, 2, 4\}) = \{(1, 3), (3, 4)\}$ .

Per i grafi orientati invece si deve distinguere tra archi uscenti ed entranti in  $S \subset V$ :

- $\delta_G^+(S) = \{(i, j) \in A : i \in S, j \notin S\};$
- $\delta_G^-(S) = \{(i, j) \in A : j \in S, i \notin S\}.$

Si noti che  $\delta_G^+(S) \equiv \delta_G^-(V \setminus S)$ . Per l'esempio di figura 13.6.b si ha  $\delta_G^+(\{1, 4\}) = \{(1, 3), (4, 2), (4, 3)\}$  e  $\delta_G^-(\{1, 4\}) = \{(3, 4)\}$ .

### 13.1.3 Cammini, circuiti e cicli

Un cammino è una sequenza di vertici  $v_1, v_2, \dots, v_k \in V$  tale che per ogni coppia di vertici consecutivi  $(v_i, v_{i+1})$  esiste il corrispondente lato (grafo non orientato) o arco (grafo orientato). Quindi, un cammino  $P$  si può rappresentare sia come una sequenza di vertici:

$$P = (v_1, v_2, v_3, \dots, v_k)$$

Però, è possibile rappresentare un cammino anche come una sequenza archi (o lati):

$$P = ((v_1, v_2), (v_2, v_3), (v_3, v_4), \dots, (v_{k-1}, v_k))$$

In figura 13.7 è proposto un grafo orientato e sono specificati alcuni esempi di cammini presenti in esso. In figura 13.8 è stato evidenziato il cammino  $P_1$ .

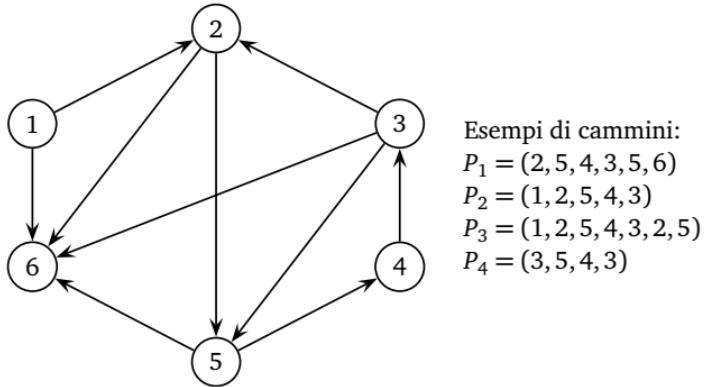
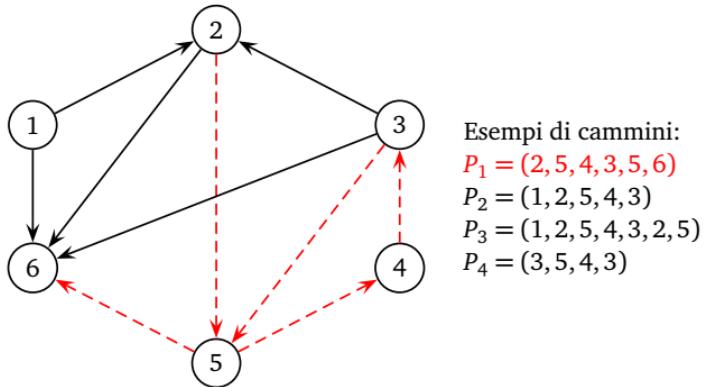


Figura 13.7: Esempio di cammini in un grafo orientato.

Figura 13.8: Esempio di cammini in un grafo orientato, in cui è evidenziato  $P_1$ .

Dato un cammino  $P = (v_1, v_2, v_3, \dots, v_k)$  il suo *costo*  $c(P)$  è dato da :

$$c(P) = \sum_{i=1}^{k-1} c_{v_i v_{i+1}}$$

dove  $c_{ij}$  è il costo dell'arco  $(i, j)$ . Il costo di un cammino, a seconda del contesto e dell'applicazione, è anche detto *lunghezza*, *peso*, etc.

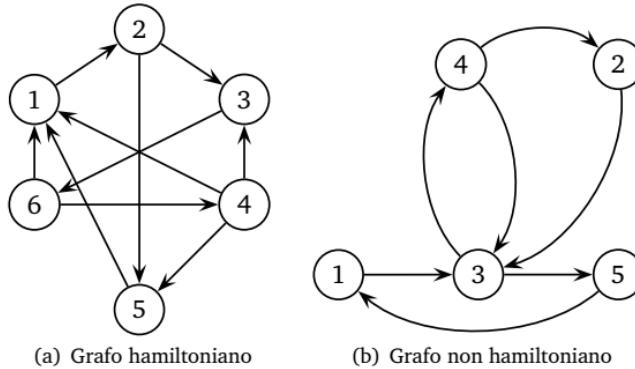
Tra i diversi cammini in un grafo possiamo identificare i seguenti casi particolari di notevole interesse:

- *Cammino semplice*: non usa più di una volta lo stesso arco/lato.
- *Cammino elementare*: non passa più di una volta per lo stesso vertice.
- *Cammino hamiltoniano*: usa una ed una sola volta tutti i vertici del grafo; quindi deve visitare tutti vertici del grafo.

- *Cammino euleriano*: usa una ed una sola volta tutti gli archi/lati del grafo.
- *Circuito*: è un cammino in un grafo orientato in cui il vertice iniziale coincide con il vertice terminale.
- *Circuito elementare*: è un circuito che, a parte il primo e l'ultimo vertice (che coincidono), non passa più di una volta per lo stesso vertice.
- *Circuito hamiltoniano*: è un circuito elementare che passa attraverso ogni vertice del grafo. Oppure, equivalentemente, è un cammino hamiltoniano chiuso (i.e., con un arco che collega l'ultimo vertice con il primo del cammino).
- *Circuito euleriano*: è un circuito elementare che passa attraverso ogni arco del grafo. Oppure, equivalentemente è un cammino euleriano chiuso.
- *Ciclo*: controparte non orientata di un circuito.

Nel grafo in figura 13.7  $P_1$ ,  $P_2$  e  $P_4$  sono cammini semplici, mentre  $P_3$  non lo è. Invece, mentre  $P_2$  è un cammino elementare,  $P_1$ ,  $P_3$  e  $P_4$  non lo sono. Nel grafo in figura 13.7 non ci sono cammini hamiltoniani ed euleriani e lo si può determinare piuttosto facilmente osservando che il vertice 1 ha solo archi uscenti e il vertice 6 ha solo archi entranti. Considerando sempre il grafo in figura 13.7 il cammino  $P = (2, 5, 4, 3, 2)$  è un circuito, che è anche elementare.

I grafi che possiedono almeno un circuito/ciclo hamiltoniano sono detti *grafi hamiltoniani*. Invece, i grafi che possiedono almeno un circuito/ciclo euleriano sono detti *grafi euleriani*. Non tutti i grafi possiedono un circuito/ciclo hamiltoniano o euleriano. Nella figura 13.9 è proposto l'esempio di due grafi, dove uno è hamiltoniano mentre l'altro non lo è.



Circuito elementare (a)  $C_1 = (1, 2, 3, 6, 1)$ , (b)  $C_2 = (3, 4, 2, 3)$ .  
 Circuito hamiltoniano (a)  $C_3 = (1, 2, 3, 6, 4, 5, 1)$ , (b) non ne possiede.

Figura 13.9: Esempio di grafo hamiltoniano e non hamiltoniano.

Un grafo che non contiene circuiti (cicli) è detto *grafo aciclico*. In figura 13.10 è riportato un esempio.

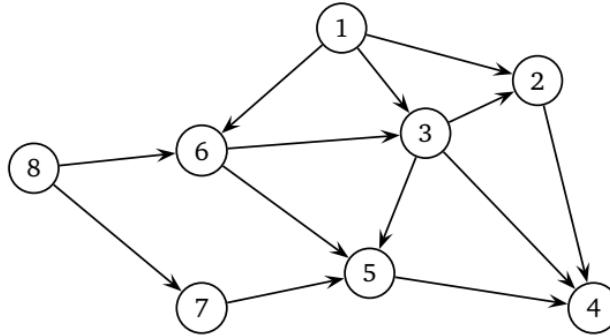


Figura 13.10: Esempio di grafo aciclico.

#### 13.1.4 Grafi parziali, sottografi e componenti

Dato un grafo  $G = (V, A)$ , un suo *grafo parziale* è il grafo  $G' = (V, A')$  dove  $A' \subset A$ . Invece, un *sottografo* di  $G = (V, A)$  è il grafo  $G' = (V', A')$  dove  $V' \subseteq V$  e  $A' \subseteq A$ .

In figura 13.11 è proposto un grafo orientato e sono proposti un suo grafo parziale e suo sottografo.

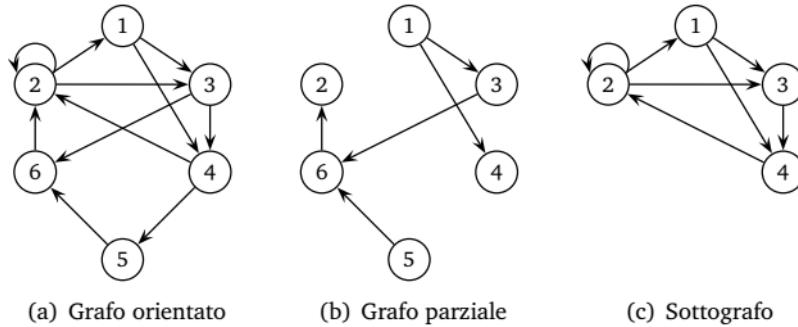


Figura 13.11: Esempio di grafo parziale e sottografo.

Un grafo è detto *connesso* se il grafo non orientato relativo al grafo orientato ha almeno un cammino che congiunge ogni coppia di vertici. Se, invece, tale cammino non esiste allora il grafo viene detto *disconnesso* (o semplicemente *non connesso*). In figura 13.12 sono forniti un esempio di grafo connesso e uno non connesso.

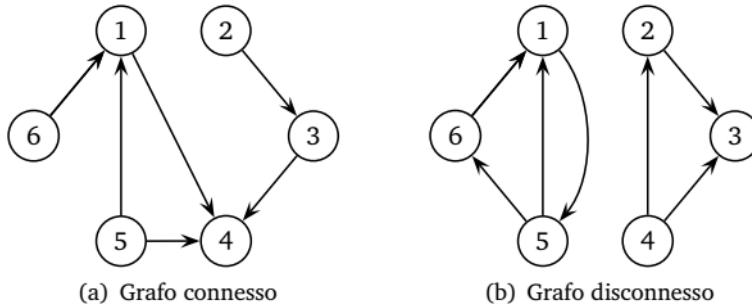


Figura 13.12: Esempio di grafo connesso e di grafo disconnesso.

Un grafo è detto *fortemente connesso* se nel grafo esiste almeno un cammino orientato che congiunge ogni coppia di vertici.

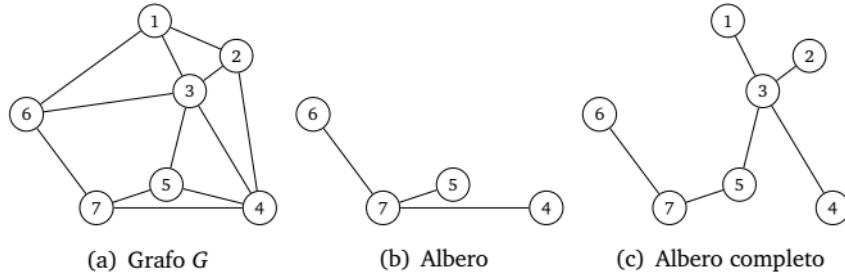
### 13.1.5 Alberi

Un grafo  $G_a$  non orientato di  $n$  vertici è un *albero* se rispetta le seguenti condizioni, che sono equivalenti:

- $G_a$  è connesso e aciclico;
- $G_a$  è aciclico e si crea un ciclo semplice se si aggiunge un lato al grafo  $G_a$ ;
- $G_a$  è connesso, ma diventa non connesso non appena si elimina un solo lato di  $G_a$ ;
- $G_a$  è connesso e ha  $n - 1$  lati;
- $G_a$  non ha cicli semplici e ha  $n - 1$  lati.

In letteratura esistono anche altre condizioni equivalenti. Ognuna di queste definizioni equivalenti può essere utile a "identificare" e "utilizzare" gli alberi.

Dato un grafo  $G$ , possono essere definiti dei sottografi di  $G$  che sono alberi. Tra questi, si definisce *albero completo* di  $G$  (detto anche *spanning tree*) un grafo parziale di  $G$  (i.e., "copre" tutti i vertici) che è un albero (si considerino gli esempi riportati in figura 13.13). Ogni grafo connesso ha almeno uno spanning tree.

Figura 13.13: Esempio di alberi e alberi completi di un grafo  $G$ .

Un sottografo connesso di un albero è a sua volta un albero, detto *sottoalbero*. Invece, un grafo che non contiene cicli è una *foresta*; quindi, equivalentemente, si può affermare che una foresta è un insieme di alberi.

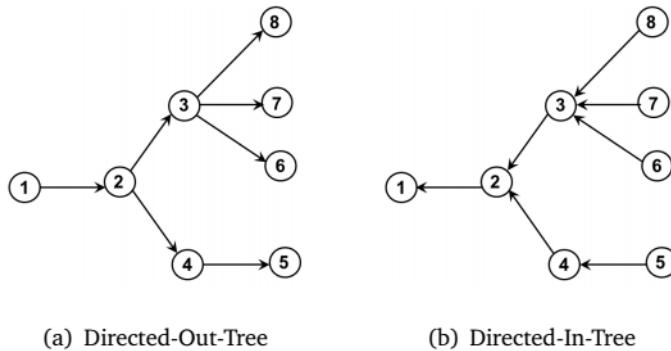


Figura 13.14: Esempio di directed-out-tree e directed-in-tree.

Dato un albero si può indicare come *radice* (*root*) uno dei suoi nodi. In questo caso l'albero è denominato *albero radicato* (*rooted tree*). Se si denota con  $s$  la radice di un albero, si possono definire due tipologie di alberi (vedi figura 13.14):

- *Directed-out-tree*: è un albero in cui l'unico cammino dal nodo  $s$  a tutti gli altri nodi è diretto.
- *Directed-in-tree*: è un albero in cui l'unico cammino da un qualsiasi altro nodo al nodo  $s$  è diretto.

### 13.1.6 Rappresentazione dei grafi

Il ruolo delle strutture dati è cruciale nello sviluppo di algoritmi efficienti. Il modo in cui sono salvati i dati del grafo (*rete*) nella memoria del calcolatore determina le performance degli algoritmi che operano su tali dati. Alcune delle operazioni che devono essere svolte dagli algoritmi sono le seguenti:

- Accedere alle informazioni dei vertici;
- Accedere alle informazioni degli archi;
- Determinare tutti gli archi che partono da un vertice  $i$ ;
- Determinare tutti gli archi che arrivano a un vertice  $i$ ;
- Determinare tutti gli archi che incidono su un vertice  $i$ .

In letteratura sono presentate numerose proposte. Alcune permettono un efficiente accesso ai dati, ma sono dispendiose dal punto di vista dell'occupazione di memoria, altre forniscono efficaci compromessi.

La scelta della struttura dati più opportuna dipende principalmente dall'algoritmo che si deve implementare e dalle “risorse” a disposizione.

La matrice di adiacenza  $Q$  di un grafo non orientato semplice  $G = (V, E)$  è la matrice simmetrica  $|V| \times |V|$  con elementi:

$$q_{ij} = \begin{cases} 1 & \text{se } \{i, j\} \in E; \\ 0 & \text{altrimenti.} \end{cases}$$

Invece, la matrice di adiacenza  $Q$  di un grafo orientato semplice  $G = (V, A)$  è la matrice  $|V| \times |V|$ , non necessariamente simmetrica, con elementi:

$$q_{ij} = \begin{cases} 1 & \text{se } (i, j) \in A; \\ 0 & \text{altrimenti.} \end{cases}$$

Nelle figure 13.15 e 13.16 sono riportati due esempi di matrice di adiacenza per un grafo non orientato e uno orientato, rispettivamente. Si noti che per il grafo orientato ogni riga e colonna  $i$  definiscono l'insieme dei successori  $\Gamma^+(i)$  e dei predecessori  $\Gamma^-(i)$ . Per un grafo orientato la matrice è simmetrica e le righe e colonne definiscono entrambe  $\Gamma(i)$ .

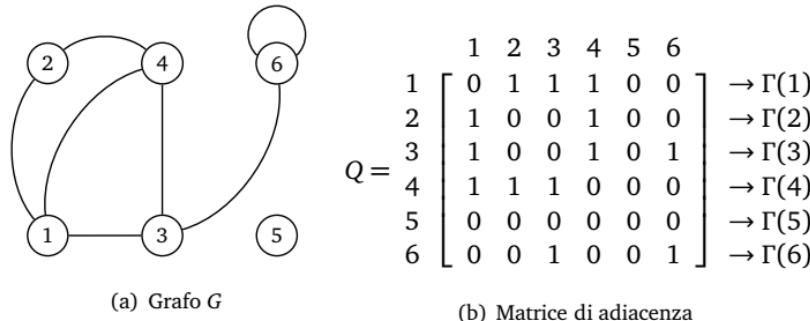


Figura 13.15: Esempio di matrice di adiacenza per un grafo non orientato.

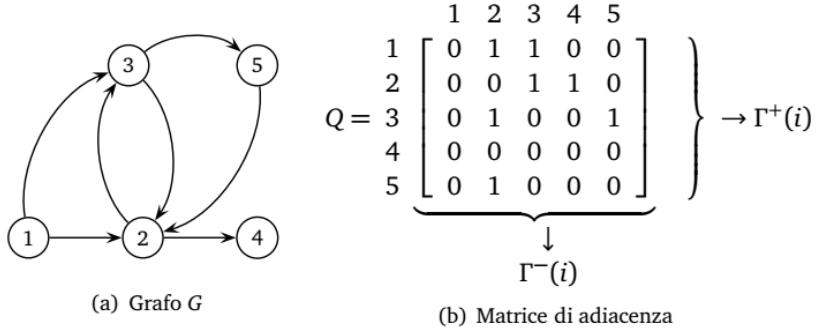


Figura 13.16: Esempio di matrice di adiacenza per un grafo orientato.

La *matrice di incidenza* vertici-lati  $D$  di un grafo non orientato  $G = (V, E)$  è la matrice  $|V| \times |E|$  con elementi:

$$d_{ik} = \begin{cases} 1 & \text{se il } k\text{-esimo lato è incidente nel vertice } i \text{ (i.e., } e_k = \{i, j\}\text{);} \\ 0 & \text{altrimenti.} \end{cases}$$

Invece, la matrice di incidenza vertici-archi  $D$  di un grafo orientato  $G = (V, A)$  è la matrice  $|V| \times |A|$  con elementi:

$$d_{ik} = \begin{cases} 1 & \text{se il } k\text{-esimo arco esce dal vertice } i \text{ (i.e., } a_k = (i, j)\text{);} \\ -1 & \text{se il } k\text{-esimo arco entra nel vertice } i \text{ (i.e., } a_k = (j, i)\text{);} \\ 0 & \text{se } i \text{ non è vertice terminale di } a_k. \end{cases}$$

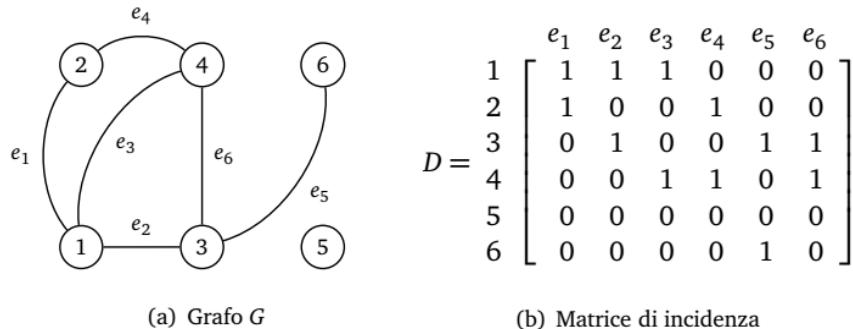


Figura 13.17: Esempio di matrice di incidenza per un grafo non orientato.

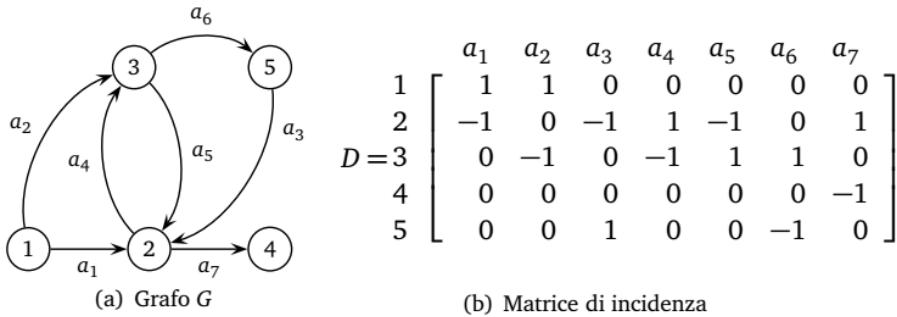


Figura 13.18: Esempio di matrice di incidenza per un grafo orientato.

Nelle figure 13.18 e 13.18 sono riportati due esempi di matrice di incidenza per un grafo non orientato e uno orientato, rispettivamente. Per un grafo non orientato la matrice ha sempre due "1" in ogni colonna (lato) e per ogni riga (vertice) un numero di "1" pari al numero di lati incidenti nel corrispondente vertice (denominato *grado del vertice* o "vertex degree"). Invece, per un grafo orientato la matrice ha sempre un "1" e un "-1" in ogni colonna (arco) e per ogni riga (vertice) un numero di "1" pari al numero di archi uscenti dal vertice (*grado di uscita* o *outdegree*) e un numero di "-1" pari al numero di archi entranti nel vertice (*grado di entrata* o *indegree*).

Le *liste di adiacenza* conservano per ogni vertice  $i$  la lista  $A(i)$  degli archi che partono da esso. Per cui, è necessario un vettore  $n$ -dimensionale ( $n = |V|$ ) *first*, dove  $first(i)$  memorizza il puntatore al primo elemento della lista  $A(i)$ . In figura 13.19 è riportato un esempio di liste di adiacenza per un grafo orientato in cui per ogni arco sono "salvate" una coppia di informazioni (per esempio, un costo e una capacità associate all'arco).

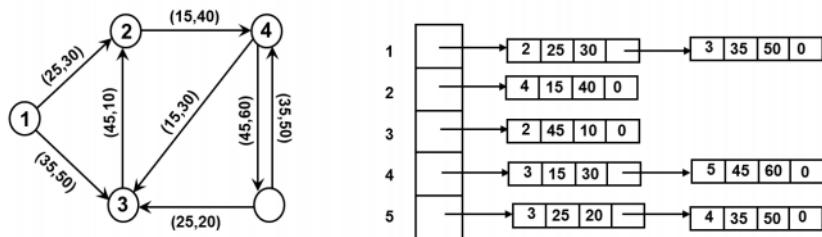


Figura 13.19: Esempio di liste di adiacenza per un grafo orientato.

Impiegando le liste di adiacenza si risparmia tempo calcolo e spazio di memoria. Però richiedono una "gestione" più complessa.

La rappresentazione *forward star* richiede di salvare le informazioni degli archi in un vettore  $m$ -dimensionale ( $m = |A|$ ). Gli archi devono essere ordinati per indice del vertice *iniziale* (*tail node*) crescente. Inoltre è necessario un vettore

$n$ -dimensionale ( $n = |V|$ ) di puntatori *point*, dove  $\text{point}(i)$  memorizza l'indice in cui sono salvate le informazioni del primo arco che *parte* dal vertice  $i$  nel corrispondente vettore. Gli archi che partono dal vertice  $i$  sono posizionati da  $\text{point}(i)$  fino a  $\text{point}(i + 1) - 1$ .

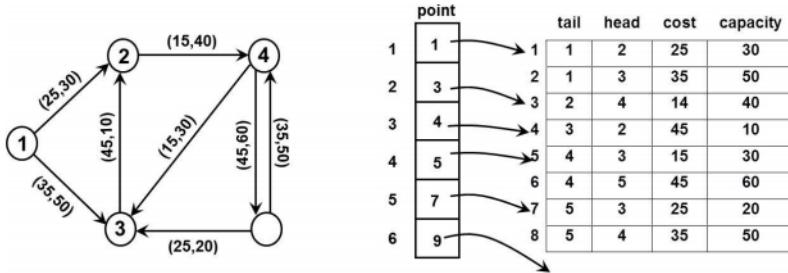


Figura 13.20: Esempio di forward star per un grafo orientato.

In figura 13.20 è riportato un esempio di forward star per lo stesso grafo orientato usato nell'esempio di figura 13.19.

La rappresentazione forward star consente di accedere in modo efficiente agli archi che partono da un determinato vertice  $i$ .

Nel caso sia necessario accedere efficacemente agli archi che arrivano a un determinato vertice  $i$  la forward star non permette un'equivalente performance. Quindi, nel caso l'algoritmo necessiti di accedere agli archi che arrivano a un determinato vertice  $i$  è necessario utilizzare la *backward star*. La rappresentazione backward star è analoga alla forward star tranne che:

- gli archi sono ordinati per indice del *vertice finale (head node)* crescente;
- il vettore  $\text{point}(i)$  memorizza l'indice in cui sono salvate le informazioni del primo arco che *arriva* al vertice  $i$  nel corrispondente vettore. Gli archi che arrivano al vertice  $i$  sono posizionati da  $\text{point}(i)$  fino a  $\text{point}(i + 1) - 1$ .

## 13.2 NEW: (Cammini minimi)

Consideriamo un grafo orientato  $G = (V, A)$  con  $n$  vertici e  $m$  archi in cui  $c_{ij}$  è il costo associato ad ogni arco  $(i, j) \in A$ . Il costo di un cammino da  $s \in V$  a  $t \in V$  è pari alla somma dei costi degli archi che lo compongono.

Il cammino di costo minimo (*cammino minimo*) da  $s$  a  $t$  è quello che, fra tutti i cammini da  $s$  a  $t$ , ha il costo più piccolo.

Nel caso in cui  $c_{ij} \geq 0$ , per ogni  $(i, j) \in A$ , il cammino minimo è elementare. Mentre, se alcuni dei costi  $c_{ij}$  sono negativi allora il grafo  $G$  può contenere circuiti di costo negativo. In questo caso il circuito di costo negativo può essere usato un numero infinito di volte per ridurre il costo.

Nel caso si voglia calcolare il *cammino minimo elementare* in un grafo in cui alcuni dei costi  $c_{ij}$  sono negativi è necessario imporre esplicitamente la

restrizione che il cammino passi attraverso ciascun vertice al massimo una sola volta. Purtroppo in presenza di cicli di costo negativo il problema è NP-Hard.

Esistono tuttavia casi particolari in cui non esistono sicuramente cicli di costo negativo e che possono essere risolti in tempo polinomiale, fra i quali: grafi aciclici; grafi con costi positivi.

### 13.2.1 Formulazione matematica

Definiamo per ogni arco  $(i, j) \in A$  la variabile decisionale  $x_{ij} \in \{0, 1\}$  che deve valere 1 se e solo se  $(i, j)$  viene scelto nel cammino, altrimenti vale 0.

Il problema del cammino minimo elementare da  $s$  a  $t$  ( $s \neq t$ ) può essere formulato come segue:

$$z_{SP} = \min \sum_{(i,j) \in A} c_{ij} x_{ij} \quad (13.1)$$

$$\text{s.t. } \sum_{j \in \Gamma^+(i)} x_{ij} - \sum_{j \in \Gamma^-(i)} x_{ji} = b_i, \quad i \in V \quad (13.2)$$

$$x_{ij} \in \{0, 1\}, \quad \forall (i, j) \in A \quad (13.3)$$

dove  $b_i = 1$ , se  $i = s$ ,  $b_i = -1$ , se  $i = t$ , e  $b_i = 0$ , se  $i \in V \setminus \{s, t\}$ . La funzione obiettivo (13.1) minimizza il costo degli archi in soluzione. Le equazioni (13.2) sono i cosiddetti vincoli di conservazione del flusso che prevedono l'immissione di una unità di flusso nel vertice  $s$ , l'assorbimento di una unità di flusso nel vertice  $t$ , mentre garantiscono che negli altri vertici il flusso entrante sia uguale al flusso uscente. I vincoli (13.3) impongono che le variabili decisionali siano binarie. In realtà potremo rilassare i vincoli di interezza (13.3) sostituendoli con i vincoli  $0 \leq x_{ij} \leq 1$ , perché la soluzione ottima sarà comunque intera (si veda la sezione sul problema dei cammini di costo minimo).

Nel caso possano esserci cicli di costo negativo è necessario aggiungere i seguenti vincoli:

$$\sum_{(i,j) \in A(S)} x_{ij} \leq |S| - 1, \quad \forall S \subseteq V, \quad S \neq \emptyset \quad (13.4)$$

dove  $A(S)$ ,  $S \subseteq V$ , è l'insieme degli archi con entrambi gli estremi in  $S$ , i.e.,  $A(S) = \{(i, j) \in A : i \in S, j \in S\}$ . Questi vincoli, che sono  $2^n - 1$ , impediscono il formarsi di *subtour* (sottocicli) e per questa ragione sono noti come vincoli di *subtour elimination*.

### 13.2.2 Assunzioni

In questa sezione descriviamo alcune assunzioni che saranno utili nel proseguo della sezione.

Innanzitutto si assume che tutti i costi degli archi  $c_{ij}$  siano interi e denotiamo con  $C$  il costo più alto, i.e.,  $C = \max\{c_{ij} : (i, j) \in A\}$ . Si noti che in linea di

principio tutti i costi “*razionali*” possono essere convertiti in interi, mentre i costi “*irrazionali*” (e.g.,  $\sqrt{2}, \pi, \dots$ ) non possono essere gestiti come interi.

Il grafo contiene un cammino diretto dal nodo  $s$  a ogni altro nodo. Per soddisfare questa assunzione si possono aggiungere degli archi artificiali con un costo “*sufficientemente*” grande.

Per alcuni algoritmi si assume che non esistano cicli di costo negativo, perché nel caso vi siano dei cicli di costo negativo, la soluzione ottima del problema sarebbe illimitata. Chiaramente questi algoritmi non possono essere utilizzati per grafi in cui vi sono cicli di costo negativo, perché non garantirebbero la soluzione e/o il corretto funzionamento.

Quando non è esplicitamente definito, si assume che il grafo sia orientato. Per soddisfare questa assunzione si può sostituire ogni arco non orientato (lato)  $\{i, j\}$  di costo  $c_{ij}$  con due archi diretti  $(i, j)$  e  $(j, i)$  entrambi di costo  $c_{ij}$ .

### 13.2.3 Distance label

Diversi algoritmi per calcolare i cammini minimi impiegano il vettore delle *distance label*. Per ogni vertice è definita una label  $d(i)$ , che rappresenta il costo di un qualche cammino diretto dal vertice sorgente  $s$  al nodo  $i$ . Le distance label sono un *upper bound* al costo del cammino minimo dal vertice sorgente  $s$  al vertice  $i$ .

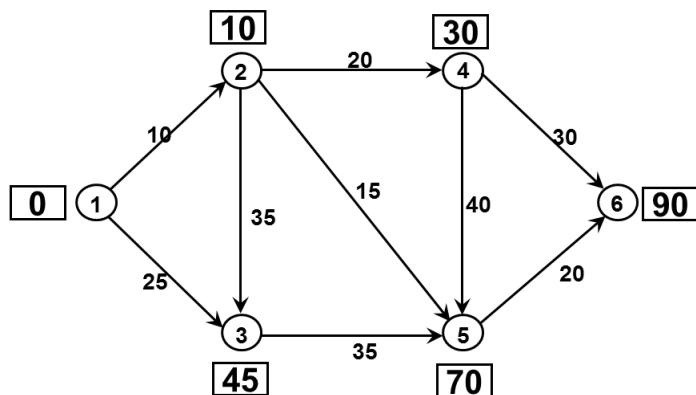


Figura 13.21: Esempio di grafo e corrispondenti distance label.

In Figura 13.21 è riportato un esempio dove il vertice di partenza del cammino è  $s = 1$  e la sua label è impostata a  $d(1) = 0$ . Le rimanenti label  $d(i)$  sono sicuramente degli upper bound al costo dei cammini necessari per raggiungere il vertice  $i$  dal vertice  $s$ . Infatti, per esempio, la distance label  $d(6) = 70$  corrisponde al costo del cammino  $P = (1, 2, 4, 5)$ .

### 13.2.4 Condizioni di ottimalità

Per definire quando le distance label  $d(i)$  rappresentano il costo del cammino di costo minimo sono necessarie delle condizioni di ottimalità.

**Lemma 13.1.** *Se le distance label  $d(i)$  rappresentano il costo del cammino di costo minimo dal vertice  $s$  al vertice  $i$ , allora devono soddisfare la seguente condizione:*

$$d(j) \leq d(i) + c_{ij} \quad \text{per ogni arco } (i, j) \in A. \quad (13.5)$$

**Dim.: 13.1.** *Se per qualche arco  $(i, j) \in A$  dovesse accadere che  $d(j) > d(i) + c_{ij}$ , allora la distance label  $d(j)$  non rappresenterebbe il costo del cammino di costo minimo dal vertice  $s$  al vertice  $j$  perché esisterebbe un cammino meno costoso che arriva dal vertice  $i$  con l'arco  $(i, j)$  (vedi figura 13.22).*

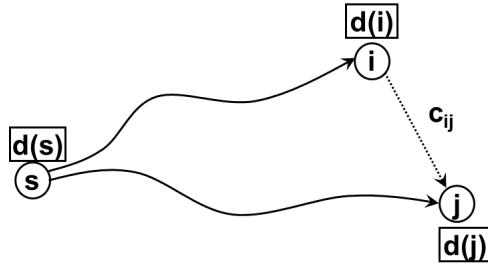


Figura 13.22: Condizioni di ottimalità: Teorema 13.1.

Il seguente teorema stabilisce le condizioni di ottimalità che sono alla base dei metodi *label setting* and *label correcting* che verranno introdotti in seguito.

**Teorema 13.1.** *Le distance label  $d(i)$  rappresentano il costo del cammino minimo se e solo se:*

$$\bar{c}_{ij} = c_{ij} + d(i) - d(j) \geq 0 \quad \text{per ogni arco } (i, j) \in A \quad (13.6)$$

**Dim.: 13.2.** *Per il Lemma 13.1 se le distance label  $d(i)$  rappresentano il costo del cammino minimo, allora  $\bar{c}_{ij} = c_{ij} + d(i) - d(j) \geq 0$  per ogni arco  $(i, j) \in A$ .*

*Ora si vuole dimostrare che se  $\bar{c}_{ij} = c_{ij} + d(i) - d(j) \geq 0$  per ogni arco  $(i, j) \in A$ , allora le distance label  $d(i)$  rappresentano il costo del cammino minimo. Dato un qualsiasi cammino  $P$  diretto dal nodo  $s$  al nodo  $k$ .*

$$\sum_{(i,j) \in P} \bar{c}_{ij} = \sum_{(i,j) \in P} (c_{ij} + d(i) - d(j))$$

*Si noti che dati due archi consecutivi  $(i, j)$  e  $(j, k)$  del path  $P$ ,  $-d(j)$  si semplifica con  $d(j)$ . Quindi, se si semplificano le distance label per tutti gli archi inclusi*

nel path  $P$ , si ha:

$$\sum_{(i,j) \in P} \bar{c}_{ij} = \left( \sum_{(i,j) \in P} c_{ij} \right) + d(s) - d(k)$$

Siccome  $d(s) = 0$  e  $\bar{c}_{ij} \geq 0$  per ogni arco  $(i,j) \in P$ , allora abbiamo:

$$d(k) \leq \sum_{(i,j) \in P} c_{ij} \quad (13.7)$$

Per cui  $d(k)$  è senz'altro un lower bound al costo di ogni cammino dal nodo  $s$  al nodo  $k$ . Dato che  $d(k)$  è anche la lunghezza di un qualche cammino da  $s$  a  $k$ , allora deve essere il costo del cammino minimo.

### 13.2.5 Algoritmi Label Correcting

Sulla base del Teorema 13.1, si può definire un algoritmo che determina i cammini di costo minimo correggendo le etichette fino a quando non soddisfano le condizioni di ottimalità.

Ipotizzando che il grafo orientato  $G(V, A)$  sia connesso e non abbia cicli di costo negativo (per esempio, ha costi  $c_{ij} \geq 0$  per ogni  $(i,j) \in A$ ), un algoritmo label correcting generico, che determina i cammini di costo minimo dal vertice  $s$  a tutti gli altri vertici, è riassunto nell'Algoritmo 5.

---

#### Algorithm 5 Label Correcting (Generico)

---

**Input:** Grafo orientato connesso senza cicli di costo negativo;  
**Output:** Cammini minimi da  $s$  a  $V \setminus \{s\}$  definiti da  $pred[j]$ ,  $\forall j \in V \setminus \{s\}$ ;

```

// Inizializzazione
for j = 1 to n do
    d[j] = ∞;
    pred[j] = -1;
end for
d[s] = 0;
// Ripete finché c'è una condizione violata
while (∃(i, j) ∈ A : d[j] > d[i] + cij) do
    d[j] = d[i] + cij;
    pred[j] = i;
end while

```

---

Se nel grafo  $G(V, A)$  esistono cicli di costo negativo, l'algoritmo label correcting generico qui proposto non termina mai. Se invece il grafo non ha cicli di costo negativo l'algoritmo termina sempre. In particolare, ad ogni iterazione l'algoritmo deve considerare tutti gli archi con una complessità pari a  $O(m)$ . Il numero di iterazioni è  $O(2nC)$  perché, sebbene all'inizio dell'algoritmo  $d(j) = \infty$  per ogni  $j \in V \setminus \{s\}$ , ogni distance label finita  $d(i)$  è limitata superiormente dal

valore  $nC$  e limitata inferiormente dal valore  $-nC$ , inoltre ad ogni iterazione una distance label diminuisce di almeno un'unità mentre nessuna distance label aumenta. La complessità computazionale complessiva è pari a  $O(2nmC)$ , quindi è pseudo-polinomiale.

La complessità può essere diminuita a  $O(nm)$ . Per diminuire la complessità costruiamo un nuovo algoritmo label correcting basato sul seguente teorema.

**Teorema 13.2.** *Se ad ogni iterazione si esaminano tutti gli archi uno alla volta, verificando le condizioni di ottimalità e aggiornando le distance label quando necessario, allora dopo  $k$  iterazioni saranno determinati tutti i cammini minimi contenenti al più  $k$  archi.*

**Dim.: 13.3.** *Si dimostra per induzione rispetto a  $k$  (vedi Figura 13.23).*

Insieme di nodi per i quali sono stati calcolati cammini minimi contenenti al più  $k$  archi

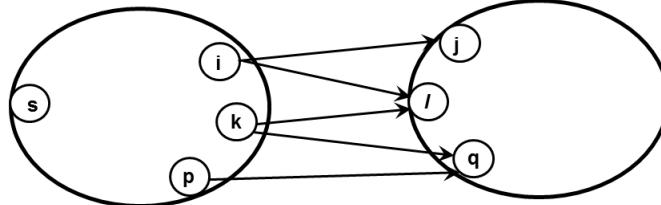


Figura 13.23: Dimostrazione per induzione per il Teorema 13.2.

Il Teorema 13.2 garantisce che al termine dell'iterazione  $k = n - 1$ , dopo che sono state aggiornate le label dopo aver considerato tutti gli archi, tutti i cammini di costo minimo di lunghezza al più  $n - 1$  sono stati generati. Siccome, in assenza assenza di cicli di costo negativo, i cammini minimi non possono avere più di  $n - 1$  archi, non è necessario svolgere altre iterazioni. Se al termine dell'iterazione  $k = n - 1$  ci sono ancora delle label che per qualche arco violano delle condizioni di ottimalità, significa che esiste un ciclo di costo negativo, che può essere identificato con  $\text{pred}[j]$ .

Nel caso in cui nel ciclo principale non vengano trovate condizioni di ottimalità violate, allora nessuna distance label  $d(i)$  verrà aggiornata. Però, se nessuna distance label verrà modificata allora anche nell'iterazione successiva nessuna condizione di ottimalità sarà violata. Quindi, nell'algoritmo si può permettere un'uscita anticipata quando nessuna distance label potrà essere ulteriormente modificata.

L'algoritmo che ne deriva è noto come Algoritmo di Bellman-Ford ed è riassunto nell'Algoritmo 6.

### 13.2.6 Algoritmo di Dijkstra

L'algoritmo di Dijkstra calcola i cammini minimi dal vertice  $s \in V$  a ogni altro vertice  $t \in V \setminus \{s\}$  solo se i costi degli archi sono *non negativi* (i.e.,  $c_{ij} \geq 0$ ,  $\forall (i, j) \in A$ ). L'algoritmo di Dijkstra si basa sul Teorema 13.3.

---

**Algorithm 6 Bellman-Ford**

---

**Input:** Grafo orientato;

**Output:** Cammini minimi da  $s$  a  $V \setminus \{s\}$  definiti da  $pred[j]$ ,  $\forall j \in V \setminus \{s\}$ ;

```

// Inizializzazione
for j = 1 to n do
    d[j] = ∞;
    pred[j] = -1;
end for
d[s] = 0;
// Controlla gli archi per  $n - 1$  iterazioni
for k = 1 to  $n - 1$  do
    update=False;
    for (i, j) ∈ A do
        if ( $d[j] > d[i] + c_{ij}$ ) then
            d[j] = d[i] +  $c_{ij}$ ;
            pred[j] = i;
            update=True;
        end if
    end for
    if (update==False) then
        Esci dal ciclo;
    end if
end for
// Controlla se ci sono cicli di costo negativo
for (i, j) ∈ A do
    if ( $d[j] > d[i] + c_{ij}$ ) then
        Il grafo contiene cicli di costo negativo;
    end if
end for

```

---

**Teorema 13.3.** *Dato un sottoinsieme  $S \subseteq V$  che include  $s$  (i.e.,  $s \in S$ ), sia  $L_i$  il costo del cammino di costo minimo da  $s$  al vertice  $i$ , per ogni vertice  $i \in S$ . Se  $(v, h) = \operatorname{argmin}\{L_i + c_{ij} : (i, j) \in \delta^+(S)\}$ , allora  $L_v + c_{vh}$  rappresenta il costo del cammino minimo da  $s$  ad  $h$ .*

**Dim.: 13.4.**  *$L_v + c_{vh}$  rappresenta il costo di un cammino da  $s$  ad  $h$ . Si consideri un altro cammino  $P$  che termina in  $h$ . Sia  $(i, j) \in P \cap \delta^+(S)$  e si partizionino  $P$  in  $P_1 \cup \{(i, j)\} \cup P_2$ , dove  $P_1$  e  $P_2$  sono due cammini da  $s$  ad  $i$  e da  $j$  ad  $h$ , rispettivamente (vedi Figura 13.24). Si ha:*

$$\begin{aligned} C(P) &= \underbrace{c(P_1)}_{\geq L_i} + c_{ij} + \underbrace{C(P_2)}_{\geq 0} \geq L_i + c_{ij} \geq L_v + c_{vh}. \end{aligned}$$

Per cui,  $L_v + c_{vh}$  rappresenta il costo del cammino di costo minimo da  $s$  ad  $h$ .

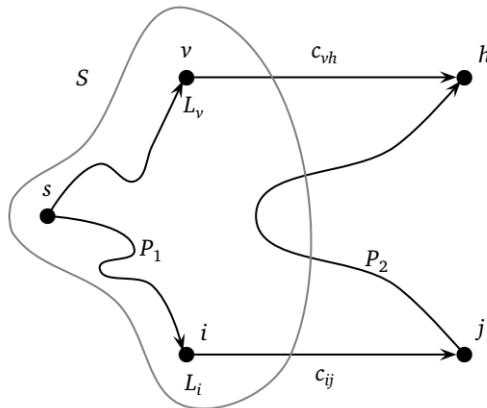


Figura 13.24: Dimostrazione del Teorema 13.3.

Il Teorema 13.3 suggerisce un algoritmo iterativo, noto come Algoritmo di Dijkstra, per la determinazione dei cammini minimi da  $s \in V$  ad ogni  $t \in V$ . Nell'algoritmo di Dijkstra l'insieme  $S$  può essere interpretato come l'insieme dei vertici *permanent* le cui label rappresentano i costi del cammino di costo minimo. Mentre, il vertice  $h$  dato da  $(v, h) = \operatorname{argmin}\{L_i + c_{ij} : (i, j) \in \delta^+(S)\}$  rappresenta il nuovo vertice che entra nell'insieme  $S$  dei vertici permanenti. Il vertice  $h$  oltre a entrare nell'insieme  $S$  viene anche *espanso* per eventualmente aggiornare le label dei vertici successori. Una prima versione base dell'Algoritmo di Dijkstra è proposto nell'Algoritmo 7.

L'algoritmo 7 calcola i cammini minimi dal vertice  $s$  a tutti gli altri vertici del grafo. Si noti che se si è interessati a calcolare i cammini minimi dal vertice  $s$  a un sottoinsieme di vertici  $T \subseteq V$  (eventualmente contenente un solo vertice  $t$ , i.e.,  $T = \{t\}$ ), possiamo fermare l'Algoritmo di Dijkstra non appena le etichette di tutti i vertici di  $T$  sono in  $S$ , quindi permanenti.

---

**Algorithm 7 Dijkstra (1<sup>a</sup> versione)**

---

**Input:** Grafo orientato con costi  $\{c_{ij}\}$  non-negativi;  
**Output:** Cammini minimi da  $s$  a  $V \setminus \{s\}$  definiti da  $pred[j]$ ,  $\forall j \in V \setminus \{s\}$ ;

```

 $S = \{s\};$ 
 $L[s] = 0;$ 
 $pred[s] = s;$ 
while ( $|S| \neq n$ ) do
    if ( $\delta^+(S) \neq \emptyset$ ) then
         $(v, h) = argmin\{L[i] + c_{ih} : (i, h) \in \delta^+(S)\};$ 
         $L[h] = L[v] + c_{vh};$ 
         $pred[h] = v;$ 
         $S = S \cup \{h\};$ 
    else
        Grafo  $G$  disconnesso;
        STOP;
    end if
end while
```

---

La complessità computazionale dell'Algoritmo 7 è pari a  $O(nm)$ . Ma è possibile ottenere una complessità  $O(n^2)$  se ad ogni iterazione si sfruttano opportunamente le informazioni già acquisite nelle iterazioni precedenti. In particolare, l'algoritmo di Dijkstra potrebbe utilizzare le seguenti strutture dati definite per ogni  $j \in V$ :

- $flag[j] = \begin{cases} 1 & \text{se } j \in S \\ 0 & \text{altrimenti} \end{cases}$
- $L[j] = \begin{cases} \text{costo del cammino da } s \text{ a } j, & \text{se } j \in S; \\ min\{L[i] + c_{ij} : i \in S\}, & \text{se } j \notin S; \end{cases}$
- $pred[j] = \begin{cases} \text{predecessore di } j \text{ nel cammino da } s \text{ a } j, & \text{se } j \in S \\ argmin\{L[i] + c_{ij} : i \in S\}, & \text{se } j \notin S \end{cases}$

La versione revisionata dell'Algoritmo di Dijkstra di complessità  $O(n^2)$  è proposta nell'Algoritmo 8.

La complessità dell'Algoritmo di Dijkstra può essere ulteriormente ridotta utilizzando delle opportune strutture dati per mantenere *ordinate* le label dei vertici non ancora inseriti nell'insieme dei permanenti  $S$  con l'obiettivo di rendere più efficiente la selezione del nodo. Tra le diverse opzioni vi è l'impiego di una *heap* e se si utilizza una Fibonacci Heap, si ottiene una complessità  $O(m+n \log n)$ .

Un'altra opzione molto interessante è l'utilizzo dell'approccio di Dial che permette di avere una complessità  $O(m+nC)$ , che nonostante sia pseudopolino-

---

**Algorithm 8 Dijkstra (versione  $O(n^2)$ )**

---

**Input:** Grafo orientato connesso con costi  $\{c_{ij}\}$  non-negativi;  
**Output:** Cammini minimi da  $s$  a  $V \setminus \{s\}$  definiti da  $pred[j]$ ,  $\forall j \in V \setminus \{s\}$ ;

```

// Inizializzazione
for j = 1 to n do
    flag[j] = 0;
    pred[j] = s;
    L[j] = c_{sj};
end for
flag[s] = 1;
L[s] = 0;
for k = 1 to n - 1 do
    // Individua  $h = argmin\{L[j] : j \notin S\}$ 
    min = +∞;
    for j = 1 to n do
        if (flag[j] = 0) and ( $L[j] < min$ ) then
            min = L[j];
            h = j;
        end if
    end for
    // Aggiorna  $S = S \cup \{h\}$ 
    flag[h] = 1;
    // Aggiorna  $L[j]$  e  $pred[j]$  per ogni  $j \notin S$ 
    for j = 1 to n do
        if (flag[j] = 0) and ( $L[h] + c_{hj} < L[j]$ ) then
            L[j] = L[h] + c_{hj};
            pred[j] = h;
        end if
    end for
end for
end for

```

---

miale raramente raggiunge il caso peggiore. Inoltre, per grafi in cui  $C$  è piccolo anche il caso peggiore risulta competitivo.

In Figura 13.25 è riportato un esempio di applicazione dell'algoritmo di Dijkstra, in cui per ogni vertice è riportata la coppia  $[pred[j], L[j]]$ , per ogni  $j \in V$ . Il primo passo dell'algoritmo consiste nell'inizializzazione delle label  $L[j]$  e del predecessore  $pred[j]$  per ogni vertice  $j \in V$ , come mostrato in Figura 13.25a. L'insieme  $S$  dei vertici permanenti è inizializzato come  $S = \{s\} = \{1\}$ . Dopodiché, in Figura 13.25b l'algoritmo esegue la prima iterazione includendo il vertice 8 nell'insieme dei vertici permanenti  $S$ , perché è il vertice in  $V \setminus S$  con la label  $L[j]$  più piccola. La prima iterazione termina con l'espansione del vertice 8, ossia si aggiorna la label  $L[j]$  e  $pred[j]$  per tutti i vertici  $j$  successori del vertice 8. Nella seconda iterazione, in Figura 13.25c entra in  $S$  il vertice 7 che poi viene espanso. Poi nelle iterazioni successive, nelle Figure 13.25d-13.25g, entrano in  $S$  nell'ordine i vertici 6, 2, 5, 3 e 4 che di volta in volta vengono espansi. In Figura 13.25h è riportato il risultato finale in cui i cammini di costo minimo sono evidenziati. In Figura 13.26 è riportata la tabella che illustra come varia il contenuto delle strutture dati usate dall'algoritmo di Dijkstra man mano che procede con le iterazioni.

### 13.2.7 Algoritmo di Floyd-Warshall

I cammini di costo minimo fra tutte le coppie di vertici di un grafo con costi non-negativi possono essere calcolati eseguendo  $n$  volte l'Algoritmo di Dijkstra utilizzando ad ogni esecuzione come vertice iniziale  $s$  uno degli  $n$  vertici del grafo. Ad ogni esecuzione dell'Algoritmo di Dijkstra si calcola la riga  $s$  della matrice di ordine  $n$  contenente il costo del cammino di costo minimo da  $s$  a tutti gli altri vertici. La complessità dell'algoritmo risultante è  $O(n^3)$  nel caso si usi l'Algoritmo 8.

Un metodo alternativo è quello utilizzato dall'algoritmo di Floyd-Warshall che ha sempre complessità pari a  $O(n^3)$ , ma si applica a qualsiasi grafo, compresi quelli con costi negativi, ed è in grado di riconoscere circuiti di costo negativo.

L'algoritmo di Floyd-Warshall si applica a un grafo orientato definito dalla matrice quadrata di ordine  $n$  dei costi  $[c_{ij}]$ . La sua implementazione richiede due matrici quadrate di ordine  $n$ : la matrice  $U$  per memorizzare i costi dei cammini minimi e la matrice  $Pred$  per salvare i vertici predecessori nei cammini necessari per ricostruire i cammini di costo minimo.

L'idea alla base dell'Algoritmo di Floyd-Warshall, descritto nell'Algoritmo 9, è molto semplice. La matrice  $U$  e  $Pred$  sono inizializzati con i cammini di un solo arco tra ogni coppia di vertici  $i$  e  $j$ , che hanno costo  $c_{ij}$  e predecessore  $pred[i, j] = i$ . Si noti che se l'arco  $(i, j)$  non dovesse esistere potremo impostare il suo costo a infinito, i.e.,  $c_{ij} = \infty$  (nella pratica un valore “sufficientemente grande”), inoltre,  $c_{ii} = 0$  per ogni  $i \in V$ . Dopodiché, il meccanismo di funzionamento dell'Algoritmo di Floyd-Warshall si basa sul Teorema 13.4.

**Teorema 13.4.** *Per ogni coppia di vertici  $i$  e  $j$  del grafo  $G(V, A)$ ,  $u_{ii}$  sia il costo di un qualche cammino da  $i$  a  $j$ .*

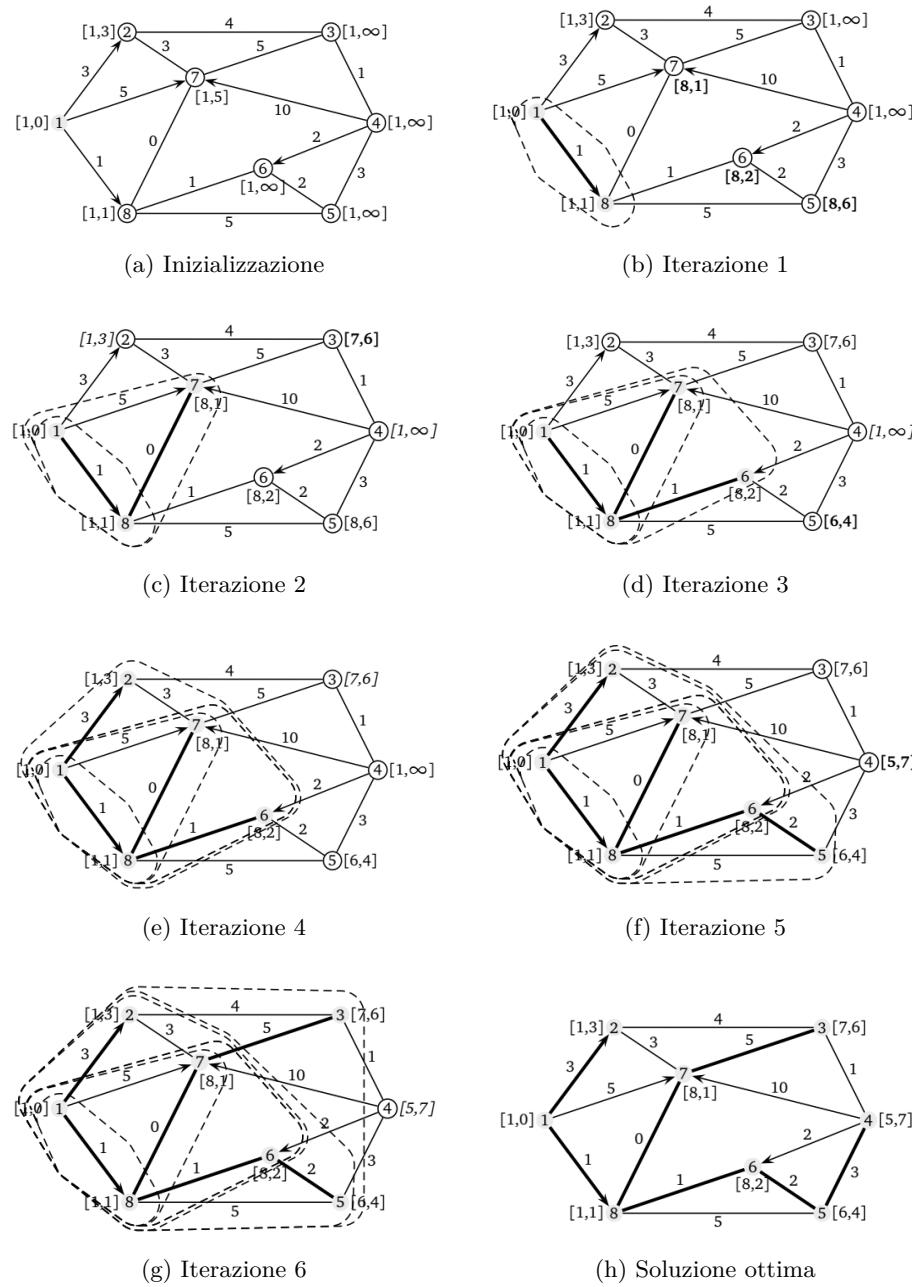


Figura 13.25: Un esempio di esecuzione dell'Algoritmo di Dijkstra

$S$	$L[j]$								$pred[j]$							
	2	3	4	5	6	7	8	2	3	4	5	6	7	8		
{1}	3	$\infty$	$\infty$	$\infty$	$\infty$	5	<span style="border: 1px solid black; padding: 2px;">1</span>	1	-	-	-	-	1	1		
{1,8}	3	$\infty$	$\infty$	6	2	<span style="border: 1px solid black; padding: 2px;">1</span>	1	1	-	-	8	8	8	1		
{1,8,7}	3	6	$\infty$	6	<span style="border: 1px solid black; padding: 2px;">2</span>	1	1	1	7	-	8	8	8	1		
{1,8,7,6}	<span style="border: 1px solid black; padding: 2px;">3</span>	6	$\infty$	4	2	1	1	1	7	-	6	8	8	1		
{1,8,7,6,2}	3	6	$\infty$	<span style="border: 1px solid black; padding: 2px;">4</span>	2	1	1	1	7	-	6	8	8	1		
{1,8,7,6,2,5}	3	<span style="border: 1px solid black; padding: 2px;">6</span>	7	4	2	1	1	1	7	5	6	8	8	1		
{1,8,7,6,2,5,3}	3	6	<span style="border: 1px solid black; padding: 2px;">7</span>	4	2	1	1	1	7	5	6	8	8	1		
{1,8,7,6,2,5,3,4}	3	6	7	4	2	1	1	1	7	5	6	8	8	1		

Figura 13.26: Tabella strutture dati usate dall’Algoritmo di Dijkstra

---

**Algorithm 9 Floyd-Warshall**

---

**Input:** Grafo orientato definito dalla matrice dei costi  $[c_{ij}]$ ;**Output:** Matrici  $[u_{ij}]$  e  $[pred[i, j]]$ ;

// Inizializzazione

for  $i = 1$  to  $n$  do    for  $j = 1$  to  $n$  do         $u_{ij} = c_{ij}$ ;         $pred[i, j] = i$ ;

end for

end for

// Operazione triangolare su  $h$ for  $k = 1$  to  $n$  do    for  $i = 1$  to  $n$  do        for  $j = 1$  to  $n$  do            if  $(u_{ik} + u_{kj} < u_{ij})$  then                 $u_{ij} = u_{ik} + u_{kj}$ ;                 $pred[i, j] = pred[k, j]$ ;

end if

end for

end for

end for

for  $i = 1$  to  $n$  do    if  $(u_{ii} < 0)$  then

STOP, circuiti negativi;

end if

end for

end for

---

I costi  $[u_{ij}]$  rappresentano i cammini di costo minimo tra tutte le coppie di vertici del grafo  $G$  se e solo se soddisfano la seguente condizione di ottimalità:

$$u_{ij} \leq u_{ik} + u_{kj}, \quad \text{per tutti } i, j \text{ e } k.$$

Si noti che ciascun valore  $u_{ij}$  calcolato all'iterazione  $k$ -esima dell'Algoritmo di Floyd-Warshall rappresenta il costo del cammino di costo minimo dal vertice  $i$  al vertice  $j$  usando come vertici *interni* al cammino i vertici dell'insieme  $\{1, 2, \dots, k\}$ . Al termine dell'algoritmo, per ogni  $i, j \in V$ ,  $u_{ij}$  rappresenta il costo del cammino di costo minimo da  $i$  a  $j$  mentre  $\text{pred}[i, j]$  rappresenta il predecessore di  $j$  nel cammino di costo minimo da  $i$  a  $j$ .

Se  $u_{ii} < 0$  allora esiste un circuito di costo negativo, che può essere ricostruito a partire da  $\text{pred}[i, i]$ .

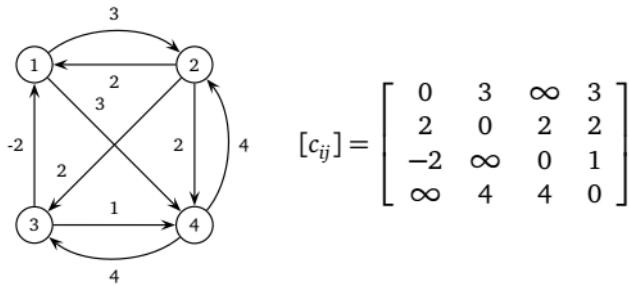


Figura 13.27: Grafo usato nell'esempio dell'Algoritmo di Floyd-Warshall.

In Figura 13.27 è riportato un grafo orientato e la corrispondente matrice dei costi  $[c_{ij}]$ . L'algoritmo di Floyd-Warshall inizializza le matrici  $U$  e  $Pred$  come illustrato in Figura 13.28. Dopotutto, l'algoritmo svolge le quattro iterazioni, illustrate in Figura 13.29, che consentiranno di ottenere le matrici  $U$  e  $Pred$  che forniranno i cammini di costo minimo per tutte le coppie di vertici  $i$  e  $j$  del grafo.

$u_{ij}$				$\text{pred}[i,j]$			
0	3	$\infty$	3	1	1	1	1
2	0	2	2	2	2	2	2
-2	$\infty$	0	1	3	3	3	3
$\infty$	4	4	0	4	4	4	4

Figura 13.28: Inizializzazione delle matrici  $U$  e  $Pred$  con l'Algoritmo di Floyd-Warshall.

A ogni iterazione  $k$ , per ogni coppia di vertici  $i$  e  $j$  deve essere verificato se  $u_{ij} > u_{ik} + u_{kj}$  e nel qual caso è necessario aggiornare il costo del cammino da  $i$  a  $j$  (i.e.,  $u_{ij} = u_{ik} + u_{kj}$ ) e il predecessore (i.e.,  $\text{pred}[i, j] = \text{pred}[k, j]$ ) per consentire poi la ricostruzione del cammino.

$u_{ij}$	$\Rightarrow$	$u_{ij}$
0 3 $\infty$ 3		0 3 $\infty$ 3
2 0 2 2		2 0 2 2
-2 $\infty$ 0 1		-2 1 0 1
$\infty$ 4 4 0		$\infty$ 4 4 0

$pred[i,j]$	$\Rightarrow$	$pred[i,j]$
1 1 1 1		1 1 1 1
2 2 2 2		2 2 2 2
3 3 3 3		3 1 3 3
4 4 4 4		4 4 4 4

(a) Iterazione  $k = 1$ 

$u_{ij}$	$\Rightarrow$	$u_{ij}$
0 3 $\infty$ 3		0 3 5 3
2 0 2 2		2 0 2 2
-2 1 0 1		-2 1 0 1
$\infty$ 4 4 0		6 4 4 0

$pred[i,j]$	$\Rightarrow$	$pred[i,j]$
1 1 1 1		1 1 2 1
2 2 2 2		2 2 2 2
3 1 3 3		3 1 3 3
4 4 4 4		2 4 4 4

(b) Iterazione  $k = 2$ 

$u_{ij}$	$\Rightarrow$	$u_{ij}$
0 3 5 3		0 3 5 3
2 0 2 2		0 0 2 2
-2 1 0 1		-2 1 0 1
6 4 4 0		2 4 4 0

$pred[i,j]$	$\Rightarrow$	$pred[i,j]$
1 1 2 1		1 1 2 1
2 2 2 2		3 2 2 2
3 1 3 3		3 1 3 3
2 4 4 4		3 4 4 4

(c) Iterazione  $k = 3$ 

$u_{ij}$	$\Rightarrow$	$u_{ij}$
0 3 5 3		0 3 5 3
0 0 2 2		0 0 2 2
-2 1 0 1		-2 1 0 1
2 4 4 0		2 4 4 0

$pred[i,j]$	$\Rightarrow$	$pred[i,j]$
1 1 2 1		1 1 2 1
3 2 2 2		3 1 3 3
3 1 3 3		3 4 4 4
3 4 4 4		3 4 4 4

(d) Iterazione  $k = 4$ 

Figura 13.29: Un esempio di esecuzione dell'Algoritmo di Floyd-Warshall

Data la soluzione ottenuta al termine dell'ultima iterazione, se si vuole conoscere il cammino di costo minimo da 4 a 1, sappiamo che ha costo 2 perché  $u_{41} = 2$ , mentre  $\text{pred}[4, 1] = 3$  indica che il cammino termina con l'arco  $(3, 1)$  e  $\text{pred}[4, 3] = 4$  che il cammino inizia con l'arco  $(4, 3)$ . In generale, data la coppia  $i$  e  $j$  di cui vogliamo conoscere il cammino di costo minimo, l'ultimo arco è  $(g, j)$  dove  $g = \text{pred}[i, j]$ , il penultimo arco è  $(h, g)$  dove  $h = \text{pred}[i, g]$  e così via fino a quando  $\text{pred}$  restituisce  $i$ .

### 13.3 NEW: (Alberi di copertura)

Sia  $G = (V, E)$  un grafo non orientato连通的 con  $n$  vertici e  $m$  lati e sia  $c_e$  o  $c_{ij}$  il costo associato ad ogni lato  $e = \{i, j\} \in E$ .

Dato un albero completo di  $G$  definito dai vertici di  $V$  e dal sottoinsieme di lati  $T \subseteq E$ , il costo  $c(T)$  dell'albero  $G_T(V, T)$  è dato dalla somma dei costi degli lati che lo compongono:

$$c(T) = \sum_{e \in T} c_e$$

Spesso l'albero completo è anche chiamato *albero di copertura*, *albero di supporto*, oppure *albero coprente*.

Il *Problema dell'Albero di Copertura di Costo Minimo (Shortest Spanning Tree, SST)* consiste nell'individuare, fra i diversi alberi completi di  $G$ , un albero  $G_T = (V, T)$  di costo  $c(T)$  minimo.

#### 13.3.1 Applicazioni

Un esempio di applicazione è il disegno di una rete di comunicazione in cui, dato un grafo non orientato连通的  $G$  che rappresenta una rete dei possibili collegamenti fra  $n$  città, si vuole determinare la rete di costo minimo per connettere le  $n$  città. La rete deve consentire che ogni città sia collegata direttamente con almeno un'altra città e deve esistere un cammino che collega ogni coppia di città.

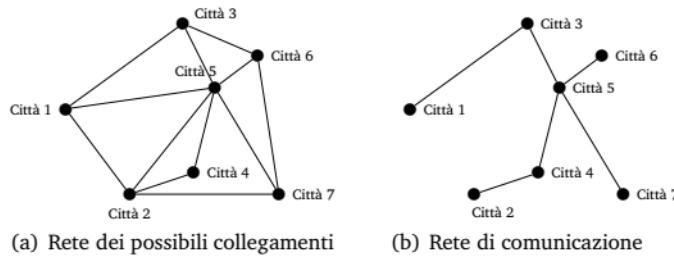


Figura 13.30: Disegno di una rete di comunicazione

In Figura 13.30 è riportato un esempio in cui dato il grafo dei possibili collegamenti tra le città (Figura 13.30a) viene mostrato un possibile albero di copertura che consente di collegare tutte le città (Figura 13.30b).

### 13.3.2 Formulazione matematica

Dato un grafo non orientato  $G = (V, E)$  con  $n$  vertici ed  $m$  lati, un suo albero completo  $G' = (V, A')$  è definito da una delle seguenti definizioni equivalenti:

- $G'$  è *connesso* e *non ha cicli*;
- $G'$  ha  $n - 1$  *lati* e *non ha cicli*;
- $G'$  è *connesso* e contiene esattamente  $n - 1$  *lati*.

Queste definizioni sono utili per definire i vincoli da includere nel modello matematico del problema. Inoltre, per formulare matematicamente il problema si deve definire per ogni lato  $e \in E$  la variabile decisionale:

$$x_e = \begin{cases} 1 & \text{se il lato } e \text{ viene scelto nell'albero minimo;} \\ 0 & \text{altrimenti.} \end{cases}$$

Una formulazione matematica per il problema è la seguente:

$$(SST) \quad z_{SST} = \min \sum_{e \in E} c_e x_e \quad (13.8)$$

$$\text{s.t. } \sum_{e \in E} x_e = n - 1, \quad (13.9)$$

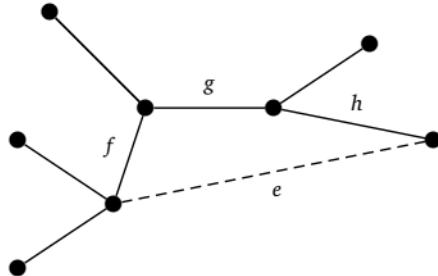
$$\sum_{e \in E(S)} x_e \leq |S| - 1, \quad \forall S \subseteq V, |S| \geq 3 \quad (13.10)$$

$$x_e \in \{0, 1\}, \quad \forall e \in E \quad (13.11)$$

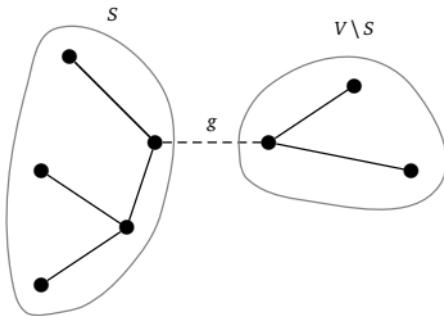
La funzione obiettivo (13.8) minimizza il costo dell'albero di copertura. Il vincolo (13.9) impone che siano scelti esattamente  $n - 1$  lati. Mentre, i vincoli (13.10) garantiscono l'assenza di cicli nel grafo parziale  $G_T = (V, T)$  definito dalla soluzione, i.e.,  $T = \{e \in E : x_e = 1\}$ . Il numero dei vincoli (13.10) è  $O(2^n)$ .

### 13.3.3 Condizioni di ottimalità

Dato un albero  $G_T = (V, T)$  ed un lato  $e \notin T$ , se aggiungiamo il lato  $e$  all'albero otteniamo un *ciclo*. Si denota con  $C(T, e)$  l'insieme dei lati del ciclo contenuto in  $T \cup \{e\}$  (nell'esempio  $C(T, e) = (e, f, g, h)$ ). In Figura 13.31 è riportato un esempio in cui si è aggiunto il lato  $e$  nell'albero ottenendo il ciclo contenente i lati  $\{e, f, g, h\}$ .

Figura 13.31: Ciclo generato dall'inserimento del lato  $e$ 

Dato un albero  $G_T = (V, T)$  ed un suo lato  $g \in T$ , se eliminiamo il lato  $g$  dall'albero otteniamo due alberi, i cui vertici appartengono all'insieme  $S$  e all'insieme  $V \setminus S$ . Gli insiemi  $S$  e  $V \setminus S$  determinano un *taglio*  $\delta(S) = \{\{i, j\} \in E : i \in S, j \in V \setminus S \text{ oppure } j \in S, i \in V \setminus S\}$ . In Figura 13.32 è riportato un esempio in cui è stato eliminato il lato  $g$  dall'albero ottenendo due alberi che determinano i due insiemi  $S$  e  $V \setminus S$  che definiscono un taglio.

Figura 13.32: Taglio generato dall'eliminazione del lato  $g$ 

Qui di seguito presentiamo due condizioni di ottimalità equivalenti note come *Cut Optimality Conditions* e *Path Optimality Conditions*.

**Teorema 13.5 (Cut Optimality Conditions).** L'albero di copertura  $G_T = (V, T)$  è di costo minimo se e solo se soddisfa le seguenti condizioni di ottimalità: per ogni lato appartenente all'albero  $e \in T$ ,  $c_e \leq c_g$  per ogni lato  $g$  appartenente al taglio  $S$  e  $V \setminus S$  generato dall'eliminazione del lato  $e$ .

**Dim.: 13.5. (a)** Se l'albero di copertura è di costo minimo, allora deve soddisfare le condizioni di ottimalità.

Se per un lato  $e \in T$  accadesse che  $c_e > c_g$  per un lato  $g$  appartenente al taglio  $S$  e  $V \setminus S$  generato dall'eliminazione del lato  $e$ , allora basterebbe sostituire il lato  $e$  con  $g$  in  $T$  per ottenere un albero di copertura di costo più basso contraddicendo l'ipotesi iniziale.

**(b)** Se l'albero di copertura rispetta le condizioni di ottimalità, allora è un albero di copertura di costo minimo.

Supponiamo che l'albero di copertura  $T$  non sia quello di costo minimo e che quello ottimo sia  $T^*$ . Quindi, esiste un lato  $e \in T$  che non è nell'albero di copertura di costo minimo, i.e.,  $e \notin T^*$ .

Eliminare il lato  $e$  da  $T$  crea un taglio corrispondente ai due alberi così ottenuti, mentre aggiungere il lato  $e$  all'albero ottimo  $T^*$  crea un ciclo (vedi Figura 13.33).

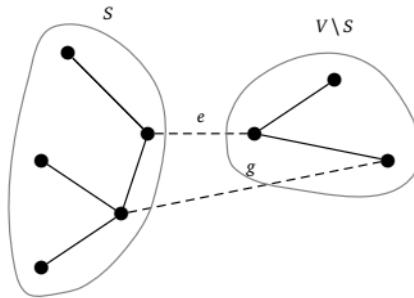


Figura 13.33: Esempio usato nelle dimostrazioni dei Teoremi 13.5 e 13.6

L'albero  $T$  soddisfa le condizioni di ottimalità, quindi  $c_e \leq c_g$  per tutti i lati del taglio compreso il lato  $g$  che assieme al lato  $e$  crea un ciclo nell'albero ottimo  $T^*$  (vedi Figura 13.33). Però, se  $T^*$  è l'albero ottimo  $c_e \geq c_g$ , altrimenti potremo trovare un albero con costo più basso scambiando i lati. Per cui,  $c_e = c_g$ . Se ripetiamo questo processo per tutti i lati di  $T$  che non sono in  $T^*$  otterremo lo stesso risultato. Quindi, l'albero di copertura  $T$  è di costo minimo.

Il Teorema 13.5 sarà alla base dell'algoritmo di Prim-Dijkstra presentato nella Sezione 13.3.4, mentre il successivo Teorema 13.6 sarà alla base dell'algoritmo di Kruskal descritto nella Sezione 13.3.5.

**Teorema 13.6 (Path Optimality Conditions).** L'albero di copertura  $G_T = (V, T)$  è di costo minimo se e solo se soddisfa le seguenti condizioni di ottimalità: per ogni lato  $g$  non appartenente all'albero (i.e.,  $g \notin T$ ),  $c_g \geq c_e$  per ogni lato  $e$  contenuto nel path in  $T$  che collega gli estremi del lato  $g$ .

**Dim.: 13.6. (a)** Se l'albero di copertura è di costo minimo, allora deve soddisfare le condizioni di ottimalità.

Se  $e$  è un lato del path in  $T$  che collega gli estremi del lato  $g \notin T$  e  $c_e > c_g$ , potremo sostituire il lato  $e$  con  $g$  ottenendo un albero di copertura con un costo più basso contraddicendo l'ipotesi iniziale.

**(b)** Se l'albero di copertura rispetta le condizioni di ottimalità, allora è un albero di copertura di costo minimo.

*Dimostriamo che un albero  $T$  che soddisfa le path optimality conditions soddisfa anche le cut optimality conditions, così, per il teorema precedente,  $T$  è l'albero di copertura di costo minimo.*

*Sia  $e$  un qualsiasi lato appartenente all'albero  $T$ , e sia  $S$  e  $V \setminus S$  il taglio generato dalla sua eliminazione. Per ogni lato  $g$  del taglio, nell'albero  $T$  ci sarà un unico path  $P$  che collega gli estremi di  $g$  (vedi Figura 13.33). Siccome  $e$  era l'unico lato del taglio che apparteneva a  $T$  (per costruzione), deve appartenere al path  $P$ .*

*Per ipotesi le condizioni di ottimalità sono valide, quindi  $c_g \geq c_e$  per ogni lato  $e$  contenuto nel path in  $T$  che collega gli estremi del lato  $g$ . Siccome questa condizione deve essere valida per tutti i lati non appartenenti a  $T$  del taglio ottenuto eliminando un qualsiasi lato di  $T$ , allora deve soddisfare le cut optimality conditions. Quindi l'albero  $T$  è quello di costo minimo.*

La dimostrazione del Teorema 13.6 dimostra anche l'equivalenza tra le *cut optimality conditions* e le *path optimality conditions*.

### 13.3.4 Algoritmo di Prim-Dijkstra

Le *cut optimality conditions* suggeriscono un algoritmo iterativo per la determinazione dell'albero di copertura di costo minimo di un grafo non orientato  $G(V, E)$ .

L'algoritmo descritto in questa sezione è stato proposto per la prima volta dal matematico ceco Vojtěch Jarník nel 1930. Poi è stato riscoperto indipendentemente da Robert Prim nel 1957 e da Edsger Dijkstra nel 1959. Per queste ragioni è spesso chiamato Algoritmo di Jarník, di Prim, di Prim–Jarník, Prim–Dijkstra oppure DJP.

---

#### Algorithm 10 Prim-Dijkstra (1<sup>a</sup> versione)

---

**Input:** Grafo non orientato  $G(V, E)$  pesato con costi  $c_e$ ,  $\forall e \in E$ ;

**Output:** Albero di copertura di costo minimo  $T^*$ ;

```

// Inizializzazione
 $T^* = \emptyset$ ;
 $S = \{1\}$ ;
// Costruisci l'albero di copertura
while  $|T^*| \neq n - 1$  do
    individua il lato  $\{i, j\} \in \delta(S)$  di costo minimo, con  $i \in S$  e  $j \notin S$ ;
     $T^* = T^* \cup \{\{i, j\}\}$ ;
     $S = S \cup \{j\}$ ;
end while

```

---

L'Algoritmo 10 definisce un insieme iniziale  $S$  contenente un solo vertice del grafo (e.g.,  $S = \{1\}$ ), mentre l'albero iniziale  $T$  è vuoto (i.e.,  $T = \emptyset$ ). Ad ogni iterazione dell'algoritmo si espande la componente连通的  $S$  determinando il lato  $\{i, j\}$ , con  $i \in S$  e  $j \in V \setminus S$ , di costo minimo. L'algoritmo termina quando l'albero  $T$  ha  $n - 1$  archi e, quindi, copre tutti i vertici  $V$ . Avendo costruito

l'albero nel rispetto delle cut optimality conditions,  $T$  è l'albero di copertura di costo minimo.

L'algoritmo richiede  $O(m)$  confronti ad ogni iterazione, quindi la sua complessità è pari perciò a  $O(nm)$ .

In Figura 13.34 è riportato un esempio di esecuzione della prima versione proposta dell'algoritmo di Prim-Dijkstra. L'algoritmo in Figura 13.34a inizializza l'albero  $T^* = \emptyset$  e l'insieme  $S = \{1\}$ . Alla prima iterazione in Figura 13.34b seleziona il lato del taglio  $\delta(S) = \{\{1, 2\}, \{1, 7\}, \{1, 8\}\}$  di costo minimo che risulta essere  $\{1, 8\}$ , per cui aggiorna  $T^* = \{\{1, 8\}\}$  e  $S = \{1, 8\}$ . Alla seconda iterazione in Figura 13.34c il taglio è  $\delta(S) = \{\{1, 2\}, \{1, 7\}, \{8, 7\}, \{8, 6\}, \{8, 5\}\}$  e in questo caso il lato di costo minimo è  $\{8, 7\}$ , per cui aggiorna  $T^* = \{\{1, 8\}, \{8, 7\}\}$  e  $S = \{1, 8, 7\}$ . Alla terza iterazione in Figura 13.34d il taglio è  $\delta(S) = \{\{1, 2\}, \{2, 7\}, \{7, 4\}, \{8, 6\}, \{8, 5\}\}$  e il lato di costo minimo è  $\{8, 6\}$ , per cui aggiorna  $T^* = \{\{1, 8\}, \{8, 7\}, \{8, 6\}\}$  e  $S = \{1, 8, 7, 6\}$ . L'algoritmo procede eseguendo le rimanenti iterazioni mostrate nelle Figure 13.34e-13.34g fino a giungere alla soluzione ottima riportata in Figura 13.34h.

Il grafo dell'esempio in Figura 13.34 ha due diversi alberi di copertura di costo minimo. L'algoritmo di Prim-Dijkstra ne trova solo uno dei due e la soluzione finale dipenderà dall'ordine con cui verranno scelti i lati di pari costo.

Allo scopo di migliorare l'algoritmo, se si considera l'esecuzione dell'iterazione 4 in Figura 13.34e, per evitare di esaminare tutti i lati  $\{i, j\} \in E$  tali che  $i \in S$  e  $j \in V \setminus S$  (i.e.,  $(i, j) \in \delta(S)$ ) si potrebbe supporre di conoscere per ogni vertice  $j \in V \setminus S$  le seguenti *etichette*:

- $pred[j]$ : indica il vertice di  $S$  più vicino a  $j$ , ossia il vertice collegato con il lato  $\{pred[j], j\}$  del taglio  $\delta(S)$  con il costo minimo, i.e.,  $c_{\{pred[j], j\}} = \min_{i \in S} \{c_{\{i, j\}}\}$ ;
- $L[j]$ : corrisponde al costo del lato  $\{pred[j], j\}$ , i.e.,  $L[j] = c_{\{pred[j], j\}}$ .

In questo caso per l'iterazione 4 si avrà quanto riportato in Figura 13.35. Il lato di costo minimo  $\{i^*, j^*\} \in E$ , con  $i^* \in S$  e  $j^* \in V \setminus S$ , può essere calcolato determinando  $j^* \in V \setminus S$  tale che:

$$L[j^*] = \min_{j \in V \setminus S} \{L[j]\} \quad (13.12)$$

e ponendo quindi  $i^* = pred[j^*]$ . Questo metodo per calcolare, all'iterazione 4, il lato  $(i^*, j^*)$  richiede  $|V - S| \leq n$  confronti, quindi la complessità è  $O(n)$ . Le etichette  $[pred[j], L[j]]$ ,  $\forall j \in V \setminus S$ , devono essere opportunamente aggiornate ad ogni iterazione dell'algoritmo; ciò può essere fatto con complessità  $O(n)$ . La seconda versione dell'algoritmo che ne risulta ha perciò una complessità pari a  $O(n^2)$ .

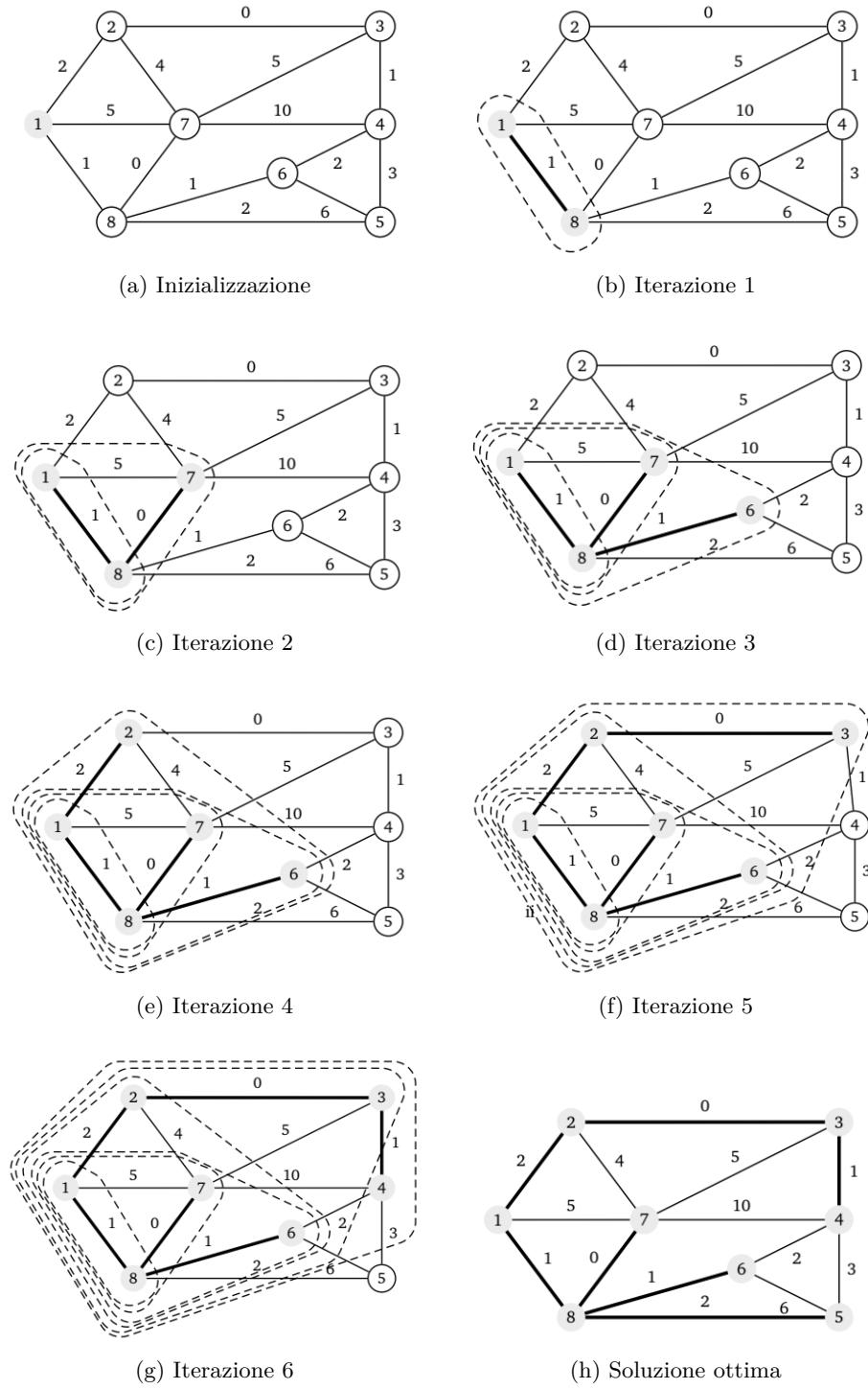


Figura 13.34: Un esempio di esecuzione dell’Algoritmo di Prim-Dijkstra

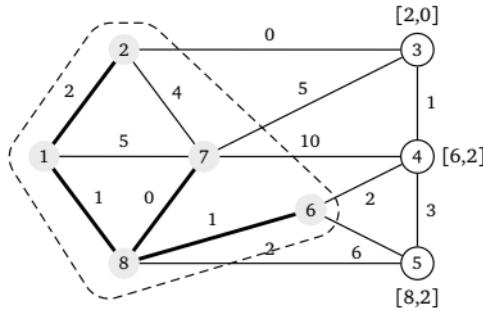


Figura 13.35: Miglioramento dell'algoritmo di Prim usando delle etichette

L'algoritmo 11 per calcolare l'albero di costo minimo  $T$  utilizza le seguenti strutture dati:

- $flag[i] = \begin{cases} 1, & \text{se } i \in S \\ 0, & \text{se } i \in V \setminus S \end{cases}$
- $L[j] = \min\{c_{ij} : i \in S\}, \quad \text{per ogni } j \notin S;$
- $pred[j] = \begin{cases} \operatorname{argmin}\{c_{ij} : i \in S\}, & \text{se } j \notin S \\ \text{predecessore di } j \text{ in } T, & \text{se } j \in S \end{cases}$

In Figura 13.36 è mostrata l'esecuzione della seconda versione dell'algoritmo di Prim-Dijkstra sullo stesso esempio utilizzato in Figura 13.34. L'algoritmo in Figura 13.36a inizializza l'albero  $T^* = \emptyset$ , l'insieme  $S = \{1\}$  e le etichette  $[pred[j], L[j]]$  per ogni vertice  $j \in V$ .

Alla prima iterazione in Figura 13.36b l'algoritmo identifica il vertice  $j \notin S$  con l'etichetta  $L[j]$  minima, che in questo caso è il vertice  $j = 8$ . Quindi, il lato da inserire nell'albero di copertura di costo minimo è  $\{pred[8], 8\}$ , ossia il lato  $\{1, 8\}$ , per cui  $T^* = \{\{1, 8\}\}$  e  $S = \{1, 8\}$ . Alla seconda iterazione in Figura 13.36c il vertice con l'etichetta  $L[j]$  minima è  $j = 7$ . Il lato  $\{pred[7], 7\} = \{8, 7\}$  è aggiunto all'albero di copertura, i.e.,  $T^* = \{\{1, 8\}, \{8, 7\}\}$  e  $S = \{1, 8, 7\}$ . Alla terza iterazione in Figura 13.36d il vertice con l'etichetta  $L[j]$  minima è  $j = 6$ . Il lato  $\{pred[6], 6\} = \{8, 6\}$  è aggiunto all'albero di copertura, i.e.,  $T^* = \{\{1, 8\}, \{8, 7\}, \{8, 6\}\}$  e  $S = \{1, 8, 7, 6\}$ . L'algoritmo procede eseguendo le rimanenti iterazioni mostrate nelle Figure 13.36e-13.36g fino a giungere alla soluzione ottima riportata in Figura 13.36h.

### 13.3.5 Algoritmo di Kruskal

Le *path optimality conditions* suggeriscono un nuovo algoritmo iterativo per la determinazione dell'albero di copertura  $T$  di costo minimo di un grafo non orientato  $G(V, E)$ .

---

**Algorithm 11 Prim-Dijkstra (2<sup>a</sup> versione)**

---

**Input:** Grafo non orientato  $G(V, E)$  pesato con costi  $c_e$ ,  $\forall e \in E$ ;

**Output:** Albero di copertura di costo minimo  $T^*$  definito da  $pred[j]$ ,  $j \in V$ ;

```

// Inizializza le strutture dati partendo da  $S = \{1\}$ 
flag[1] = 1;
pred[1] = 1;
for  $j = 2$  to  $n$  do
    flag[j] = 0;
     $L[j] = c_{1j}$ ;
    pred[j] = 1;
end for
// Seleziona gli  $n - 1$  lati dell'albero
for  $k = 1$  to  $n - 1$  do
    // Sceglie il lato minimo in  $\delta(S)$ 
    min =  $+\infty$ ;
    for  $j = 2$  to  $n$  do
        if ( $flag[j] = 0$ ) and ( $L[j] < min$ ) then
            min =  $L[j]$ ;  $h = j$ ;
        end if
    end for
    // Include il vertice  $h$  in  $S$  e aggiorna  $L[j]$  e  $pred[j]$  per ogni  $j \notin S$ 
    flag[h] = 1;
    for  $j = 2$  to  $n$  do
        if ( $flag[j] = 0$ ) and ( $c_{hj} < L[j]$ ) then
             $L[j] = c_{hj}$ ; pred[j] = h;
        end if
    end for
end for

```

---

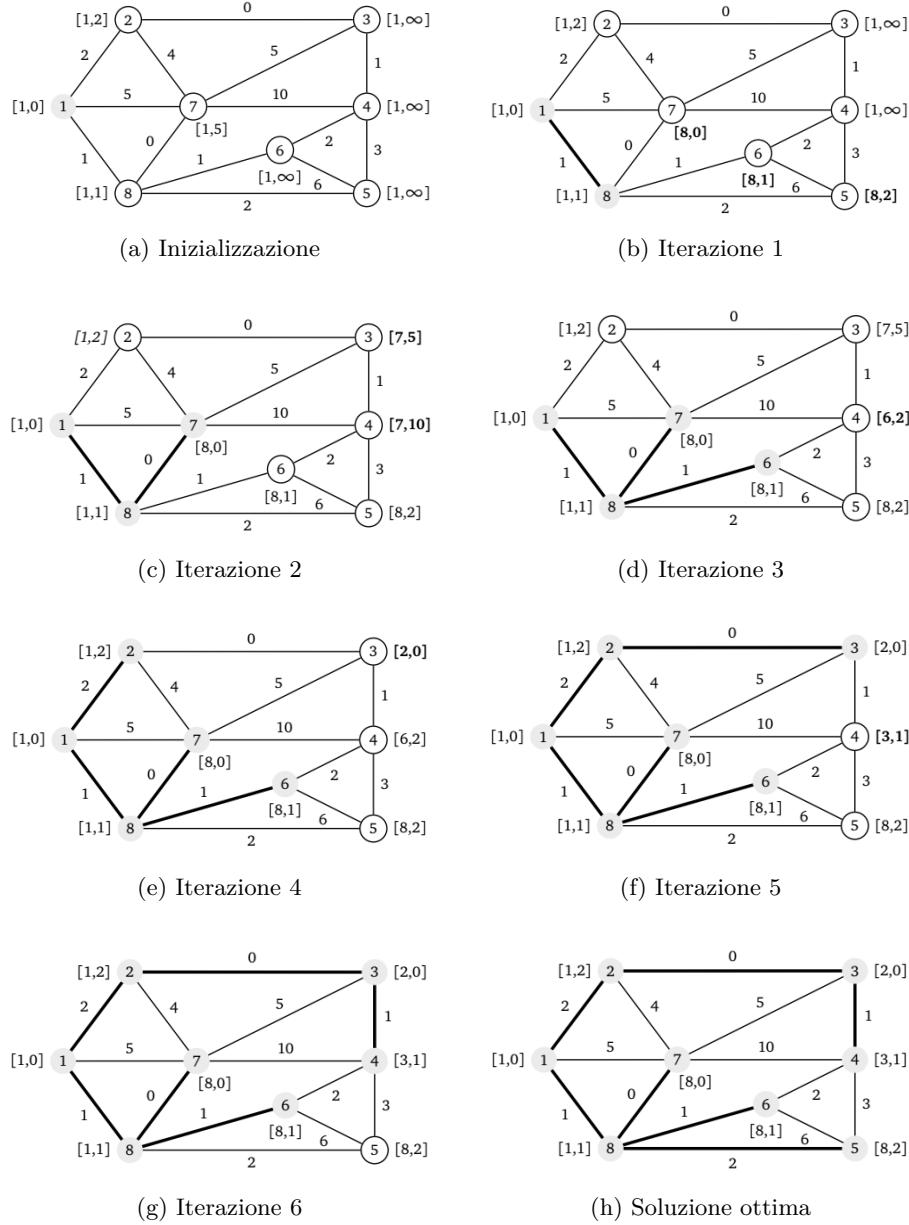


Figura 13.36: Un esempio di esecuzione della seconda versione dell’Algoritmo di Prim-Dijkstra

Il principio di funzionamento dell’algoritmo di Kruskal, descritto nell’Algoritmo 12, è piuttosto semplice. I lati sono inseriti nell’albero  $T$  per costi crescenti, quindi devono essere ordinati. Ciascun lato, considerato in ordine di

costo crescente, è inserito in  $T$  se non forma un ciclo con i lati già in  $T$ , altrimenti viene scartato. Se un lato  $e = \{i, j\}$  non viene inserito perché forma un ciclo, sicuramente non costerà di meno di quelli già inseriti nel path da  $i$  a  $j$ , quindi rispetta le *path optimality conditions*.

---

**Algorithm 12 Kruskal (1<sup>a</sup> versione)**


---

**Input:** Grafo non orientato  $G(V, E)$  pesato con costi  $c_e$ ,  $\forall e \in E$ ;  
**Output:** Albero di copertura di costo minimo  $T^*$ ;

```

// Inizializza strutture dati e ordina i lati
 $T^* = \emptyset$ ;
Ordina i lati per costi crescenti ( $E = \{e_1, \dots, e_m\}$ :  $c(e_i) \leq c(e_{i+1})$ );
// Costruisci l'albero di copertura di costo minimo
 $h = 1$ ;
while ( $|T^*| < n - 1$ ) and ( $h < m$ ) do
    Se il lato  $e_h$  non chiude un ciclo in  $T^*$ , allora  $T^* = T^* \cup \{e_h\}$ ;
     $h = h + 1$ ;
end while
```

---

Il costo computazionale maggiore è dovuto all'ordinamento degli  $m$  lati, quindi la complessità dell'algoritmo è  $O(m \log m)$ .

L'Algoritmo 12 rappresenta un'implementazione di base dell'algoritmo di Kruskal che può essere migliorata. Per definire una modalità efficiente per decidere se aggiungere a  $T^*$  il lato  $e_k = (\alpha, \beta)$ , si hanno i seguenti casi:

- (a)  $\alpha$  e  $\beta$  fanno parte di due sottoalberi distinti (vedi Figura 13.37a);
- (b)  $\alpha$  e  $\beta$  appartengono al medesimo sottoalbero, quindi l'introduzione di  $e_k$  chiude un ciclo (vedi Figura 13.37b).

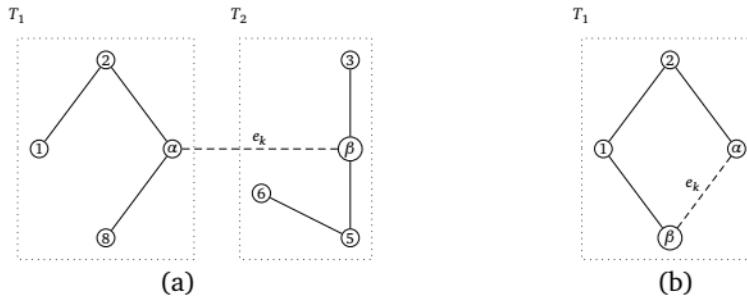


Figura 13.37: Casi da considerare per decidere se aggiungere un lato all'albero di copertura

Per verificare se per il lato  $e_k = (\alpha, \beta)$  si ha il caso (a) o (b) si associa ad ogni vertice  $i \in V$  una etichetta  $comp[i]$  che rappresenta l'indice associato al sottoalbero a cui  $i$  appartiene. Nell'esempio riportato in Figura 13.37a si ha:

- $comp[1] = comp[2] = comp[8] = comp[\alpha] = 1$
- $comp[3] = comp[5] = comp[6] = comp[\beta] = 2$

Mentre, nell'esempio in Figura 13.37b si ha:

- $comp[1] = comp[2] = comp[\alpha] = comp[\beta] = 1$

Perciò, quando si deve stabilire se il lato  $e_k = (\alpha, \beta)$  può essere inserito nell'albero di supporto di costo minimo, sarà sufficiente svolgere il seguente test:

- Se  $comp[\alpha] \neq comp[\beta]$  allora si ha il caso (a), ovvero  $\alpha$  e  $\beta$  fanno parte di due sottoalberi distinti.
- Se  $comp[\alpha] = comp[\beta]$  allora si ha il caso (b), ovvero  $\alpha$  e  $\beta$  fanno parte del medesimo sottoalbero.

Nel caso (a) si provvederà a inserire il lato  $e_k = (\alpha, \beta)$ ) nell'albero  $T_1$  e a unire l'albero  $T_1$  a  $T_2$ . Per unire gli alberi  $T_1$  e  $T_2$  si dovrà aggiornare l'etichetta  $comp[i]$  dei vertici in  $T_2$  con quella dei vertici in  $T_1$ .

L'Algoritmo 13 riassume l'algoritmo di Kruskal che utilizza la struttura dati  $comp[j]$ .

In Figura 13.38 è riportato un esempio dell'esecuzione dell'algoritmo di Kruskal. L'algoritmo inizializza  $comp[j] = j$ , in quanto ogni vertice rappresenta una singola componente连通 (vedi Figura 13.38a). Una volta ordinati i lati per costo crescente, l'algoritmo prova a inserire a turno i lati. Nella prima iterazione, riportata in Figura 13.38b, l'algoritmo seleziona il lato  $\{2, 3\}$  che ha il costo minore. Siccome i vertici 2 e 3 hanno  $comp[2] \neq comp[3]$ , l'algoritmo lo inserisce nell'albero di copertura di costo minimo e aggiorna le etichette  $comp[3] = 2$ . Dopodiché, l'algoritmo ripete l'operazione per i successivi lati. Si noti che all'iterazione 5, quando si inserisce il lato  $\{6, 8\}$  è necessario aggiornare tutte le etichette del sottoalbero contenente i lati  $\{\{1, 2\}, \{1, 8\}, \{6, 8\}, \{7, 8\}\}$ . In Figura 13.38h è riportata la soluzione ottima.

## 13.4 NEW: (Flusso massimo)

Dato un grafo direzionato  $G = (N, A)$ , in cui ad ogni arco  $(i, j) \in A$  è associata una capacità  $u_{ij} \geq 0$ , si vuole determinare il *flusso massimo* che può essere inviato dal vertice *origine*  $s \in N$  al vertice *destinazione*  $t \in N$ .

In Figura 13.39 è proposto un esempio di un'istanza del problema del flusso massimo in cui per ogni arco  $(i, j) \in A$  è riportata la capacità  $u_{ij}$ , mentre l'origine è il vertice  $s = 1$  e la destinazione è  $t = 9$ .

---

**Algorithm 13 Kruskal (2<sup>a</sup> versione)**

---

**Input:** Grafo non orientato  $G(V, E)$  pesato con costi  $c_e$ ,  $\forall e \in E$ ;  
**Output:** Albero di copertura di costo minimo  $T^*$ ;

```

// Inizializza le strutture dati e ordina i lati
Ordina i lati per costi crescenti ( $E = \{e_1, \dots, e_m\} : c(e_i) \leq c(e_{i+1})$ );
k = 0; h = 0;
// Ogni vertice rappresenta una componente distinta
for i = 1 to n do
    comp[i] = i;
end for
// Costruisci l'albero di copertura di costo minimo
while (k < n - 1) and (h < m) do
    Seleziona il lato  $e_h = \{i, j\}$ ;
    C1 = comp[i]; C2 = comp[j];
    if (C1 ≠ C2) then
        Inserisci  $e_h$  nell'albero  $T^*$ ; k=k+1;
        // Unisci le componenti C1 e C2
        for q = 1 to n do
            if (comp[q] = C2) then
                comp[q] = C1;
            end if
        end for
    end if
    h = h + 1;
end while
if (k ≠ n - 1) then
    Il grafo  $G$  è disconnesso;
end if

```

---

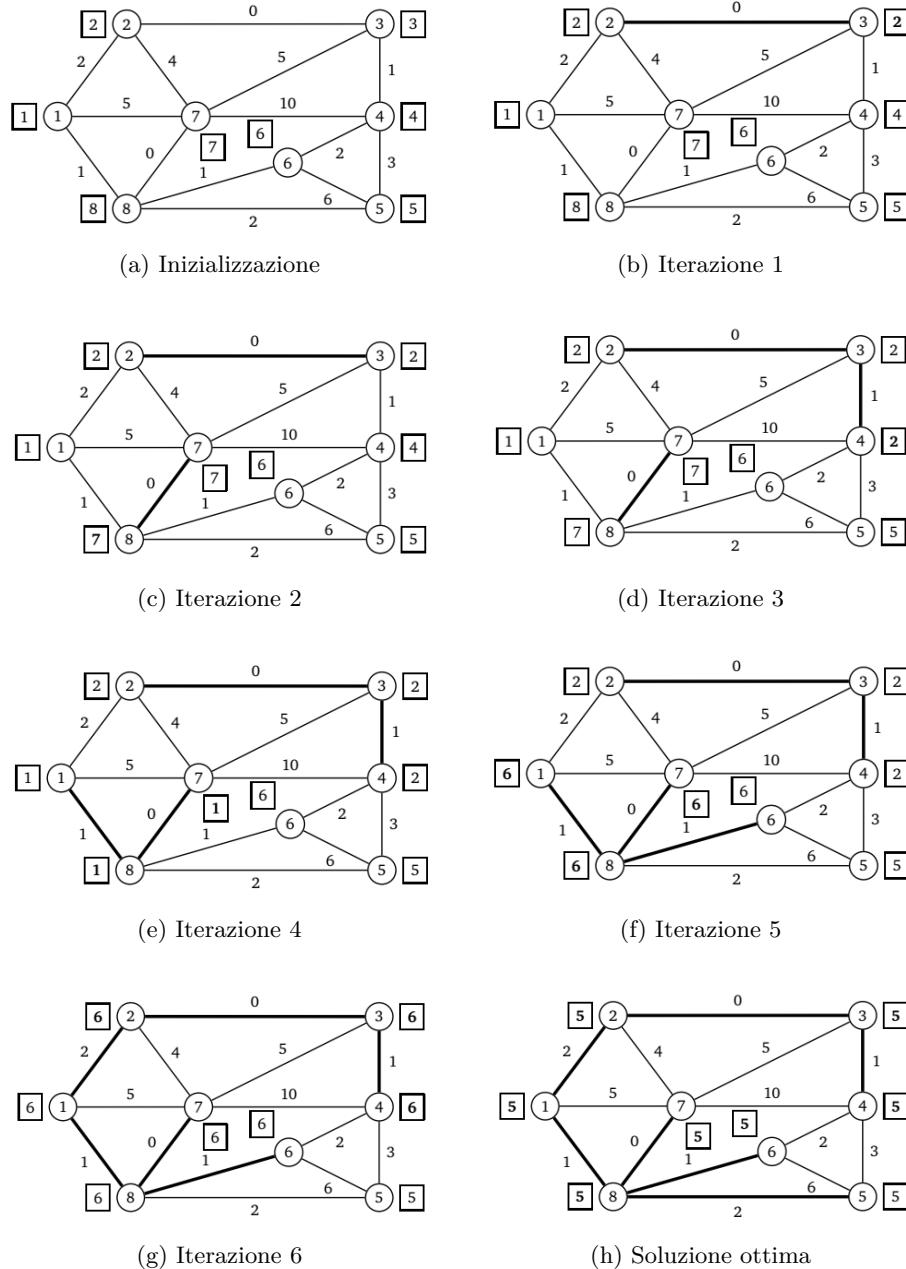


Figura 13.38: Un esempio di esecuzione della seconda versione dell’Algoritmo di Kruskal

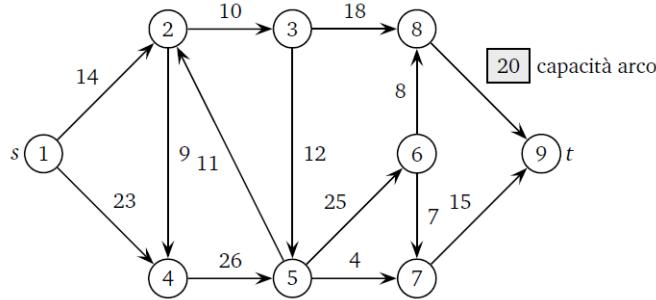


Figura 13.39: Esempio di istanza per il Problema del Flusso Massimo

### 13.4.1 Formulazione matematica

Per formulare il problema del flusso massimo sono necessarie le seguenti variabili decisionali:

- $x_{ij}$ : quantità di flusso assegnata a ciascun arco  $(i, j) \in A$ ;
- $v$ : flusso introdotto all'origine  $s$  e ricevuto alla destinazione  $t$ .

Il problema del flusso massimo può essere formulato come segue:

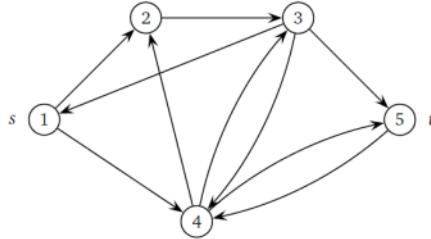
$$(MF) \quad z_{MF} = \max v \quad (13.13)$$

$$s.t. \quad \sum_{j \in \Gamma_i} x_{ij} - \sum_{j \in \Gamma_i^{-1}} x_{ji} = \begin{cases} v, & i = s \\ -v, & i = t \\ 0, & i \neq N \setminus \{s, t\} \end{cases} \quad (13.14)$$

$$0 \leq x_{ij} \leq u_{ij}, \quad (i, j) \in A \quad (13.15)$$

dove  $\Gamma_i = \{j : (i, j) \in A\}$  e  $\Gamma_i^{-1} = \{j : (j, i) \in A\}$ . La funzione obiettivo (13.13) massimizza flusso introdotto all'origine  $s$  e ricevuto alla destinazione  $t$ . I vincoli (13.14) garantiscono la conservazione del flusso ad ogni vertice  $i \in N$ , tenendo conto che al vertice di origine  $s$  deve entrare il flusso  $v$  che poi uscirà al vertice destinazione  $t$ . I vincoli (13.15) assicurano che il flusso assegnato a ciascun arco  $(i, j) \in A$  sia non negativo e non superi la capacità  $u_{ij}$ .

Nella Figura 13.40 è riportato un esempio di un grafo con i corrispondenti vincoli di conservazione del flusso.



Vertice	$x_{12}$	$x_{14}$	$x_{23}$	$x_{31}$	$x_{34}$	$x_{35}$	$x_{42}$	$x_{43}$	$x_{45}$	$x_{54}$	$v$
$s = 1$	$x_{12} + x_{14}$		$-x_{31}$								$= v$
2	$-x_{12}$	$+x_{23}$					$-x_{42}$				$= 0$
3		$-x_{23} + x_{31} + x_{34} + x_{35}$					$-x_{43}$				$= 0$
4		$-x_{14}$		$-x_{34}$		$-x_{35}$	$+x_{42} + x_{43} + x_{45} - x_{54}$				$= 0$
$t = 5$							$-x_{45}$	$-x_{45} + x_{54}$			$= -v$

Figura 13.40: Esempio di vincoli di conservazione del flusso

### 13.4.2 Assunzioni e definizioni

Introduciamo delle assunzioni e definizioni utili nel proseguo di questa sezione quando saranno descritte le condizioni di ottimalità e gli algoritmi di soluzione.

Si assume che il grafo  $G(N, A)$  sia orientato e che abbia  $n$  vertici e con  $m$  archi. Inoltre, si assume che tutte le capacità  $u_{ij}$  sono intere non negative (i.e.,  $u_{ij} \geq 0$ ) e che il grafo non contiene un cammino orientato dal vertice  $s$  al vertice  $t$  composto da soli archi di capacità infinita, per cui la soluzione ottima è limitata. Si denota con  $U$  la capacità massima degli archi, i.e.,  $U = \max\{u_{ij} : (i, j) \in A\}$ .

Per semplicità, ma senza perdere di generalità, si ipotizza che se in  $A$  è contenuto l'arco  $(i, j)$  allora è contenuto anche l'arco  $(j, i)$ . Si noti che questa assunzione non limita la possibilità di utilizzo dei risultati che seguiranno, in quanto uno dei due archi potrebbe avere capacità nulla.

Il *grafo residuo*  $G(x)$  corrispondente al flusso  $x$  nel grafo  $G$  è ottenuto sostituendo ogni arco  $(i, j) \in A$  con un arco “diretto”  $(i, j)$  e un arco “inverso”  $(j, i)$  con una *capacità residua* pari a  $r_{ij} = u_{ij} - x_{ij}$  e  $r_{ji} = x_{ij}$ .

Dato un flusso  $x$  nel grafo  $G$ , un *cammino aumentante* è un cammino da  $s$  a  $t$  nel grafo residuo  $G(x)$ . In alternativa, e in modo equivalente, si può definire il *cammino aumentante* come il cammino non orientato da  $s$  a  $t$  nel grafo  $G$ , in cui si può aumentare il flusso di almeno una unità negli archi percorsi nella direzione originaria e diminuire il flusso di almeno una unità negli archi percorsi nella direzione contraria rispetto a quella originaria.

Si definisce *Taglio  $s-t$*  una partizione dei vertici  $N$  del grafo  $G$  in due sottoinsiemi  $S$  e  $\bar{S} = N \setminus S$  tali che  $s \in S$  e  $t \in \bar{S}$ . La capacità di un taglio  $s-t$  è data da:

$$u(S, \bar{S}) = \sum_{i \in S} \sum_{j \in \bar{S}} u_{ij} = \sum_{(i,j) \in \Gamma(S, \bar{S})} u_{ij}$$

dove  $\Gamma(S, \bar{S}) = \{(i, j) \in A : i \in S, j \in \bar{S}\}$ . Si noti che il flusso che attraversa gli archi del taglio appartenenti all'insieme  $\Gamma(\bar{S}, S) = \{(i, j) \in A : i \in \bar{S}, j \in S\}$  riduce il flusso da  $s$  a  $t$ .

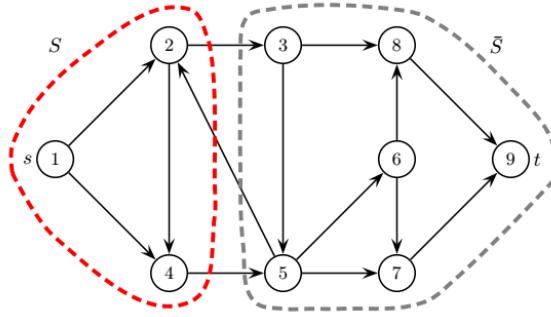


Figura 13.41: Esempio di taglio  $s-t$

In Figura 13.41 è mostrato un esempio di taglio  $s-t$  in cui  $S = \{1, 2, 4\}$  e  $\bar{S} = V \setminus S = \{3, 5, 6, 7, 8, 9\}$ , per cui si ha  $\Gamma(S, \bar{S}) = \{(2, 3), (4, 5)\}$  e  $\Gamma(\bar{S}, S) = \{(5, 2)\}$ . Si noti che il taglio è un taglio  $s-t$  perché  $s \in S$  e  $t \in \bar{S}$ .

### 13.4.3 Condizioni di ottimalità

In questa sezione vengono introdotti dei teoremi utili alla definizione di condizioni di ottimalità che sono alla base degli algoritmi di soluzione descritti in seguito, in particolare l'Algoritmo di Ford Fulkerson descritto in Sezione 13.4.4.

**Teorema 13.7.** *Dato un flusso ammissibile  $x$  pari a  $v$ , per ogni taglio  $s-t$   $(S, \bar{S})$  il flusso che attraversa il taglio è dato da:*

$$v = \sum_{(i,j) \in \Gamma(S, \bar{S})} x_{ij} - \sum_{(i,j) \in \Gamma(\bar{S}, S)} x_{ij}$$

**Dim.: 13.7.** *Sommendo i vincoli di conservazione del flusso per tutti i nodi di  $S$  si ottiene quanto segue:*

$$\begin{aligned} v &= \sum_{h \in S} \left[ \sum_{j \in \Gamma_h} x_{hj} - \sum_{j \in \Gamma_h^{-1}} x_{jh} \right] = \sum_{h \in S} \sum_{j \in \Gamma_h} x_{hj} - \sum_{h \in S} \sum_{j \in \Gamma_h^{-1}} x_{jh} \\ &= \left[ \sum_{(i,j) \in A(S)} x_{ij} + \sum_{(i,j) \in \Gamma(S, \bar{S})} x_{ij} \right] - \left[ \sum_{(i,j) \in A(S)} x_{ij} + \sum_{(i,j) \in \Gamma(\bar{S}, S)} x_{ij} \right] \\ &= \sum_{(i,j) \in \Gamma(S, \bar{S})} x_{ij} - \sum_{(i,j) \in \Gamma(\bar{S}, S)} x_{ij} \end{aligned}$$

dove  $A(S) = \{(i, j) \in A : i \in S, j \in S\}$ .

**Teorema 13.8.** Il valore  $v$  di ogni flusso ammissibile  $x$  in  $G$  è minore o uguale alla capacità  $u(S, \bar{S})$  di un qualunque taglio  $s-t$ .

**Dim.: 13.8.** Per il Teorema precedente il valore  $v$  del flusso  $x$  è dato da:

$$v = \sum_{(i,j) \in \Gamma(S, \bar{S})} x_{ij} - \sum_{(i,j) \in \Gamma(\bar{S}, S)} x_{ij}$$

Un limite superiore al valore di  $v$  si ottiene sostituendo  $x_{ij}$  con  $u_{ij}$ , per ogni  $(i, j) \in \Gamma(S, \bar{S})$ , e  $x_{ij}$  con 0, per ogni  $(i, j) \in \Gamma(\bar{S}, S)$ . Per cui:

$$v \leq \sum_{(i,j) \in \Gamma(S, \bar{S})} u_{ij} = u(S, \bar{S})$$

In particolare, anche il valore del flusso massimo è minore o uguale alla capacità del taglio  $s-t$  di capacità minima.

**Teorema 13.9** (Augmentig Path Theorem). Sia  $x$  un flusso ammissibile in  $G$  da  $s$  a  $t$  e sia  $G(x)$  il corrispondente grafo residuo. Il flusso  $x$  è massimo se e solo se non esiste in  $G(x)$  un cammino da  $s$  a  $t$  (i.e., non esiste un cammino aumentante da  $s$  a  $t$  in  $G$ ).

**Dim.: 13.9.** (a) Se il flusso  $x$  è massimo, allora non esiste un cammino  $P$  in  $G(x)$ .

Se per assurdo esistesse un cammino aumentante  $P$  in  $G(x)$  tale che  $\delta = \min\{r_{ij} : \forall (i, j) \in P\} > 0$ , allora sarebbe possibile aggiornare il flusso per ogni arco  $(i, j) \in P$ :

- $x_{ij} = x_{ij} + \delta$ , se  $(i, j)$  è un arco diretto;
- $x_{ji} = x_{ji} - \delta$ , se  $(i, j)$  è un arco inverso;

Il nuovo flusso aggiornato  $x$  è un flusso ammissibile di valore  $v + \delta$ , per cui il flusso  $x$  prima dell'aggiornamento non era ottimo.

(b) Se non esiste un cammino  $P$  in  $G(x)$ , allora il flusso  $x$  è massimo.

Se non esiste un cammino  $P$  in  $G(x)$ , allora esiste un taglio  $(S, \bar{S})$ , dove  $\bar{S} = V \setminus S$ , nel grafo residuo  $G(x)$  tale che  $\Gamma(S, \bar{S}) = \emptyset$ . Nella rete originale si ha allora che:

- ogni arco  $(i, j) \in \Gamma(S, \bar{S})$  è saturo (i.e.,  $x_{ij} = u_{ij}$ );
- ogni arco  $(i, j) \in \Gamma(\bar{S}, S)$  è scarico (i.e.,  $x_{ij} = 0$ ).

Quindi, il valore del flusso per il taglio  $(S, \bar{S})$  è uguale a:

$$v = \sum_{(i,j) \in \Gamma(S, \bar{S})} x_{ij} - \sum_{(i,j) \in \Gamma(\bar{S}, S)} x_{ij} = \sum_{(i,j) \in \Gamma(S, \bar{S})} u_{ij} = u(S, \bar{S})$$

e dal teorema precedente si ha che il flusso  $x$  è ottimo e  $(S, \bar{S})$  è il taglio  $s-t$  di capacità minima.

Un ulteriore importante teorema che ne deriva è il seguente:

**Teorema 13.10** (Max-Flow Min-Cut Theorem). *Il flusso massimo da  $s$  a  $t$  in  $G$  è uguale alla capacità del taglio  $s$ - $t$  di capacità minima.*

Inoltre, una importante proprietà del problema del flusso massimo è definita dal seguente teorema.

**Teorema 13.11** (Integrality Theorem). *Se tutti gli archi hanno una capacità  $u_{ij}$  intera, allora il problema del flusso massimo ha una soluzione intera.*

#### 13.4.4 Algoritmo di Ford-Fulkerson

Un algoritmo per risolvere il problema del flusso massimo è l'Algoritmo di Ford-Fulkerson che è basato sul *Augmenting Path Theorem*.

##### Algoritmo di Ford-Fulkerson (versione 1)

Step 1. [Inizializza]

Poni  $x_{ij} = 0$ , per ogni  $(i, j) \in A$ , e  $v = 0$ .

Step 2. [Determina Cammino Aumentante]

Costruisci il grafo residuo  $G(x)$ .

Trova un cammino da  $s$  a  $t$  in  $G(x)$ .

Se non esiste un cammino da  $s$  a  $t$ , allora il flusso  $x$  è massimo, quindi STOP.

Step 3. [Aumenta il Flusso  $x$ ]

Sia  $P$  il cammino da  $s$  a  $t$  trovato e ponli  $\delta = \min\{r_{ij} : (i, j) \in P\}$ .

Per ogni  $(i, j) \in P$ :

- (i) Se  $(i, j)$  corrisponde all'arco  $(i, j) \in A$ :  $x_{ij} = x_{ij} + \delta$ ;
- (ii) Se  $(i, j)$  corrisponde all'arco  $(j, i) \in A$ :  $x_{ji} = x_{ji} - \delta$ ;

Aggiorna il flusso  $v = v + \delta$ .

Ritorna allo Step 2.

L'algoritmo di Ford-Fulkerson può essere riformulato senza usare il grafo residuo  $G(x)$  esplicitamente.

Per consentire il calcolo del cammino aumentante nel grafo residuo senza doverlo generare si possono usare delle etichette  $[pred[j], \delta_j]$  per ciascun vertice  $j \in N$ .  $pred[j]$  è il “predecessore” del vertice  $j$  nel cammino aumentante:

- Se  $pred[j] = i > 0$ , allora nell'arco  $(i, j) \in A$  il flusso potrebbe essere aumentato di  $\delta_j$ .
- Se  $pred[j] = -i < 0$ , allora nell'arco  $(j, i) \in A$  il flusso potrebbe essere diminuito di  $\delta_j$ .

Ciascun  $\delta_j$  indica il massimo aumento consentito nel cammino da  $s$  fino a  $j$ . Mentre, l'effettivo aumento del flusso nel cammino da  $s$  a  $t$  è determinato da  $\delta_t$ .

### Algoritmo di Ford-Fulkerson (versione 2)

#### Step 1. [Inizializza]

Poni  $x_{ij} = 0$ , per ogni  $(i, j) \in A$ , e  $v = 0$ .

#### Step 2. [Determina Cammino Aumentante]

Dichiara il vertice  $s$  non espanso con etichetta  $[+s, \infty]$ , mentre dichiara tutti gli altri vertici non etichettati (e non espansi).

- (i) Se tutti i nodi etichettati sono già espansi, allora il flusso  $v$  è massimo (i.e., la soluzione  $x$  è ottima), quindi STOP.
- (ii) Se esiste un vertice  $i$  etichettato ma non ancora espanso, provvedi a espanderlo:
  - Per ogni vertice  $j \in \Gamma_i$  non etichettato per cui  $x_{ij} < u_{ij}$  assegna l'etichetta  $[+i, \delta_j]$ , dove  $\delta_j = \min\{u_{ij} - x_{ij}, \delta_i\}$ ;
  - Per ogni vertice  $j \in \Gamma_i^{-1}$  non etichettato per cui  $x_{ji} > 0$  assegna l'etichetta  $[-i, \delta_j]$ , dove  $\delta_j = \min\{x_{ji}, \delta_i\}$ .
- (iii) Se  $t$  non risulta etichettato torna a passo (i).

#### Step 3. [Aumenta il Flusso $x$ ]

Sia  $P$  il cammino aumentante dal vertice  $s$  al vertice  $t$  (ricostruito usando le etichette).

Per ogni  $(i, j) \in P$ :

- (i) Se l'arco  $(i, j)$  è percorso nel suo verso originario:  $x_{ij} = x_{ij} + \delta_t$ ;
- (ii) Se l'arco  $(i, j)$  è percorso nel verso contrario:  $x_{ij} = x_{ij} - \delta_t$ .

Aggiorna il flusso  $v = v + \delta_t$ .

Annnulla tutte etichette e ritorna allo Step 2.

Quando l'Algoritmo di Ford-Fulkerson termina, l'insieme dei vertici etichettati  $S$  e quello dei vertici non etichettati  $\bar{S}$  costituiscono un *taglio di capacità minima*.

L'algoritmo di Ford-Fulkerson ha complessità  $O(nmU)$ , perché ad ogni iterazione il flusso aumenta di  $\delta \geq 1$  e il valore del flusso è limitato superiormente da  $nU$ , mentre il calcolo di un cammino aumentante ha complessità  $O(m)$ .

Nel 1972 Edmonds e Karp hanno proposto alcune varianti dell'algoritmo di Ford-Fulkerson:

- calcolando il cammino aumentante di capacità massima la complessità è pari a  $O(nm \log U)$ ;
- calcolando il cammino aumentante di cardinalità minima la complessità è pari a  $O(n^2m)$ .

Esistono altri algoritmi con “minore” complessità, per esempio:

- FIFO preflow-push con complessità  $O(n^3)$ ;
- Highest-label preflow push con complessità  $O(n^2\sqrt{m})$ ;
- Excess scaling con complessità  $O(nm + n^2 \log U)$ .

In Figura 13.42 è riportato un esempio di esecuzione della prima versione dell’Algoritmo di Ford-Fulkerson. Questa versione usando esplicitamente il grafo residuo inizializzando la soluzione e il flusso iniziale a  $x = 0$  e  $v = 0$ . Nella prima iterazione (Figura 13.42a) il flusso può essere aumentato di  $\delta = \min\{r_{13}, r_{34}\} = 4$  unità sul cammino aumentante  $P = (1, 3, 4)$  e  $v = v + \delta = 0 + 4 = 4$ . Dopo aver aggiornato il grafo residuo  $G(x)$ , nella seconda iterazione (Figura 13.42b) si cerca un nuovo cammino aumentante. Il flusso può essere aumentato di  $\delta = \min\{r_{12}, r_{23}, r_{34}\} = 1$  unità sul cammino aumentante  $P = (1, 2, 3, 4)$  e  $v = v + \delta = 4 + 1 = 5$ . Nella terza iterazione si aggiorna nuovamente il grafo residuo  $G(x)$  (Figura 13.42c) e si scopre che il flusso può essere aumentato di  $\delta = \min\{r_{12}, r_{24}\} = 1$  unità sul cammino aumentante  $P = (1, 2, 4)$  e  $v = c + \delta = 5 + 1 = 6$ . Nell’ultima iterazione (Figura 13.42d) si aggiorna un’ultima volta il grafo residuo  $G(x)$  e si scopre che il grafo residuo non contiene alcun cammino aumentante. Quindi, il flusso massimo è  $v = 6$  e il taglio di capacità minima è definito da  $S = \{1\}$  e  $\bar{S} = \{2, 3, 4\}$ ; inoltre, si può notare che  $v = \sum_{i \in S} \sum_{j \in \bar{S}} u_{ij} = 6$ .

In Figura 13.43 è riportata l’esecuzione della prima iterazione della seconda versione dell’Algoritmo di Ford-Fulkerson che non usa esplicitamente il grafo residuo e che è stato applicato all’esempio in Figura 13.39. La prima iterazione (Figura 13.43a) parte inizializzando la soluzione e il flusso fissando  $x = 0$  e  $v = 0$  e definendo la sola etichetta del vertice  $s = 1$  impostando il valore  $[1, +\infty]$ . Nel secondo passo della prima iterazione (Figura 13.43b) si procede a espandere il vertice 1 definendo le etichette dei vertici 2 e 4. Nel terzo passo (Figura 13.43c) tra i vertici etichettati e non espansi (i.e., 2 e 4), viene espanso il vertice 2 considerando i vertici adiacenti 3 e 4, però quest’ultimo è già stato etichettato. Dopodiché, tra i nodi etichettati e non espansi (i.e., 3 e 4), viene espanso il nodo 3 considerando i nodi adiacenti 5 e 8 (Figura 13.43d). Poi, tra i nodi etichettati e non espansi (i.e., 4, 5 e 8), viene espanso il nodo 4, ma il nodo adiacente 5 è già stato etichettato. Quindi, si espande il nodo 5 che ha i nodi adiacenti 2 (già etichettato), 6 e 7 (Figura 13.43e). Infine, tra i nodi etichettati e non espansi (i.e., 6, 7 e 8), viene espanso il nodo 6, ma i nodi adiacenti 7 e 8 sono già etichettati. Quindi, si espande il nodo 7 che ha il nodo adiacente  $t = 9$  (Figura 13.43f). Siccome il nodo  $t = 9$  è stato etichettato con  $\delta_t = 4$ , possiamo aumentare il flusso di 4 unità lungo il cammino aumentante. Il cammino aumentante  $P = (1, 2, 3, 5, 7, 9)$  è stato ricostruito utilizzando le etichette a partire dal nodo  $t = 9$ . Il flusso viene aggiornato:  $v = v + \delta_t = 0 + 4 = 4$  e il flusso degli archi (i.e., la soluzione  $x$ ) si modifica come segue:

- arco (7,9):  $x_{79} = x_{79} + \delta_t = 0 + 4 = 4$ ;

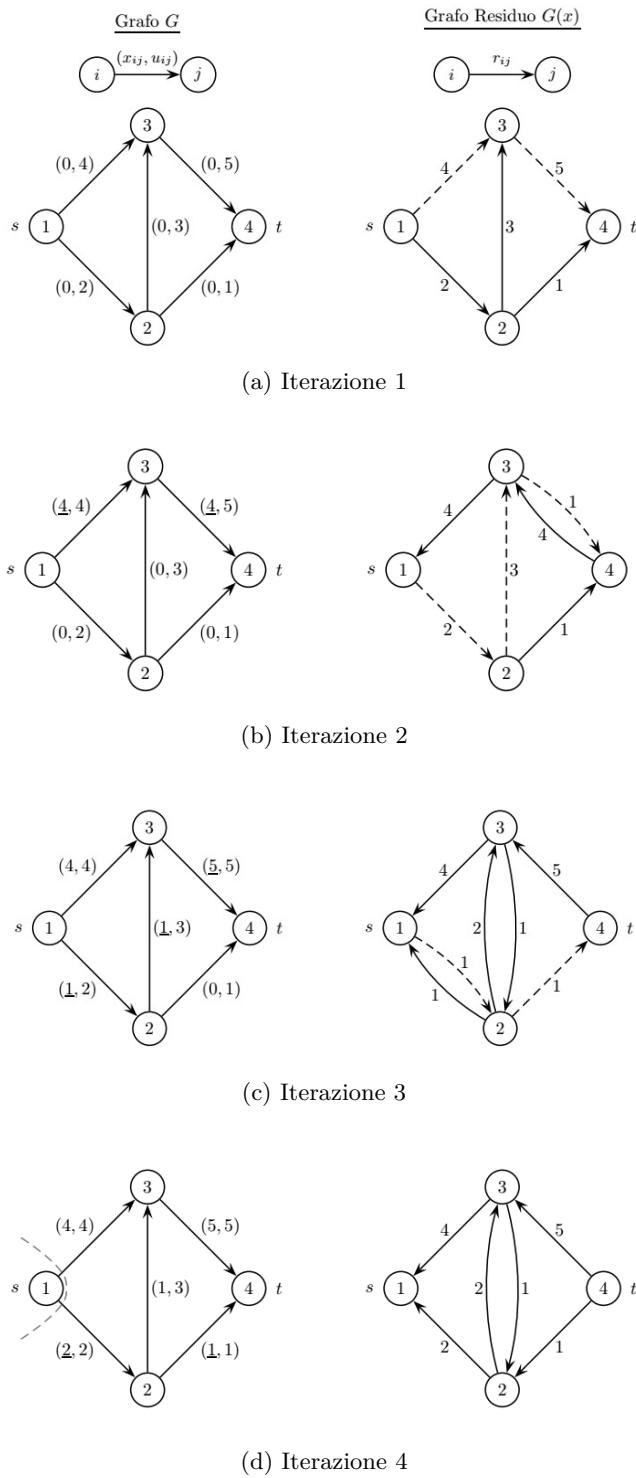


Figura 13.42: Un esempio di esecuzione della prima versione dell'Algoritmo di Ford-Fulkerson

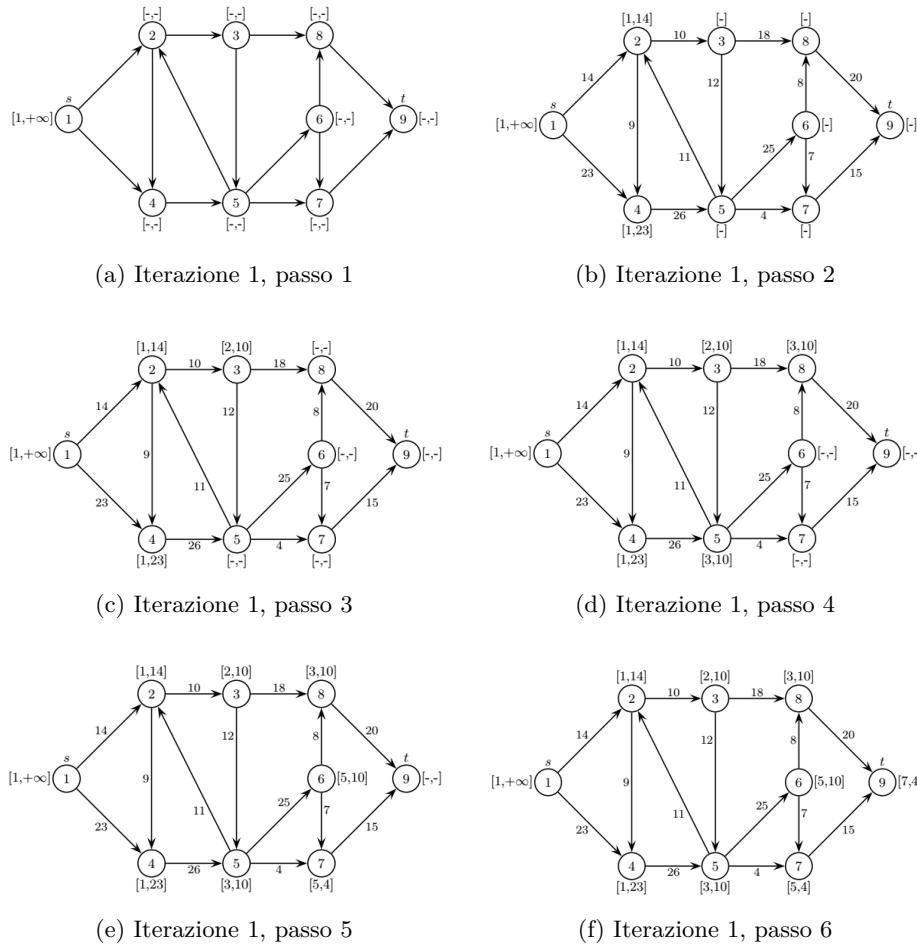


Figura 13.43: Un esempio di esecuzione della prima iterazione della seconda versione dell’Algoritmo di Ford-Fulkerson

- arco (5,7):  $x_{57} = x_{57} + \delta_t = 0 + 4 = 4$ ;
- arco (3,5):  $x_{35} = x_{35} + \delta_t = 0 + 4 = 4$ ;
- arco (2,3):  $x_{23} = x_{23} + \delta_t = 0 + 4 = 4$ ;
- arco (1,2):  $x_{12} = x_{12} + \delta_t = 0 + 4 = 4$ .

Nella Figura 13.44 sono riportate tutte le iterazioni della seconda versione dell’Algoritmo di Ford-Fulkerson. Nella seconda iterazione (Figura 13.43b), al termine dell’etichettamento, si può aumentare il flusso di  $\delta_t = 6$  unità lungo il cammino aumentante  $P = (1, 2, 3, 5, 6, 7, 9)$  e il nuovo flusso è  $v = v + \delta_t = 4 + 6 = 10$ . Nella terza iterazione (Figura 13.43c), si può aumentare il flusso di  $\delta_t = 1$  unità lungo il cammino aumentante  $P = (1, 4, 5, 6, 7, 9)$  e il nuovo flusso è  $v = v + \delta_t = 10 + 1 = 11$ . Nella quarta iterazione (Figura 13.43d), il flusso è aumentato di  $\delta_t = 10$  unità lungo il cammino aumentante  $P = (1, 4, 5, 3, 8, 9)$  e il nuovo flusso è  $v = v + \delta_t = 11 + 10 = 21$ . Poi, nella quinta iterazione (Figura 13.43e), il flusso è aumentato di  $\delta_t = 8$  unità lungo il cammino aumentante  $P = (1, 4, 5, 6, 8, 9)$  e il nuovo flusso è  $v = v + \delta_t = 21 + 8 = 29$ . Infine, nell’ultima iterazione (Figura 13.43e), si scopre che non esiste un cammino aumentante che raggiunge il nodo  $t = 9$ , per cui la soluzione corrente è ottima e il flusso massimo è pari a 29. Il taglio di capacità minima è determinato dall’insieme  $S = \{1, 2, 4, 5, 6\}$  dei nodi etichettati e l’insieme  $\bar{S} = \{3, 7, 8, 9\}$  dei nodi non etichettati.

### 13.5 NEW: (Flusso di costo minimo)

Sia dato un grafo orientato  $G = (N, A)$ , in cui ad ogni arco  $(i, j) \in A$  è associato un costo  $c_{ij}$  e una capacità  $u_{ij}$ . Ad ogni vertice  $i \in N$  è associata una disponibilità  $b_i > 0$  oppure una richiesta  $b_i < 0$  oppure  $b_i = 0$ .

Il problema del flusso di costo minimo può essere formulato come segue:

$$(MCF) \quad z_{MCF} = \min \sum_{(i,j) \in A} c_{ij} x_{ij} \quad (13.16)$$

$$\text{s.t. } \sum_{j \in \Gamma_i} x_{ij} - \sum_{j \in \Gamma_i^{-1}} x_{ji} = b_i, \quad i \in N \quad (13.17)$$

$$0 \leq x_{ij} \leq u_{ij}, \quad (i, j) \in A \quad (13.18)$$

dove  $\Gamma_i = \{j : (i, j) \in A\}$  e  $\Gamma_i^{-1} = \{j : (j, i) \in A\}$ .

Al vincolo di conservazione del flusso corrispondente al nodo  $i \in N$  è associata una variabile duale  $\pi_i$  e al vincolo di capacità relativo all’arco  $(i, j) \in A$  è associata la variabile duale  $\alpha_{ij}$ .

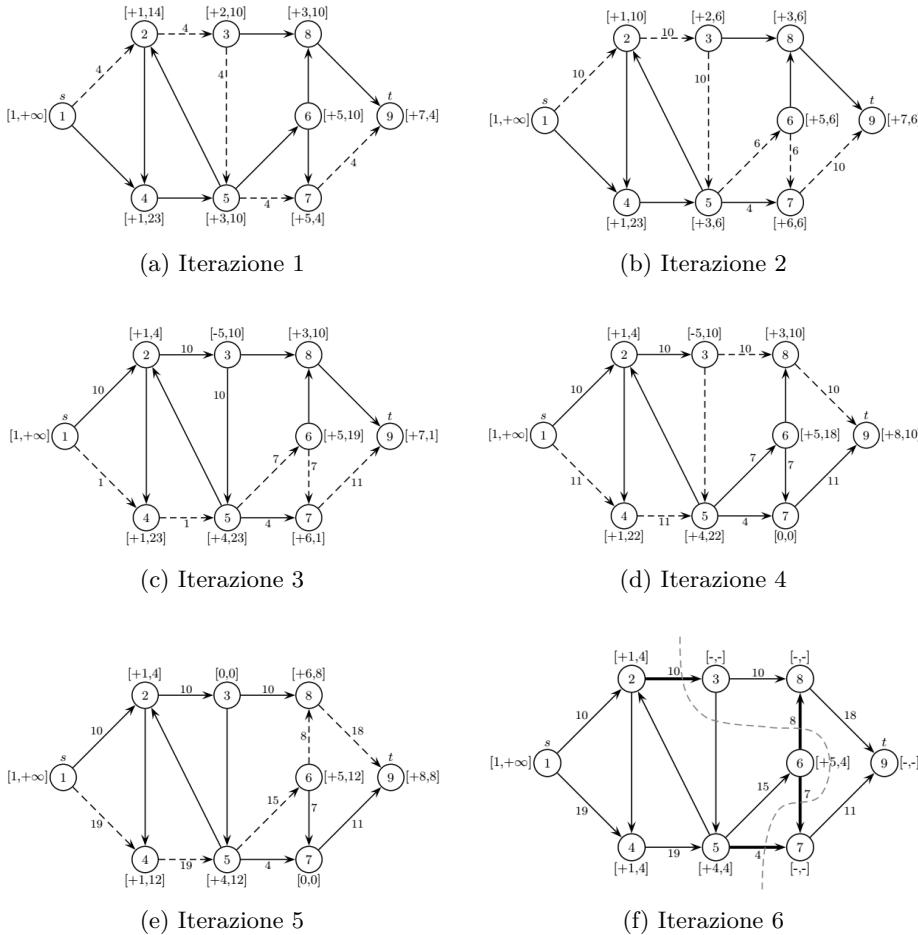


Figura 13.44: Un esempio di esecuzione di tutte le iterazioni della seconda versione dell’Algoritmo di Ford-Fulkerson

Il duale della formulazione *MCF* del problema del flusso di costo minimo è il seguente:

$$(D) \quad z_D = \max \sum_{i \in N} b_i \pi_i - \sum_{(i,j) \in A} u_{ij} \alpha_{ij} \quad (13.19)$$

$$\text{s.t. } \pi_i - \pi_j - \alpha_{ij} \leq c_{ij}, \quad (i, j) \in A \quad (13.20)$$

$$\pi_i \text{ qualsiasi,} \quad i \in N \quad (13.21)$$

$$\alpha_{ij} \geq 0. \quad (i, j) \in A \quad (13.22)$$

### 13.5.1 Assunzioni e definizioni

In questa sezione si riassumono tutte le assunzioni che permettono di semplificare la presentazione e, in alcuni casi, garantire la convergenza alla soluzione ottima (e quindi ammissibile) del problema del flusso di costo minimo.

Si assume che il grafo  $G(N, A)$  sia orientato con  $n$  vertici ed  $m$  archi. Tutti i parametri del problema (i.e., costi, richieste, disponibilità e capacità) sono valori interi. Inoltre, tutti i costi  $c_{ij}$  e le capacità  $u_{ij}$  sono valori non negativi. Definiamo  $U = \max\{u_{ij} : (i, j) \in A\}$  e  $C = \max\{c_{ij} : (i, j) \in A\}$ .

Le richieste e le disponibilità dei diversi vertici soddisfano la condizione  $\sum_{i \in N} b_i = 0$  e che il problema di flusso di costo minimo ha una soluzione ammissibile. Si assume anche che il grafo contiene un cammino diretto di capacità infinita per ogni coppia di vertici. Con l'uso di eventuali variabili artificiali si può garantire l'esistenza di una soluzione "ammissibile" (che potrebbe non esserlo per il problema originario).

Per il problema del flusso di costo minimo, il grafo residuo  $G(x)$  corrispondente al flusso  $x$  è ottenuto sostituendo ogni arco  $(i, j) \in A$  con due archi  $(i, j)$  e  $(j, i)$  tali che:

- il loro costo è pari a  $c_{ij} = c_{ij}$  e  $c_{ji} = -c_{ij}$ .
- la loro capacità residua è pari a  $r_{ij} = u_{ij} - x_{ij}$  e  $r_{ji} = x_{ij}$ .

Si denota con  $\pi_i \in \mathbb{R}$  il *potenziale* associato ad ogni vertice  $i \in N$  e si definisce il *costo ridotto*  $c_{ij}^\pi$  come segue:

$$c_{ij}^\pi = c_{ij} - \pi_i + \pi_j$$

Si noti che il potenziale  $\pi_i$  corrisponde anche alla variabile duale associata al vincolo di conservazione del flusso relativo al vertice  $i \in N$ . I potenziali hanno le seguenti proprietà:

- Per ogni cammino diretto  $P$  dal vertice  $k$  al vertice  $l$  si ha:

$$\sum_{(i,j) \in P} c_{ij}^\pi = \sum_{(i,j) \in P} c_{ij} - \pi_k + \pi_l$$

- Per ogni ciclo diretto  $W$  si ha:

$$\sum_{(i,j) \in W} c_{ij}^\pi = \sum_{(i,j) \in W} c_{ij}$$

### 13.5.2 Condizioni di ottimalità

Se si considera una soluzione ammissibile  $x^*$  del problema di flusso di costo minimo, si può stabilire se è una soluzione ottima applicando una delle seguenti condizioni di ottimalità equivalenti:

- **Assenza Cicli di Costo Negativo**

La soluzione  $x^*$  è ottima se e solo se il grafo residuo  $G(x^*)$  non contiene cicli di costo negativo.

- **Costi Ridotti Positivi**

La soluzione  $x^*$  è ottima se e solo se è possibile trovare dei *potenziali*  $\pi$  tali che per ogni arco  $(i, j)$  di  $G(x^*)$  soddisfano la seguente condizione:

$$c_{ij}^\pi = c_{ij} - \pi_i + \pi_j \geq 0$$

- **Relazione degli Scarti Complementari**

La soluzione  $x^*$  è ottima se e solo se è possibile trovare dei *potenziali*  $\pi$  tali che per ogni arco  $(i, j) \in A$  il *costo ridotto*  $c_{ij}^\pi$  soddisfa le seguenti condizioni degli scarti complementari:

$$\begin{aligned} \text{If } c_{ij}^\pi > 0, \text{ then } x_{ij}^* &= 0 \\ \text{If } c_{ij}^\pi = 0, \text{ then } 0 \leq x_{ij}^* &\leq u_{ij} \\ \text{If } c_{ij}^\pi < 0, \text{ then } x_{ij}^* &= u_{ij} \end{aligned}$$

### 13.5.3 Cycle Cancelling Algorithm

L'algoritmo si basa sulla seguente osservazione: “se nel grafo  $G(x)$  si identifica un ciclo  $W$  di costo negativo, allora è possibile aumentare il flusso in  $W$  diminuendo il costo complessivo della soluzione”.

```
algorithm cycle cancelling
begin
    Stabilisci un flusso ammissibile  $x$  nel grafo  $G$ ;
    while  $G(x)$  contiene un ciclo di costo negativo do
        begin
            Applica un algoritmo per identificare un ciclo di costo negativo  $W$ ;
            Calcola  $\delta = \min\{r_{ij} : (i, j) \in W\}$ ;
            Aumenta di  $\delta$  unita' il flusso nel ciclo  $W$ ;
            Aggiorna  $G(x)$ ;
        end
    end
```

L'algoritmo mantiene ad ogni passo l'ammissibilità della soluzione e cerca di ottenere il soddisfacimento delle condizioni di ottimalità.

Un flusso ammissibile può essere calcolato risolvendo un problema di flusso massimo sul grafo  $G'$  costruito come segue:

- Si aggiunge a  $G$  due nuovi vertici  $s$  e  $t$ ;
- Per ogni vertice  $i \in N$  con  $b_i > 0$  si aggiunge un *arco sorgente*  $(s, i)$  con capacità  $u_{si} = b_i$ ;
- Per ogni vertice  $i \in N$  con  $b_i < 0$  si aggiunge un *arco destinazione*  $(i, t)$  con capacità  $u_{it} = -b_i$ .

Se il flusso massimo da  $s$  a  $t$  calcolato satura tutti gli archi sorgente e destinazione, allora il flusso trovato è ammissibile per il grafo  $G$ , altrimenti per  $G$  non può esistere un flusso ammissibile.

Per identificare un ciclo  $W$  di costo negativo in  $G(x)$  si può usare un algoritmo di tipo “label-correcting” per il calcolo del cammino di costo minimo come, per esempio, l’algoritmo di Bellman-Ford.

Nel caso peggiore il Cycle Cancelling Algorithm esegue  $O(mCU)$  iterazioni. Ad ogni iterazione l’algoritmo risolve il problema del cammino di costo minimo con costi qualsiasi per identificare un ciclo di costo negativo. Nel caso in cui sia utilizzato Bellman-Ford, la ricerca di un ciclo di costo negativo ha complessità pari a  $O(nm)$ . Quindi, nel caso sia impiegato Bellman-Ford, la complessità computazionale del Cycle Cancelling Algorithm è pari a  $O(nm^2CU)$ .

In Figura 13.45 è riportato un esempio di esecuzione del Cycle Cancelling Algorithm.

### 13.5.4 Successive Shortest Path Algorithm

L’algoritmo ad ogni passo mantiene soddisfatte le condizioni di non negatività dei costi ridotti e i vincoli di capacità, mentre cerca di soddisfare i vincoli di conservazione del flusso:

$$\sum_{j \in \Gamma_i} x_{ij} - \sum_{j \in \Gamma_i^{-1}} x_{ji} = b_i$$

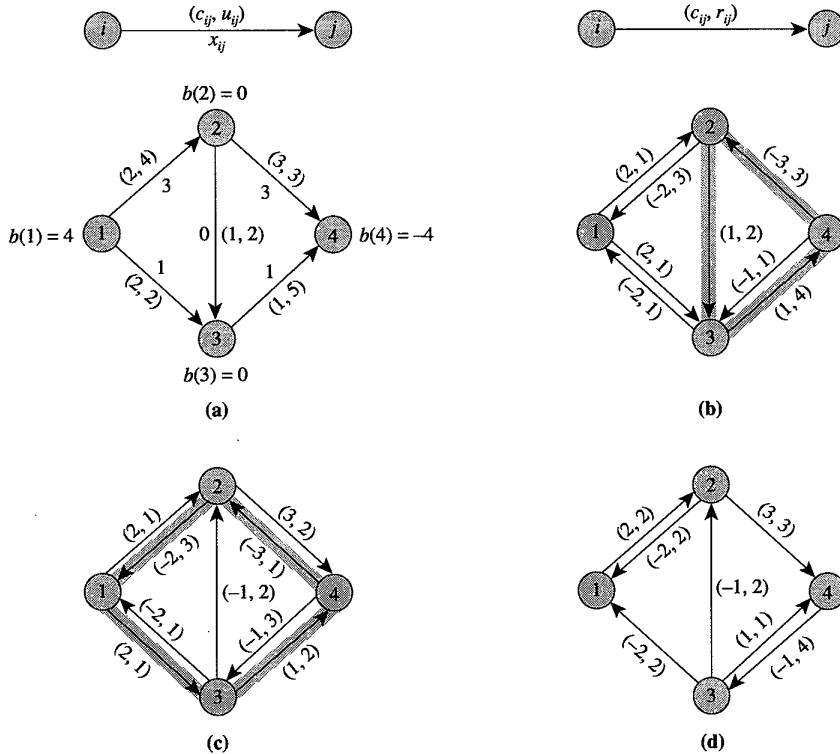
Il flusso  $x$  sarà ammissibile solo al termine dell’algoritmo e quindi nelle fasi intermedie si dirà che  $x$  è un *pseudoflusso*. Inoltre, per ogni vertice  $i$  definiamo:

$$e_i = b_i - \sum_{j \in \Gamma_i} x_{ij} + \sum_{j \in \Gamma_i^{-1}} x_{ji}$$

**Lemma 13.2.** *Sia  $x$  un pseudoflusso che soddisfa le condizioni di non negatività dei costi ridotti per un qualche  $\pi$ , i.e.  $c_{ij}^\pi = c_{ij} - \pi_i + \pi_j \geq 0$ ,  $\forall (i, j)$  in  $G(x)$ .*

*Sia  $d = (d_1, \dots, d_n)$  la distanza minima in  $G(x)$  da un vertice  $s$  a tutti gli altri vertici rispetto ai costi ridotti  $c_{ij}^\pi$ . Le seguenti proprietà sono valide:*

- Il pseudoflusso  $x$  soddisfa le condizioni di non negatività dei costi ridotti anche per i potenziali  $\pi' = \pi - d$ ;
- I costi ridotti  $c_{ij}^{\pi'}$  sono nulli per tutti gli archi  $(i, j)$  nel cammino minimo da  $s$  a ogni altro nodo.  $\square$



**Figure 9.8** Illustrating the cycle canceling algorithm: (a) network example with a feasible flow  $x$ ; (b) residual network  $G(x)$ ; (c) residual network after augmenting 2 units along the cycle 4–2–3–4; (d) residual network after augmenting 1 unit along the cycle 4–2–1–3–4.

Figura 13.45: Esempio di esecuzione del Cycle Cancelling Algorithm (Fonte: Orlin, Ahuja, Magnanti, “Network Flows: Theory, Algorithms, and Applications”, Pearson)

**Lemma 13.3.** Sia  $x$  un pseudoflusso che soddisfa le condizioni di non negatività dei costi ridotti per un qualche potenziale  $\pi$ .

Se  $x'$  è il pseudoflusso ottenuto da  $x$  aumentando il flusso lungo il cammino di costo minimo da  $s$  a un altro vertice  $k$ , allora anche  $x'$  soddisfa le condizioni di non negatività dei costi ridotti per un qualche  $\pi'$ , e.g.  $\pi' = \pi - d$ .  $\square$

Il Lemma suggerisce la seguente strategia: “Ad ogni iterazione l’algoritmo deve selezionare un vertice  $s$  con un *excesso* di flusso, i.e.,  $e_s > 0$ , e un vertice  $t$  con un *deficit* di flusso, i.e.,  $e_t < 0$ , e aumentare il flusso lungo il cammino di costo minimo da  $s$  a  $t$  nel grafo  $G(x)$ ”.

Si noti che nel calcolo del cammino di costo minimo si impiegano i costi ridotti  $c_{ij}^\pi$  che sono positivi. Pertanto, può essere utilizzato l’Algoritmo di Dijkstra.

```

algorithm successive shortest path
begin
   $x = 0$ ,  $\pi = 0$  e  $e_i = b_i$  per ogni  $i \in N$ ;
   $E = \{i \in N : e_i > 0\}$  e  $D = \{i \in N : e_i < 0\}$ ;
  while  $E \neq \emptyset$  do
    begin
      Seleziona un nodo  $k \in E$  e un nodo  $l \in D$ ;
      Calcola in  $G(x)$  le distanze minime  $d$  da  $k$ 
        a tutti gli altri nodi rispetto ai costi ridotti  $c_{ij}^\pi$ ;
      Sia  $P$  il cammino di costo minimo da  $k$  a  $l$ ;
      Aggiorna  $\pi = \pi - d$ ;
      Calcola  $\delta = \min\{e_k, -e_l, \min\{r_{ij} : (i, j) \in P\}\}$ ;
      Aumenta di  $\delta$  unita' il flusso nel cammino  $P$ ;
      Aggiorna  $G(x)$ ;
    end
  end

```

Nel caso peggiore il Successive Shortest Path Algorithm esegue  $O(nU)$  iterazioni. Ad ogni iterazione l'algoritmo risolve il problema del cammino di costo minimo con costi positivi o uguali a zero (i.e., non-negativi).

Nel caso sia utilizzato l'Algoritmo di Dijkstra con Fibonacci Heap la complessità del calcolo dei cammini di costo minimo è pari a  $O(m + n \log n)$ ; quindi la complessità computazionale complessiva del Successive Shortest Path Algorithm è pari a  $O(nU(m + n \log n))$ .

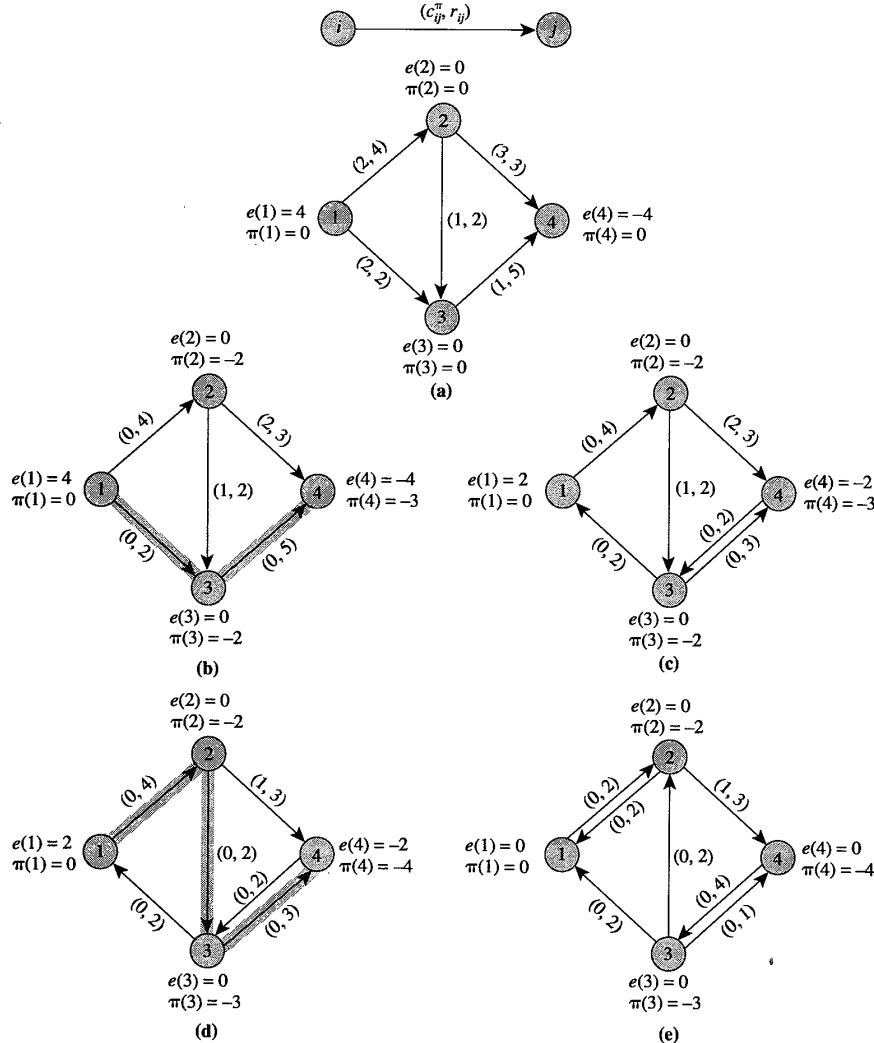
In Figura 13.46 è riportato un esempio di esecuzione del Successive Shortest Path Algorithm.

### 13.5.5 Primal-Dual Algorithm

L'algoritmo primale-duale per la soluzione del problema del flusso di costo minimo è simile al Successive Shortest Path Algorithm, in quanto anch'esso mantiene un pseudoflusso che soddisfa le condizioni di non negatività dei costi ridotti e i vincoli di capacità, mentre i vincoli di conservazione del flusso sono soddisfatti solo al raggiungimento della soluzione ottima.

L'algoritmo primale-duale prevede di trasformare il problema originale in modo tale da avere un solo vertice con un eccesso di flusso e un solo vertice con un deficit di flusso. Quindi il grafo viene modificato come segue:

- Si aggiungono a  $G$  due nuovi vertici  $s$  e  $t$ ;
- Per ogni vertice  $i \in N$  con  $b_i > 0$  si aggiunge un *arco sorgente*  $(s, i)$  con capacità  $u_{si} = b_i$ ;
- Per ogni vertice  $i \in N$  con  $b_i < 0$  si aggiunge un *arco destinazione*  $(i, t)$  con capacità  $u_{it} = -b_i$ ;
- Si pone  $b_s = \sum_{i \in N^+} b_i$ , dove  $N^+ = \{i \in N : b_i > 0\}$ ,  $b_t = -b_s$  e  $b_i = 0$  per ogni  $i \in N$ .



**Figure 9.10** Illustrating the successive shortest path algorithm: (a) initial residual network for  $x = 0$  and  $\pi = 0$ ; (b) network after updating the potentials  $\pi$ ; (c) network after augmenting 2 units along the path 1–3–4; (d) network after updating the potentials  $\pi$ ; (e) network after augmenting 2 units along the path 1–2–3–4.

Figura 13.46: Esempio di esecuzione del Successive Shortest Path Algorithm (Fonte: Orlin, Ahuja, Magnanti, “Network Flows: Theory, Algorithms, and Applications”, Pearson)

Dati i potenziali  $\pi$ , il grafo ammissibile  $G^a(x)$  è un sottografo di  $G(x)$  che contiene solo gli archi  $(i, j)$  di  $G(x)$  di costo ridotto nullo, i.e.  $c_{ij}^\pi = 0$ . La capacità residua  $r_{ij}$  è la stessa del corrispondente arco in  $G(x)$ .

```

algorithm primal-dual
begin
   $x = 0$  e  $\pi = 0$ ;
   $e_s = b_s$  e  $e_t = b_t$ ;
  while  $e_s > 0$  do
    begin
      Calcola in  $G(x)$  le distanze minime  $d$  da  $s$ 
      a tutti gli altri nodi rispetto ai costi ridotti  $c_{ij}^\pi$ ;
      Aggiorna  $\pi = \pi - d$ ;
      Definisci il grafo ammissibile  $G^a(x)$ ;
      Calcola in  $G^a(x)$  il flusso massimo dal  $s$  a  $t$ ;
      Aggiorna  $e_s$ ,  $e_t$  e  $G(x)$ ;
    end
  end

```

Nel caso peggiore l'algoritmo primale duale esegue un numero di iterazioni pari a  $O(\min\{nU, nC\})$ . Ad ogni iterazione l'algoritmo risolve un problema del cammino di costo minimo con costi non negativi e un problema del flusso massimo.

Nel caso sia utilizzato l'Algoritmo di Dijkstra con Fibonacci Heap la complessità del calcolo dei cammini di costo minimo è pari a  $O(m + n \log n)$ . Nel caso sia utilizzato l'Algoritmo Highest-Label Preflow-Push la complessità del calcolo del flusso massimo è pari a  $O(n^2 \sqrt{m})$ . Quindi, in questo caso la complessità computazionale dell'algoritmo primale duale è pari a  $O((\min\{nU, nC\})((m + n \log n) + (n^2 \sqrt{m})))$ .

In Figura 13.47 è riportato un esempio di esecuzione del Successive Primal-Dual Algorithm.

### 13.5.6 Out-of-Kilter Algorithm

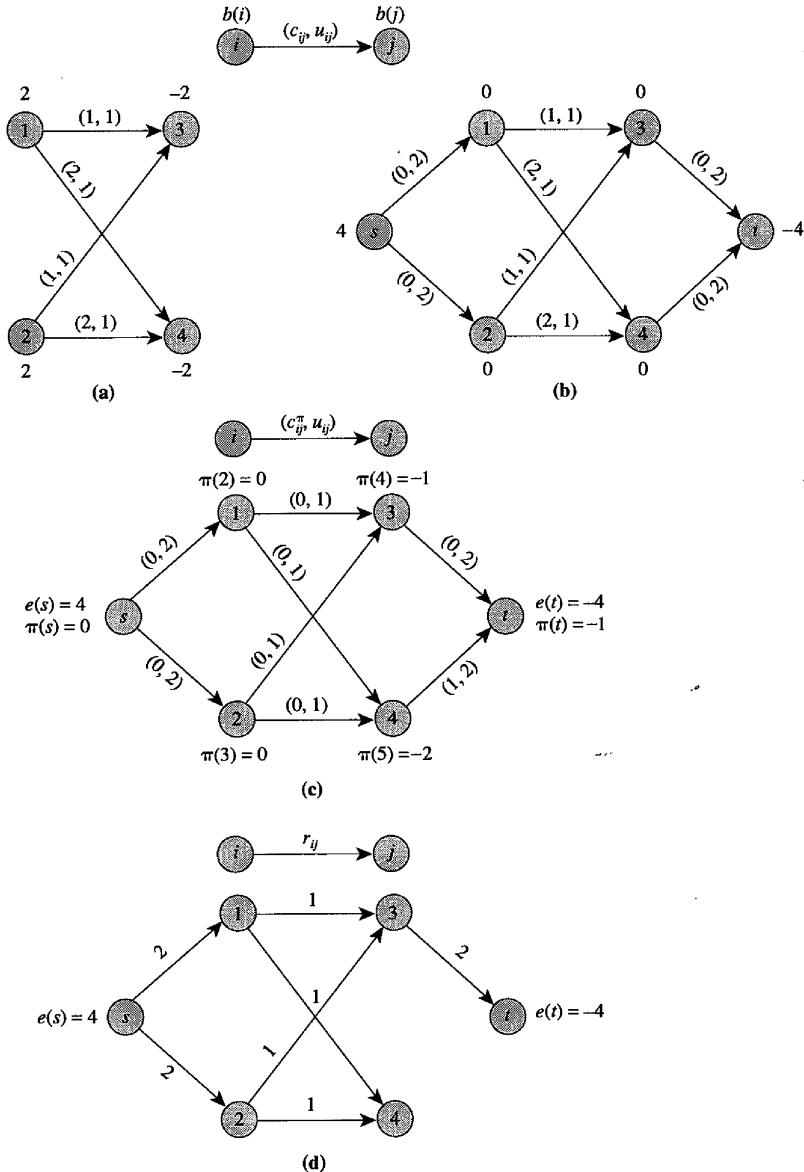
L'algoritmo Out-of-Kilter mantiene soddisfatti i vincoli di conservazione del flusso ad ogni vertice  $i \in N$ , mentre consente la violazione sia delle condizioni di ottimalità che dei vincoli di capacità degli archi  $(i, j) \in A$ .

L'algoritmo iterativamente modifica il flusso  $x$  e i potenziali  $\pi$ , in modo da *diminuire* la non ammissibilità della soluzione e convergere alla soluzione ottima.

Si ricordi che la soluzione  $x^*$  è ottima se e solo se è possibile trovare dei potenziali  $\pi$  tali che per ogni arco  $(i, j) \in A$  il corrispondente costo ridotto  $c_{ij}^\pi$  soddisfa le seguenti condizioni degli scarti complementari:

$$\begin{aligned}
 &\text{Se } c_{ij}^\pi > 0, \text{ allora } x_{ij}^* = 0 \\
 &\text{Se } c_{ij}^\pi = 0, \text{ allora } 0 \leq x_{ij}^* \leq u_{ij} \\
 &\text{Se } c_{ij}^\pi < 0, \text{ allora } x_{ij}^* = u_{ij}
 \end{aligned}$$

Se un arco  $(i, j) \in A$  soddisfa le condizioni di ottimalità si dirà che è *in-kilter*, mentre se non le soddisfa si dirà che è *out-of-kilter*.



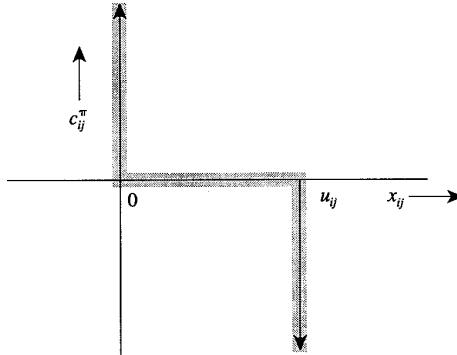
**Figure 9.12** Illustrating the primal-dual algorithm: (a) example network; (b) transformed network; (c) residual network after updating the node potentials; (d) admissible network.

Figura 13.47: Esempio di esecuzione del Primal-Dual Algorithm (Fonte: Orlin, Ahuja, Magnanti, “Network Flows: Theory, Algorithms, and Applications”, Pearson)

Il *Kilter Number*  $k_{ij} \geq 0$  indica di quanto deve essere modificato il flusso  $x_{ij}$  per rendere *in-kilter* l'arco  $(i, j) \in A$  rispetto al costo ridotto  $c_{ij}^\pi$ :

$$\begin{aligned} \text{Se } c_{ij}^\pi > 0, & \quad \text{allora } k_{ij} = |x_{ij}| \\ \text{Se } c_{ij}^\pi = 0 \text{ e } x_{ij} > u_{ij}, & \quad \text{allora } k_{ij} = x_{ij} - u_{ij} \\ \text{Se } c_{ij}^\pi = 0 \text{ e } x_{ij} < 0, & \quad \text{allora } k_{ij} = -x_{ij} \\ \text{Se } c_{ij}^\pi < 0, & \quad \text{allora } k_{ij} = |u_{ij} - x_{ij}| \end{aligned}$$

Il kilter number  $k_{ij}$  indica il livello di non ammissibilità dell'arco  $(i, j) \in A$ . L'algoritmo Out-Of-Kilter ad ogni iterazione individua un arco  $(i, j) \in A$  out-of-kilter e riduce di una quantità positiva il corrispondente kilter number  $k_{ij}$  senza dover aumentare il kilter number degli altri nodi.



**Figure 9.13** Kilter diagram for arc  $(i, j)$ .

Figura 13.48: Kilter number (Fonte: Orlin, Ahuja, Magnanti, "Network Flows: Theory, Algorithms, and Applications", Pearson)

Se si assume di avere un flusso iniziale ammissibile, allora la definizione del kilter number per ogni arco  $(i, j) \in A$  si riduce a:

$$k_{ij} = \begin{cases} 0, & \text{se } c_{ij}^\pi \geq 0 \\ r_{ij}, & \text{se } c_{ij}^\pi < 0 \end{cases}$$

L'Algoritmo Out-Of-Kilter può essere definito come segue:

**algorithm** out-of-kilter  
**begin**

Sia  $\pi = 0$  e  $x$  un flusso ammissibile;  
Calcola  $G(x)$  e il *kilter number*  $k_{ij}$  per ogni arco  $(i, j)$ ;  
**while**  $G(x)$  contiene un arco  $(p, q)$  *out-of-kilter* **do**  
**begin**

Definisci il costo di ogni arco  $(i, j)$  in  $G(x)$  come  $c'_{ij} = \max\{0, c_{ij}^\pi\}$ ;

```

Calcola in  $G(x) - \{(q, p)\}$  le distanze minime  $d$ 
da  $q$  a tutti gli altri nodi rispetto ai costi  $c'_{ij}$ ;
Sia  $P$  il cammino di costo minimo da  $q$  a  $p$ ;
Aggiorna  $\pi' = \pi - d$ ;
if  $c_{pq}^{\pi'} < 0$  then
     $W = P \cup \{(p, q)\}$ ;
    Calcola  $\delta = \min\{r_{ij} : (i, j) \in W\}$ ;
    Aumenta di  $\delta$  unita' il flusso nel ciclo  $W$ ;
    Aggiorna  $x$ ,  $G(x)$ , i costi ridotti  $c_{ij}^\pi$  e i kilter number  $k_{ij}$ ;
end if
end
end

```

Nel caso peggiore l'algoritmo Out-Of-Kilter esegue un numero di iterazioni pari a  $O(mU)$ . Ad ogni iterazione l'algoritmo risolve il problema del cammino di costo minimo con costi non negativi.

Nel caso sia utilizzato l'Algoritmo di Dijkstra con Fibonacci Heap la complessità del calcolo dei cammini di costo minimo è pari a  $O(m + n \log n)$ ; quindi, la complessità computazionale dell'algoritmo Out-Of-Kilter è pari a  $O(mU(m + n \log n))$ .

### 13.5.7 Algoritmi polinomiali

Per ottenere degli algoritmi polinomiali per risolvere il problema del flusso di costo minimo possono essere utilizzate delle tecniche di *scaling*.

#### Capacity Scaling Algorithm

Il Capacity Scaling Algorithm è una variante del Successive Shortest Path Algorithm in cui ad ogni iterazione viene garantito che la capacità residua del cammino aumentante sia “sufficientemente grande”.

Viene definito un parametro  $\Delta$  e ad ogni iterazione:

- si selezionano i vertici  $s$  e  $t$  tali che  $e_s \geq \Delta$  e  $e_t \leq -\Delta$ ;
- si sostituisce il grafo residuo  $G(x)$  con il suo sottografo  $G(x, \Delta)$  contenente solo quegli archi di capacità residua pari almeno a  $\Delta$ .

Quando non è più possibile determinare una coppia di vertici  $s$  e  $t$  tali che  $e_s \geq \Delta$  e  $e_t \leq -\Delta$ , allora si diminuisce il parametro  $\Delta$ , i.e.,  $\Delta = \Delta/2$ . Se  $\Delta = 1$  allora il flusso  $x$  corrente è ottimo.

Se si inizializza  $\Delta = 2^{\lfloor \log U \rfloor}$  allora il Capacity Scaling Algorithm nel caso peggiore eseguirà  $O(m \log U)$  iterazioni. Quindi, nel caso sia impiegato l'Algoritmo di Dijkstra con Fibonacci Heap, la complessità computazionale complessiva del Capacity Scaling Algorithm è pari a  $O((m \log U)(m + n \log n))$ .

### 13.5.8 Algoritmi Fortemente Polinomiali

Algoritmi fortemente polinomiali per risolvere il problema del flusso di costo minimo possono essere ottenuti modificando il Cycle Cancelling Algorithm e il Capacity Scaling Algorithm.

#### Minimum Mean Cycle Cancelling Algorithm

Il Minimum Mean Cycle Cancelling Algorithm è un caso particolare del Cycle Cancelling Algorithm in cui a ogni iterazione il flusso è aumentato nel ciclo  $W$  di costo negativo del grafo  $G(x)$  che ha *costo medio* minimo. Il costo medio di un ciclo  $W$  è dato da  $(\sum_{(i,j) \in W} c_{ij})/|W|$ .

Per calcolare il ciclo  $W$  di costo medio minimo può essere impiegata una procedura di programmazione dinamica di complessità  $O(nm)$ .

La complessità computazionale del Minimum Mean Cycle Cancelling Algorithm è pari a  $O(n^2m^3 \log n)$ .

#### Enhanced Capacity Scaling Algorithm

L'Enhanced Capacity Scaling Algorithm è una variante del Capacity Scaling Algorithm ed è basato sul seguente lemma.

**Lemma 13.4.** *Quando il Capacity Scaling Algorithm aggiorna il parametro  $\Delta$ , i.e.,  $\Delta = \Delta/2$ , se nell'arco  $(i, j)$  il flusso è tale che  $x_{ij} \geq 8n\Delta$  allora:*

- *in qualsiasi soluzione ottima  $x_{ij} > 0$ .*
- *il costo ridotto  $c_{ij}^\pi$  è sempre nullo comunque siano scelti i potenziali  $\pi$  ottimi.  $\square$*

Quando l'algoritmo identifica un arco  $(i, j)$  tale che  $x_{ij} \geq 8n\Delta$ , allora i potenziali dei vertici  $i$  e  $j$  possono essere fissati. Quindi, l'arco  $(i, j)$  viene “contratto” e i vertici  $i$  e  $j$  sono sostituiti da un nuovo vertice. L'operazione di contrazione da origine a un nuovo problema di flusso di costo minimo con un vertice in meno. L'algoritmo non effettua le operazioni di contrazione esplicitamente ed è in grado di procedere senza dover ricalcolare i flussi e i potenziali per il nuovo problema ridotto.

La complessità computazionale dell'Enhanced Capacity Scaling Algorithm è pari a  $O((m \log n)(m + n \log n))$ .