

Fondamenti di Ricerca Operativa

Lezioni

Riferimenti

1. Fondamenti di Ricerca Operativa

A.A. 2018/2019

Giuliana Carello

DEIB, Politecnico di Milano

Giuliana CARELLO

DEIB - edificio 20 - II piano

tel. 02 2399 3684

giuliana.carello at polimi.it

ricevimento giovedì 14.00 - 16.00 (controllare avvisi di sospensione sul Beep)

materiale del corso disponibile su Beep

- ▶ materiale didattico
 - ▶ slide
 - ▶ esercizi svolti e proposti
 - ▶ temi d'esame
- ▶ avvisi
- ▶ calendario del laboratorio
- ▶ testi suggeriti

Orari e aule

- Giovedì 10.15 - 12.15: lezione/esercitazione, aula E.G.6
- Venerdì 10.15 - 12.15: lezione/esercitazione, aula D.0.1
- 5 incontri di laboratorio (Dott. Alessandro Giovannini):
 - ▶ venerdì 28 Settembre 2018, 13.15 - 15.15, aula N.1.1
 - ▶ venerdì 12 Ottobre 2018, 13.15 - 15.15, aula N.1.1
 - ▶ venerdì 26 Ottobre 2018, 13.15 - 15.15, aula N.1.1
 - ▶ venerdì 23 Novembre 2018, 13.15 - 15.15, aula N.1.1
 - ▶ venerdì 14 Dicembre 2018, 13.15 - 15.15, aula N.1.1

Regole d'esame

- ▶ L'esame è una prova scritta (esercizi, domande teoriche, prova di laboratorio)
- ▶ Libri, appunti, portatili e notepad non sono permessi
- ▶ Non c'è prova in itinere
- ▶ Possibilità di congelare il voto (fino all'inizio del corso nel prossimo anno accademico)

Ricerca Operativa

Cosè la Ricerca Operativa?

La **Ricerca Operativa** è la disciplina che fornisce **metodi quantitativi e tecniche matematiche** (ottimizzazione, teoria dei giochi, simulazione) per risolvere problemi decisionali complessi in cui si devono:

- ▶ gestire risorse limitate (non tutte le soluzioni sono accettabili)
- ▶ tenere conto di un obiettivo per valutare la bontà delle diverse soluzioni
- ▶ selezionare una tra le molte soluzioni accettabili alternative così da ottimizzare l'obiettivo.

È un campo multidisciplinare all'intersezione tra matematica applicata, informatica, economia e ingegneria gestionale.

Scienza di supporto alle decisioni

- ▶ Relativamente giovane: nasce con la II Guerra Mondiale, quando a un gruppo di scienziati venne chiesto di trovare il modo più efficiente di gestire le operazioni militari (posizione dei radar, logistica). Dall'inglese "research on operations".
- ▶ In seguito le tecniche sono state applicati in ambito industriale, sociale o economico e hanno prodotto significativi benefici. È anche detta **Management Science**.
- ▶ Molte sono le applicazioni a casi reali: infatti oggi, in un contesto molto competitivo, è fondamentale individuare e applicare soluzioni efficienti.

Esempi di applicazioni a problemi reali

Turnazione del personale ospedaliero

Si devono determinare i migliori turni per un reparto ospedaliero e assegnarli al personale infermieristico in modo da garantire una adeguata copertura dei turni minimizzando il numero di infermieri o il loro carico di lavoro

Progetto di reti di telecomunicazione

Si devono collegare alcune città, che rappresentano gli utenti della rete, con canali trasmissivi in modo da garantire la connessione di tutte le città minimizzando il costo di installazione della rete

Esempi di applicazioni a problemi reali

Ricerca del percorso più breve

Data una mappa che rappresenta la rete stradale di un'area geografica con il loro tempo di percorrenza, si deve selezionare il percorso più rapido tra due punti

Gestione di portfolio

Si deve selezionare un insieme di investimenti (opzioni, buoni, azioni) tra quelli disponibili sul mercato che massimizzi il guadagno minimizzando i rischi

Il processo di ottimizzazione

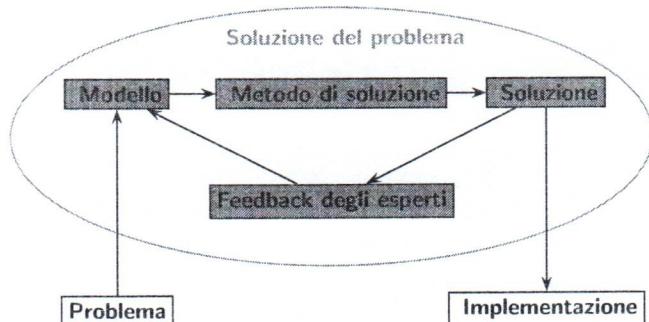
Un esempio di applicazione: le ambulanze a Milano

- Sono posizionate in postazioni (colonnine) dove attendono la chiamata per una missione
- In casi gravi devono raggiungere il paziente entro 8 minuti
- Si fanno carico del paziente finché non viene accettato dal pronto soccorso

Problema

Come posizionare le ambulanze in modo da garantire un buon servizio senza usarne troppe?

Il processo di ottimizzazione



Struttura del modello di Programmazione Matematica

- I dati → i parametri :
 - I : insieme dei punti di origine della domanda/insieme delle possibili colonnine
 - il parametro a_{ij} ci dice se dal punto i si può raggiungere il punto j in meno di 8 minuti:
 $a_{ij} = 1$ se da i si raggiunge j in meno di 8 minuti, 0 altrimenti

Modello matematico

Funzione obiettivo

$$\min \sum_{i \in I} x_i$$

Vincolo

$$\sum_{i \in I} a_{ij} x_i \geq 1, \quad \forall j \in J$$

Dominio delle variabili

$$x_i \in \{0, 1\}, \quad \forall i \in I$$

Lo stesso modello può rappresentare anche problemi legati ad altre applicazioni

Il processo di ottimizzazione

Cosa sappiamo?

- Da dove è originata la domanda (griglia della città)
- Dove possiamo mettere le ambulanze (griglia della città)
- Quali punti della griglia sono più vicini di 8 minuti

Cosa dobbiamo decidere?

Dove mettere le ambulanze

Cosa fareste?

Il processo di ottimizzazione

- Rappresentiamo il problema con un modello matematico
 → Formulazione in Programmazione Matematica
 Definiamo il problema che vogliamo modellare
 - Usare il minor numero di ambulanze
 - Garantendo di arrivare ovunque entro 8 minuti
- Risolviamo il problema con tecniche di ottimizzazione
- Analizziamo la soluzione con gli esperti: risponde alle loro esigenze?

Struttura del modello di Programmazione Matematica

- Le decisioni → le variabili decisionali: le decisioni (dove mettere le ambulanze) sono rappresentate da variabili: x_i ($\forall i \in I$) metto o no una ambulanza nel posto i ?
- L'obiettivo → funzione obiettivo: l'obiettivo (usare il minor numero di ambulanze possibile) è rappresentato dalla funzione obiettivo
- I requisiti → i vincoli: i requisiti che una soluzione deve soddisfare (posso raggiungere tutti i malati gravi in meno di 8 minuti) sono descritti da diseguaglianze, i vincoli, che i valori assegnati alle variabili devono soddisfare

Modello matematico

- Assegnando un valore alle variabili abbiamo una possibile soluzione
- Se il valore delle variabili rispetta i vincoli abbiamo una soluzione ammissibile
- L'insieme delle soluzioni ammissibili è la regione ammissibile. Nell'esempio delle ambulanze la regione ammissibile X è

$$X = \left\{ \{x_i\} : \sum_{i \in I} a_{ij} x_i \geq 1, \forall j \in J, x_i \in \{0, 1\}, \forall i \in I \right\}$$

- Valutiamo la bontà delle soluzioni con la funzione obiettivo. Nel caso delle ambulanze la funzione obiettivo $f(x)$ è $f(x) = \sum_{i \in I} x_i$.

Ottimizzazione

Ottimizzare

Ottimizzare significa trovare la soluzione ammissibile globalmente ottima, ovvero migliore (o non peggiore) di ogni altra soluzione ammissibile.

Nell'esempio delle ambulanze $x^* \in X$ è globalmente ottima se

$$f(x^*) \leq f(x), \forall x \in X$$

ovvero se usa il minor numero possibile di ambulanze

Programmazione Matematica

Modello in Programmazione Lineare

- ▶ L'espressione di vincoli e funzione obiettivo sono funzioni lineari

- ▶ Somme di variabili moltiplicate per coefficienti noti
- ▶ Non ci sono potenze superiori a 1 né prodotto di variabili

→ Modello in Programmazione Lineare

Altri approcci?

Fin qui abbiamo considerato un modello statico con un solo decisore e un solo obiettivo. Ad esempio, la programmazione lineare e la sua soluzione: rappresenta i requisiti che una soluzione deve avere, valuta le soluzioni secondo l'obiettivo, e cerca la migliore. Ma il sistema può esser dinamico e si possono considerare più obiettivi o più decisori.

- ▶ *Simulazione*: emula il comportamento dinamico del sistema permettendo di valutare l'impatto delle scelte in contesto realistico (studio what-if)
- ▶ *Teoria dei Giochi*: considera problemi in cui non c'è un unico decisore, ma molti (ad esempio per rappresentare un mercato e i suoi diversi agenti)
- ▶ *Ottimizzazione Multi-criteria*: studia i casi in cui più criteri concorrono a valutare la bontà di una soluzione

Programmazione Matematica

- ▶ Tutte le variabili sono intere → Programmazione Lineare Intera
- ▶ Alcune variabili sono intere altre sono continue → Programmazione Lineare Mista Intera
- ▶ La funzione obiettivo e/o i vincoli sono rappresentati da funzioni non lineari → Programmazione Non Lineare (eventualmente Programmazione Non Lineare Mista Intera)
- ▶ Alcuni dati sono incerti → Programmazione Stocastica

Argomento del corso

- ▶ Modellazione in Programmazione Lineare
- ▶ Proprietà della Programmazione Lineare (a variabili continue) e metodi di soluzione
- ▶ Una classe particolare di problemi di Ottimizzazione: i problemi su grafo
- ▶ Metodi per la Programmazione Lineare Intera

I laboratori sono dedicati alla modellazione e ai software per la soluzione di problemi in Programmazione Lineare (Intera e no)

2. Modelli di Programmazione Lineare

Fondamenti di Ricerca Operativa - A.A. 2018/2019

Giuliana Carello
DEIB, Politecnico di Milano

- ▶ Modellazione in Programmazione Lineare: esempi e panoramica di diversi tipi di variabili, vincoli e funzioni obiettivo
- ▶ Proprietà della Programmazione Lineare e metodi di soluzione
- ▶ Una classe particolare di problemi di Ottimizzazione: i problemi su grido
- ▶ Metodi per la Programmazione Lineare Intera

Struttura del modello

- ▶ I dati → gli insiemi e i parametri
- ▶ Le decisioni → le variabili decisionali
- ▶ I requisiti → i vincoli
- ▶ L'obiettivo → la funzione obiettivo

Il coltivatore

Un coltivatore ha a disposizione 12 ettari di terreno da coltivare a lattuga o a patate. Le risorse a sua disposizione, oltre al terreno, sono: 70 kg. di semi di lattuga, 18 t. di tuberi, 160 t. di stallatico per concimare il terreno. Supponendo che il mercato sia in grado di assorbire tutta la produzione e che i prezzi siano stabili, la resa stimata della coltivazione a lattuga è di 3000 euro per ettaro e quella delle patate è di 5000 euro per ettaro. L'assorbimento delle risorse è di 7 kg di semi e 10 t di stallatico per ettaro di lattuga, 3 t di tuberi e 20 di stallatico per le patate. Il coltivatore deve stabilire quanto terreno destinare a lattuga e quanto a patate in modo da massimizzare la resa economica e sfruttando al meglio le risorse disponibili.

Modello

Dati

- ▶ Numero e tipi delle coltivazioni
- ▶ Quantità di semi per ettaro richiesta da ogni coltivazione
- ▶ Quantità di concime per ettaro richiesta da ogni coltivazione
- ▶ Resa per ettaro di ogni coltivazione
- ▶ Quantità di risorse disponibili

Modello

Decisioni → variabili decisionali

- ▶ Ettari coltivati a lattuga → x_L
- ▶ Ettari coltivati a patate → x_P

Dominio delle variabili?

$x_L, x_P \geq 0 \rightarrow$ variabili continue non negative

Modello

Massimizzare il guadagno → funzione obiettivo
Guadagno:

$$f(x_L, x_P) = 3000x_L + 5000x_P$$

massimizzare il guadagno →

$$\max 3x_L + 5x_P$$

Modello

Requisiti e limiti delle risorse → vincoli

- ▶ Terreno disponibile: $x_P + x_L \leq 12$
- ▶ Disponibilità di semi di lattuga: $7x_L \leq 70$
- ▶ Disponibilità di tuberi: $3x_P \leq 18$
- ▶ Disponibilità di stallatico: $10x_L + 20x_P \leq 160$

Dominio $x_L, x_P \geq 0$

Vincoli che limitano l'uso di risorse → Vincoli di budget

Rappresentazione matriciale e parametrica: dati

Possibili coltivazioni

Insieme delle possibili coltivazioni $\rightarrow I$

Nel nostro caso $I = \{\text{lattuga, patate}\}$

Coefficienti della funzione obiettivo

Vettore di $|I|$ elementi:

$$c_1, c_2, \dots, c_{|I|}$$

Nel nostro caso

$$\mathbf{c}^T = \{c_L, c_P\} = \{3, 5\}$$

Il vettore $\mathbf{c} = (c_1, c_2, \dots, c_{|I|})$ rappresenta il gradiente della funzione obiettivo

Rappresentazione matriciale e parametrica: vincoli

Matrice dei coefficienti dei vincoli

matrice A di ordine a $m \times n$

dove m è il numero dei vincoli, $n = |I|$ il numero delle variabili

► righe \rightarrow vincoli

► colonne \rightarrow variabili

► a_{ij} \rightarrow coefficiente della j -esima variabile nel i -esimo vincolo

$$\text{nel nostro caso } A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 2 \end{pmatrix}$$

Rappresentazione matriciale e parametrica: vincoli

Vettore dei termini noti

$$\mathbf{b}^T = \{b_1, b_2, \dots, b_n\}$$

nel nostro caso $\mathbf{b}^T = \{12, 10, 6, 16\}$

Rappresentazione matriciale e parametrica: variabili

Variabili

Vettore di $|I|$ elementi:

$$x_1, x_2, \dots, x_{|I|}$$

Nel nostro caso $I = \{\text{lattuga, patate}\}$

$$\mathbf{x}^T = \{x_L, x_P\}$$

Rappresentazione matriciale e parametrica: modello

Modello in forma matriciale

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

Alcune definizioni

Consideriamo un problema nella forma

$$(P) \quad \begin{aligned} \max & c^T x \\ Ax & \leq b \\ x & \geq 0 \end{aligned}$$

Soluzione ammissibile

x^* è una soluzione ammissibile di P se soddisfa tutti i vincoli del problema, ovvero $Ax^* \leq b$ e $x^* \geq 0$.

Alcune definizioni

Rappresentazione grafica

$\{x \in \mathbb{R}^n : a^T x = b\}$ è un iper piano e definisce un semispazio (affine) $\{x \in \mathbb{R}^n : a^T x \leq b\}$

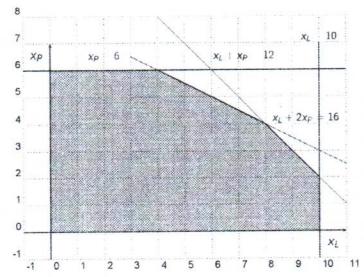


Figura: Regione ammissibile

Regione ammissibile

L'insieme delle soluzioni ammissibili di P ,

$X = \{x : Ax \leq b, x \geq 0\}$, è detta regione ammissibile di P .

Soluzione ottima

Una soluzione $\bar{x} \in X$ è ottima per P se $c^T \bar{x} \geq c^T x, \forall x \in X$.

N.B. Se consideriamo una funzione obiettivo da minimizzare, la soluzione $\bar{x} \in X$ è ottima se $c^T \bar{x} \leq c^T x, \forall x \in X$.

Coltivatore

<p>12 ettari</p> <p>160 t</p> <p>stallico per la concimazione</p>	<p>lattuga 70 kg semi → 3000 €/ettaro</p> <p>patate 18 t tuberi → 5000 €/ettaro</p>	<p>7 kg/ettaro</p> <p>3 t/ettaro</p>	<p>stallico</p> <p>10 t/ettaro</p> <p>20 t/ettaro</p>
---	---	--------------------------------------	---

$$x_p = \text{ettari coltivati di patate}$$

$$x_p \geq 0$$

$$x_L = \text{ettari coltivati di lattuga}$$

$$x_L \geq 0$$

Obiettivo: massimizzare il guadagno:

$$f(x_L, x_p) = 3000x_L + 5000x_p$$

$$\max(3x_L + 5x_p)$$

Ogni dato come un limite
le variabili?

Vincoli: $x_L + x_p \leq 12$

$$7x_L \leq 70$$

$$3x_p \leq 18$$

$$10x_L + 20x_p \leq 160$$

$$I = \{ \text{lattuga, patate} \}$$

$$c^T = [c_L, c_p] = [3, 5]$$

$$\dim(c) = \dim(I)$$

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

nighe → vincoli
colonne → variabili
 a_{ij} = coefficiente della j -esima variabile nell' i -esimo vincolo

$$\dim(A) = m \times n = \# \text{vincoli} \times \dim(I)$$

$$b = [12, 10, 6, 16]^T$$

$$\dim(b) = \# \text{vincoli}$$

$$x^T = [x_L, x_p]$$

$$\dim(x) = \dim(I)$$

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

Investimento capitale

B euro da investire: n strumenti finanziari : $\begin{cases} w_i = \max \text{ quantità acquistabile} \\ p_i = \text{ritorno se si acquista tutto} \end{cases}$

y_i = quanto acquistato di ciascun strumento

Obiettivo : $\max \sum_{i=1}^n \left(\frac{y_i}{w_i} \right) p_i = \max \sum_{i=1}^n (\text{percentuale}) \cdot (\text{ritorno se acquistata tutto})$

Vincoli : $\sum_{i=1}^n y_i \leq B$

$y_i \leq w_i$

$$\max \sum_{i=1}^n \left(\frac{y_i}{w_i} \right) p_i = \max \sum_{i=1}^n x_i p_i \quad 0 \leq x_i \leq 1 \quad x_i$$

Vincoli : $\sum_{i=1}^n x_i w_i \leq B$

$\cancel{y_i \leq w_i}$ perché dico già che $x_i \in [0, 1]$

Risoluzione grafica

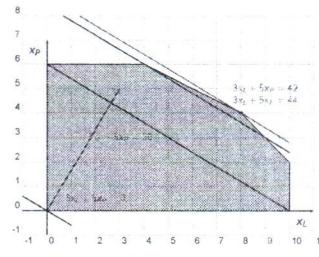


Figura: Funzione obiettivo

Le rette sono curve di livello della funzione obiettivo.

Soluzione ottima $x_L = 8, x_P = 4$

Oss: la soluzione ottima si trova sulla frontiera.

Investimento di capitale

Si consideri il problema di decidere come investire un capitale di B euro avendo a disposizione n strumenti finanziari, per massimizzare il ritorno totale. Ogni strumento è caratterizzato da una quantità massima acquistabile w_i , e dall'entità della cedola staccata p_i (ovvero il ritorno monetario se si acquista tutta l'entità disponibile).

Scrivere il modello in Programmazione Lineare del problema.

Investimento di capitale: dati

- Capitale B
- Numero degli investimenti disponibili n
- Quantità massima acquistabile w_i e entità della cedola p_i per ogni strumento finanziario $i = 1, \dots, n$

Investimento di capitale: primo modello

Variabili

$$y_i \geq 0, \quad \forall i = 1, \dots, n$$

importo investito in ogni strumento finanziario

Funzione obiettivo

$$\max \sum_{i=1}^n p_i \frac{y_i}{w_i}$$

Investimento di capitale: primo modello

Vincoli

- Vincolo sul budget: $\sum_{i=1}^n y_i \leq B$
- Vincolo sulla massima quantità acquistabile:

$$y_i \leq w_i, \quad \forall i = 1, \dots, n$$

Investimento di capitale: secondo modello

Variabili

$$\max \sum_{i=1}^n p_i \frac{y_i}{w_i}$$

$$0 \leq x_i \leq 1 \quad \forall i = 1, \dots, n$$

frazione del massimo importo disponibile investito in ogni strumento finanziario

Variabili frazionarie

$x_i \in [0, 1] \rightarrow$ variabili frazionarie

Investimento di capitale: secondo modello

Funzione obiettivo

$$\max \sum_{i=1}^n p_i x_i$$

Investimento di capitale: variante

Variante

Come cambia il modello se non è possibile frazionare l'acquisto degli strumenti? Per ogni investimento posso comprare tutta la disponibilità o nulla?

Variabili binarie

$$x_i \in [0, 1] \rightarrow x_i \in \{0, 1\}$$

Variabili binarie

Vincoli

- Vincolo sul budget: $\sum_{i=1}^n w_i x_i \leq B$
- Vincolo sulla massima quantità acquistabile non è più necessario

Problema dello zaino

Dati

- Insieme di oggetti I
 - peso $w_i \quad \forall i \in I$
 - profitto $p_i \quad \forall i \in I$
- Capacità dello zaino B

Problema dello zaino

- Decidere quali oggetti inserire nello zaino
- Garantendo che il loro peso non superi la capacità dello zaino
- Massimizzando il profitto

Problema dello zaino

Modello

$$\begin{aligned} & \max \sum_{i \in I} p_i x_i \\ & \sum_{i \in I} w_i x_i \leq B \\ & x_i \in \{0, 1\} \quad \forall i \in I \end{aligned}$$

Dieta

Si consideri il problema di determinare la dieta giornaliera in una mensa scolastica avendo a disposizione vari cibi ognuno con le proprie caratteristiche nutrizionali. Prendiamo ad esempio la tabella con i tipici cibi di una mensa scolastica, per ognuno dei quali è noto l'apporto di calorie (c_i) e il prezzo di acquisto (p_i), entrambi riferiti ad un etto:

Cibo	Spaghetti	Riso	Lonza	Carote	Insalata	Mela
Prezzo (hg)	0.2	0.5	1.2	0.10	0.10	0.2
Calorie (hg)	300	250	200	70	180	100

La dieta deve fornire almeno 700 calorie. Si vuole trovare il menù a costo minimo, decidendo la quantità di ogni cibo impiegata.

Dieta: modello

Funzione obiettivo

$$\min \sum_{i \in F} p_i x_i$$

Vincoli

- Fabbisogno di calorie: $\sum_{i \in F} c_i x_i \geq d$

Vincoli di fabbisogno

Dieta: modello

Dati

- Insieme dei cibi F
 - prezzo $p_i \quad \forall i \in F$
 - calorie fornite $c_i \quad \forall i \in F$
- Calorie richieste d

Variabili

$$x_i \geq 0, \quad \forall i \in F$$

quanti hg di ogni cibo sono usati

Dieta: varianti

Qualità

- Almeno il 30% delle calorie fornite deve derivare da verdura e frutta.

Vincolo

Insieme V della verdura e della frutta (mela, carote, insalata)

$$\frac{\sum_{i \in V} c_i x_i}{\sum_{i \in F} c_i x_i} \geq 0.3$$

Il vincolo non è lineare

$$\sum_{i \in V} c_i x_i \geq 0.3 \sum_{i \in F} c_i x_i$$

Vincolo di qualità (o di blending)

Dieta: varianti

Logico

- Non possono essere presenti nel menù sia gli spaghetti che il riso.

Variabili logiche

- $y_S \in \{0, 1\}$: vale 1 se gli spaghetti fanno parte della dieta
- $y_R \in \{0, 1\}$: vale 1 se il riso fa parte della dieta

Vincoli

- Vincolo logico: $y_R + y_S \leq 1$

Dieta: varianti

Vincoli

Vincoli che collegano x e y :

$$\begin{cases} x_S > 0 \Rightarrow y_S = 1 \\ y_S = 0 \Rightarrow x_S = 0 \end{cases} \Rightarrow \begin{cases} x_S \leq M_{y_S} \\ x_S \leq M_{y_R} \end{cases}$$

Dove M è una costante sufficientemente grande da non limitare il valore delle variabili x

Vincoli di big M

Dieta: varianti

Quantità minima

- Se inserisco il riso nella dieta devo inserirne almeno q hg.

Vincolo

$$x_R \geq qy_R$$

$$x_R \leq My_R$$

Altri vincoli logici

Richieste logiche e loro rappresentazione come vincoli

Date tre scelte, a , b e c , rappresentate da tre variabili binarie y_a, y_b, y_c

- Si deve scegliere al più una tra a e b : $y_a + y_b \leq 1$
- Si deve scegliere esattamente una tra a e b : $y_a + y_b = 1$
- Si deve scegliere almeno una tra a e b : $y_a + y_b \geq 1$

Apertura di ambulatori

In una regione rurale sono presenti alcuni villaggi, descritti dall'insieme $C = \{1, \dots, n\}$. Si vogliono aprire dei presidi sanitari per servire la zona e sono stati individuati dei siti potenzialmente adatti, descritti dall'insieme $S = \{1, \dots, m\}$. Ad ogni sito $j \in S$ è associato un costo di apertura dell'ambulatorio f_j . Ogni villaggio sarà assegnato ad un presidio che dovrà prendere in cura i suoi abitanti. Se il villaggio $i \in C$ sarà assegnato all'ambulatorio aperto in $j \in S$, i suoi abitanti dovranno percorrere p_{ij} chilometri per essere curati. Il budget complessivo per l'apertura degli ambulatori è B . Si vuole garantire il servizio a tutti gli abitanti della regione, minimizzando il più lungo percorso che gli abitanti dovranno percorrere.

Apertura di ambulatori - modello

Variabili

- Dove installare gli ambulatori? $y_j \in \{0, 1\}, \forall j \in S$
- A quale ambulatorio assegnare ogni villaggio?

$$x_{ij} \in \{0, 1\}, \forall i \in C, j \in S$$

$$x_{ij} = \begin{cases} 1, & \text{se il villaggio } i \text{ è assegnato all'ambulatorio } j; \\ 0, & \text{altrimenti.} \end{cases}$$

Dieta: varianti

- ci può essere al più una fonte di carboidrati: $y_S + y_R \leq 1$

- ci deve essere almeno una fonte di vitamine:

$$y_C + y_I + y_M \geq 1$$

- ci deve essere esattamente una verdura: $y_C + y_I = 1$

Altri vincoli logici

Implicazioni

- Se si sceglie a si deve scegliere anche b : $y_a \leq y_b$
- Se si sceglie sia a che b allora si deve scegliere anche c : $y_a + y_b - 1 \leq y_c$
- Se si sceglie almeno una tra a e b , si deve scegliere anche c : $y_a + y_b \leq 2y_c$

Estendiamo l'implicazione:

Se almeno m tra n variabili logiche y_1, \dots, y_n sono vere (hanno valore 1) allora anche w deve valere 1:

$$y_1 + y_2 + \dots + y_n - m + 1 \leq (n - m + 1)w$$

Apertura di ambulatori - modello

Dati

- $C = \{1, \dots, n\}$
- $S = \{1, \dots, m\}$
- $f_j, \forall j \in S$
- $p_{ij}, \forall i \in C, \forall j \in S$

Apertura di ambulatori - modello

Funzione obiettivo

Minimizzare il più lungo percorso per gli abitanti della regione:

$$\min \max_{\substack{i \in C \\ j \in S}} \{p_{ij}x_{ij}\}$$

$$\min f(x), f(x) = \max_{\substack{i \in C \\ j \in S}} \{p_{ij}x_{ij}\}$$

! $f(x)$ non è lineare

Apertura di ambulatori - modello

Linearizzazione

- Si usa una variabile ausiliaria z che deve rappresentare il valore di $\max_{\substack{i \in C \\ j \in S}} p_{ij} x_{ij}$
- La funzione obiettivo diventa $\min z$
- Si aggiungono vincoli per garantire il corretto valore di z

$$z \geq p_{ij} x_{ij}, \quad \forall i \in C, j \in S$$

Vincoli di bottleneck

Altri vincoli di bottleneck

$\max \min$

$$\max \min_{i \in I} \{x_i\}$$

Variabile ausiliaria $z = \min_{i \in I} \{x_i\}$

$\max z$

$$z \leq x_i, \forall i \in I$$

Variabili libere

Orario dei treni

Dato un insieme di treni $T = \{1, \dots, n\}$ e un insieme di stazioni presso le quali i treni fermano $S = \{1, \dots, m\}$, è noto il tempo di arrivo a_{ts} per ogni coppia (t, s) , $t \in T, s \in S$. Si vuole studiare l'effetto di una variazione dei tempi di arrivo. Come possiamo rappresentare questa variazione?

Variabili discrete

Dimensionamento della capacità di una rete di TLC
Nel dimensionamento delle reti di telecomunicazione spesso si pone il problema di allocare capacità di trasmissione sui collegamenti diretti tra due nodi della rete. Tale capacità è fornita installando dei canali trasmissivi, che hanno capacità fissata. Di conseguenza, la capacità su una tratta non può assumere qualunque valore, ma solo un insieme di valori prefissati $\{d_1, d_2, \dots, d_r\}$. Come rappresentare la decisione sul dimensionamento della capacità in ogni tratta?

Apertura di ambulatori - modello

Altri vincoli

- Vincolo di budget: $\sum_{j \in S} f_j y_j \leq B$
- Assegnamento di ogni villaggio ad un'ambulatorio: $\sum_{j \in S} x_{ij} = 1, \forall i \in C$
- Coerenza tra le variabili y e le variabili x :

$$x_{ij} \leq y_j, \quad \forall i \in C, j \in S$$

o anche

$$\sum_{i \in C} x_{ij} \leq |C| y_j, \quad \forall j \in S$$

Altri vincoli di bottleneck

$\min |x|$

$$|x| = \max\{x, -x\}$$

Variabile ausiliaria $z = |x|$

$\min z$

$$z \geq x$$

$$z \geq -x$$

Variabili libere

Variabili libere

Di quanto varia ogni orario di arrivo? $\rightarrow \pi_{ts} \leq 0$:

$$\begin{cases} \pi_{ts} > 0, & \text{l'orario è ritardato;} \\ \pi_{ts} < 0, & \text{l'orario è anticipato;} \\ \pi_{ts} = 0, & \text{l'orario è invariato.} \end{cases}$$

L'orario modificato sarà $a_{ts} + \pi_{ts}$.

Variabili discrete

Variabili

- Quanta capacità è installata su ogni tratta l ? $w_l \in \{d_1, d_2, \dots, d_r\}$ (w_l prende valore in un insieme)
- In alternativa si usano r variabili binarie:
 $y_i^l = \begin{cases} 1, & \text{se la capacità di } l \text{ è pari a } d_i; \\ 0, & \text{altrimenti.} \end{cases}, \forall i = 1, \dots, r$
- Bisogna garantire il corretto valore delle variabili y :
 - Uno e un solo valore di capacità è scelto: $\sum_{i=1}^r y_i^l = 1$
 - Relazione con le variabili w_l : $w_l = \sum_{i=1}^r d_i y_i^l$

Oleodotto

Un'azienda petrolifera gestisce un insieme di stabilimenti N : alcuni di questi sono pozzi ($P \subset N$), che producono greggio, altri sono raffinerie ($R \subset N$), che lo lavorano, altri ancora sono centri di smistamento che non possono immagazzinare, lavorare o produrre il greggio. Per ogni pozzo $i \in P$ è nota la quantità di greggio prodotta d_i , mentre per ogni raffineria $j \in R$ è nota la quantità di greggio richiesta r_j . I vari stabilimenti sono collegati da tratti di oleodotto: ogni tratto di oleodotto, descritto dalla coppia di stabilimenti che congiunge (i, j) , ha una massima portata u_{ij} e un costo c_{ij} che rappresenta il costo di inviare un barile di greggio sulla tratta di oleodotto (i, j) . L'azienda deve decidere come trasportare il greggio dai pozzi alle raffinerie spendendo il minimo possibile.

Grafo

Grafo direzionale

Il problema può essere rappresentato usando un **grafo**. Si dice **grafo direzionale** una coppia di insiemi (N, A)

- ▶ N : insieme dei nodi
- ▶ $A \subseteq N \times N$: sottoinsieme delle coppie ordinate dei nodi (rappresenta una relazione tra nodi)
- ▶ $(i, j) \in A$ è detto **arco**
- ▶ i e j sono gli estremi dell'arco: i è la **coda** e j è la **testa**. L'arco (i, j) è incidente in i e in j : è un arco uscente da i ed entrante in j . Se esiste l'arco (i, j) allora i e j sono adiacenti.

Grafo

Rappresentazione dell'oleodotto come grafo

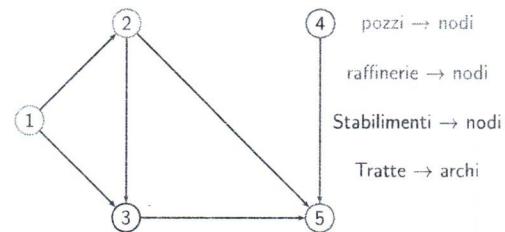


Figura: Oleodotto come grafo

Problema dell'oleodotto - modello

Variabili

Quantità di greggio che passa nella tratta (i, j) : $x_{ij} \geq 0$
Flusso sull'arco (i, j)

Funzione obiettivo

$$\min \sum_{(i,j) \in A} c_{ij} x_{ij}$$

Vincoli di capacità

- ▶ Limite di capacità sulle tratte: $x_{ij} \leq u_{ij}, \forall (i, j) \in A$

Problema dell'oleodotto - modello

Vincoli di bilanciamento

- ▶ Bilanciamento per i pozzi petroliferi ($i \in P$): la differenza tra flusso uscente e flusso entrante, ovvero quanto prodotto da i , deve essere al più pari alla disponibilità
 $\sum_{(i,j) \in A} x_{ij} - \sum_{(j,i) \in A} x_{ji} \leq d_i, \forall i \in P$
- ▶ Bilanciamento per le raffinerie ($i \in R$):
 $\sum_{(j,i) \in A} x_{ji} - \sum_{(i,j) \in A} x_{ij} \geq r_i, \forall i \in R$
- ▶ Bilanciamento per i nodi di smistamento ($i \in N \setminus \{P, R\}$):
 $\sum_{(j,i) \in A} x_{ji} - \sum_{(i,j) \in A} x_{ij} = 0, \forall i \in N \setminus \{P, R\}$

Flusso di costo minimo

Flusso di costo minimo

Problema generale che rappresenta molti casi reali.
Dati:

- ▶ Un grafo orientato $G = (N, A)$
- ▶ Due parametri per ogni arco $(i, j) \in A$:
 - ▶ la capacità u_{ij}
 - ▶ il costo c_{ij}
- ▶ Il bilancio b_i di ogni nodo $i \in N$:

$$b_i = \begin{cases} > 0, & \text{nodo sorgente del flusso} \\ < 0, & \text{nodo che richiede flusso (nodo pozzo)} \\ = 0, & \text{nodo di transito} \end{cases}$$

Flusso di costo minimo - modello

Modello

$$\min \sum_{(i,j) \in A} c_{ij} x_{ij}$$

s.t.

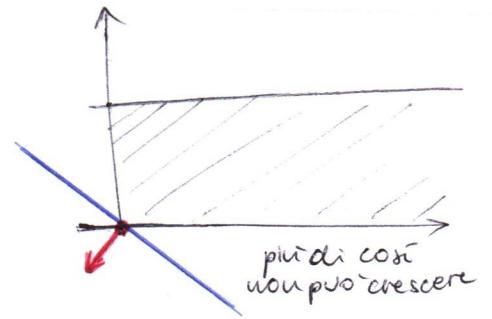
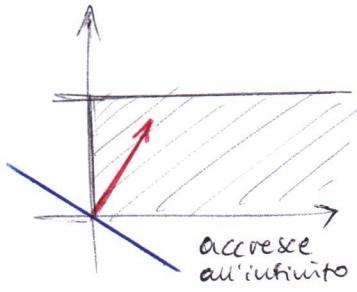
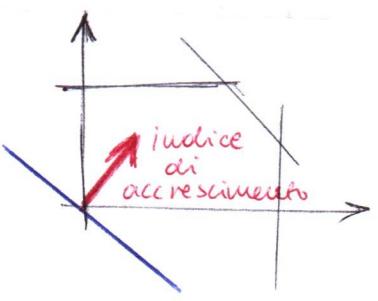
$$x_{ij} \leq u_{ij}, \quad \forall (i, j) \in A$$

Vincoli di capacità

$$\sum_{(i,j) \in A} x_{ij} - \sum_{(j,i) \in A} x_{ji} = b_i, \quad \forall i \in N$$

Vincoli di bilanciamento del flusso ai nodi

$$x_{ij} \geq 0$$



Argomento del corso

- Modellazione in Programmazione Lineare
- Proprietà della Programmazione Lineare (a variabili continue) e metodi di soluzione
 - Forme alternative
 - Geometria della Programmazione Lineare (PL)
 - Proprietà della soluzione ottima
 - Il metodo del simplex
- Una classe particolare di problemi di Ottimizzazione: i problemi su grafo
- Metodi per la Programmazione Lineare Intera

3. Programmazione Lineare (2)

Fondamenti di Ricerca Operativa - A.A. 2018/2019

Giuliana Carello
DEIB, Politecnico di Milano

Forma generale e forma standard

Forme

Forma generale	Forma standard
$\max c^T x$	$\min c^T x$
$Ax \leq b$	$Ax = b$
	$x \geq 0$
	(con $b \geq 0$)

Trasformazioni

Funzione obiettivo

- $\min f(x) \rightarrow -\max (-f(x))$
- $\max f(x) \rightarrow -\min (-f(x))$

Trasformazioni

Trasformazioni

Variabili (standard)

- $x \leq 0 \rightarrow y = -x, y \geq 0$
- x libera $\rightarrow x = x^+ - x^-, x^+, x^- \geq 0$

Variabili (generale)

L'eventuale dominio delle variabili diventa un vincolo

Vincoli (forma generale)

- $Ax \geq b \rightarrow -Ax \leq -b$
- $Ax = b \rightarrow \begin{cases} Ax \leq b \\ Ax \geq b \end{cases} \rightarrow -Ax \leq -b$

Trasformazioni

Esempio

Vincoli (forma standard)

- $Ax \leq b$:
 - si inserisce una variabile non negativa x_s - *variabile di scarto (slack)*
 - $\rightarrow Ax + x_s = b$
- $Ax \geq b$:
 - si inserisce una variabile non negativa x_s - *variabile di scarto (surplus)*
 - $\rightarrow Ax - x_s = b$

Coltivatore: forma standard

$$\begin{array}{ll} \max 3x_L + 5x_P & \rightarrow -\min -3x_L - 5x_P \\ x_L + x_P & \leq 12 \quad \rightarrow x_L + x_P + x_1 = 12 \\ x_L + 2x_P & \leq 16 \quad \rightarrow x_L + 2x_P + x_2 = 16 \\ x_L & \leq 10 \quad \rightarrow x_L + x_3 = 10 \\ x_P & \leq 6 \quad \rightarrow x_P + x_4 = 6 \\ x_P, x_L & \geq 0 \end{array}$$

Esempio

Coltivatore: forma generale

$$\begin{aligned} \max & 3x_L + 5x_P \\ \text{s.t.} & x_L + x_P \leq 12 \\ & x_L + 2x_P \leq 16 \\ & x_L \leq 10 \\ & x_P \leq 18 \\ & x_P, x_L \geq 0 \quad \rightarrow -x_L \leq 0, -x_P \leq 0 \end{aligned}$$

Rappresentazione grafica

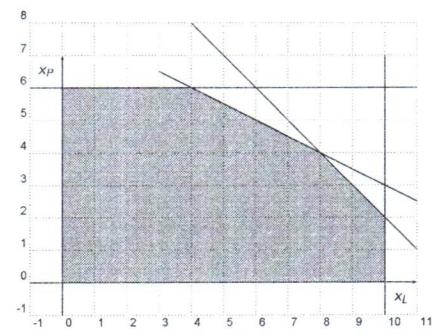


Figura: Regione ammissibile

Esempi di regione ammissibile e soluzioni ottime

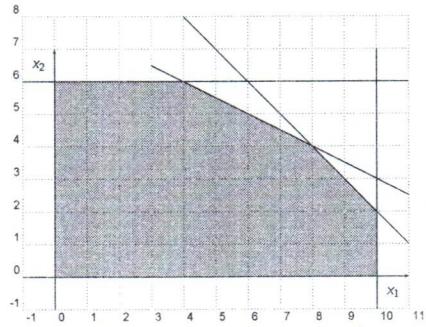


Figura: Regione ammissibile

Esempi di regione ammissibile e soluzioni ottime

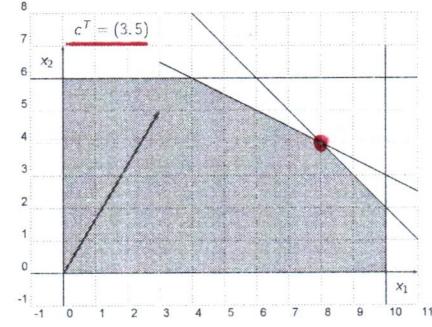


Figura: Singola soluzione ottima: (8,4)

Esempi di regione ammissibile e soluzioni ottime

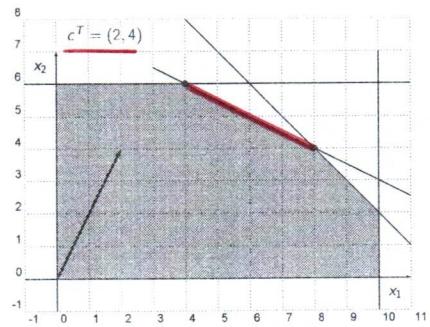


Figura: Infinite soluzioni ottime: segmento che unisce (4,6) a (8,4)

Esempi di regione ammissibile e soluzioni ottime

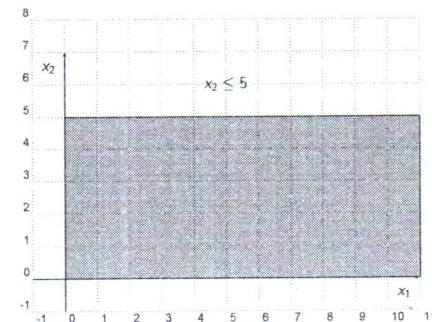


Figura: Regione illimitata

Esempi di regione ammissibile e soluzioni ottime

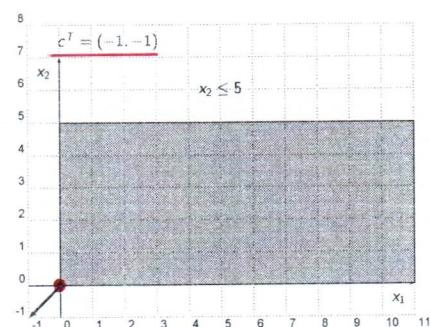


Figura: Regione illimitata: valore ottimo finito (0,0)

Esempi di regione ammissibile e soluzioni ottime

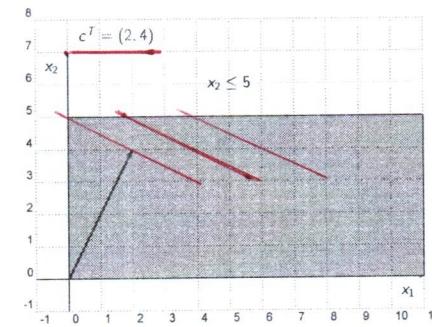


Figura: Regione illimitata: valore ottimo infinito

Esempi di regione ammissibile e soluzioni ottime

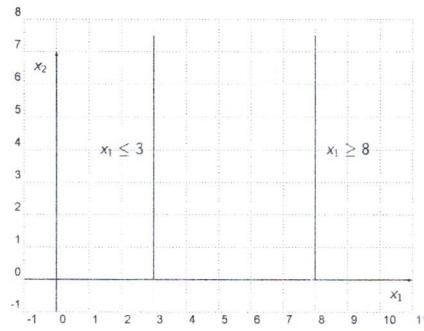


Figura: Regione ammissibile vuota

Definizioni

Def. Involucro convesso

L'involucro convesso di un insieme K ($\text{conv}(K)$) è l'insieme di tutte le combinazioni convesse degli elementi di K

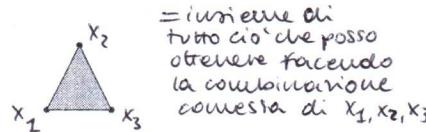


Figura: Combinazione convessa

= più piccolo insieme convesso che contiene tutti e tre i punti

Definizioni

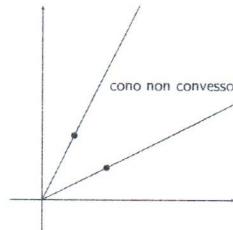
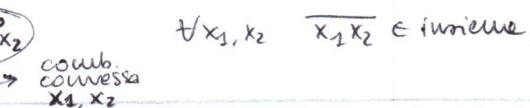


Figura: Esempi di cono



Definizioni

Def. Cono

Un insieme $K \subset \mathbb{R}^n$ si dice cono se $\forall x \in K$ e $\forall \lambda \geq 0$ si ha che $\lambda x \in K$

Def. Combinazione conica

Un punto $x \in \mathbb{R}^n$ si dice combinazione conica di $x^1, \dots, x^m \in \mathbb{R}^n$ se esistono dei coefficienti $\lambda_1, \dots, \lambda_m$ tali che

$$x = \sum_{i=1}^m \lambda_i x^i, \quad \lambda_i \geq 0, \quad \forall i$$

Def. Involucro conico

L'involucro conico di un insieme K ($\text{cono}(K)$) è l'insieme di tutte le combinazioni coniche degli elementi di K

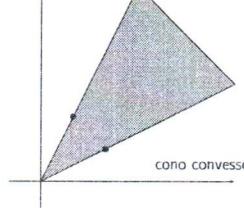
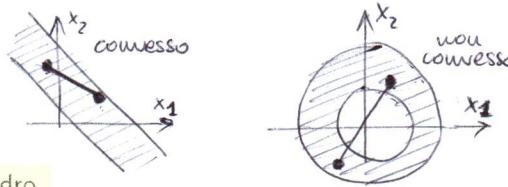


Figura: Esempi di cono

Definizioni



Definizioni

Def. Poliedro

Un poliedro di \mathbb{R}^n è l'intersezione di un numero finito di semispazi chiusi di \mathbb{R}^n .

Osservazione

Ogni poliedro può essere visto come l'insieme delle soluzioni di un sistema di m disequazioni in n incognite:

$$P = \{x \in \mathbb{R}^n : Ax \leq b\}$$

Regione ammissibile

- regione convessa
- intersezione di semispazi chiusi

La regione ammissibile di un problema in Programmazione Lineare è un poliedro.

E ancora vero se un vincolo è descritto da un'espressione non lineare?

Poliedro e politopo

- **Poliedro**: intersezione di un numero finito di semispazi chiusi
- **Politopo**: poliedro limitato

Coni e poliedri

Non tutti i coni sono poliedri.

Esempio $\{x \in \mathbb{R}^3 : x_3 \geq \sqrt{x_1^2 + x_2^2}\}$ (cono gelato)

Proposizione

P è un cono poliedrico se e solo se $\exists Q$:

$$P = \{x \in \mathbb{R}^n : Qx \leq 0\}$$

Esempio: vertici

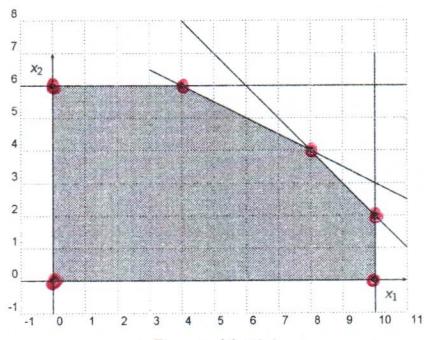


Figura: Vertici

Rappresentazione dei poliedri

Teo: Rappresentazione dei poliedri

Dato un poliedro P , esistono un sottoinsieme finito di punti $V = \{v^1, \dots, v^m\}$ di P ed un insieme finito di direzioni $E = \{e^1, \dots, e^p\}$, eventualmente anche vuoti, tali che

$$P = \text{conv}(V) + \text{cone}(E).$$

Definizioni

Def. Vertice

Un vertice di un poliedro è un punto del poliedro che non si può esprimere come combinazione convessa propria di altri punti del poliedro.

Def. Direzione di recessione

Una direzione di recessione per un poliedro P è un vettore d tale che

$$x + \lambda d \in P \quad \forall x \in P, \quad \forall \lambda \geq 0.$$

Esempio: direzioni di recessione

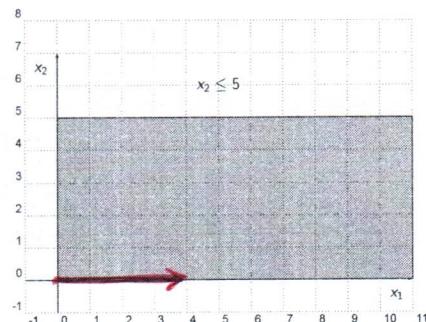


Figura: Direzione di recessione

Rappresentazione dei poliedri

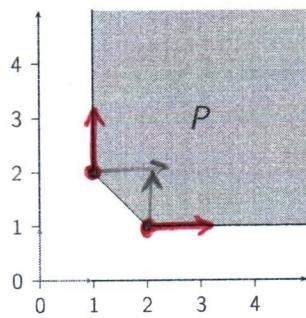
Osservazione

Un poliedro può sempre essere scomposto nella somma di

- un politopo, combinazione convessa (involucro convesso) dei vertici del poliedro
- un cono, combinazione conica di un insieme finito di direzioni, le direzioni di recessione

N.B. Sia l'uno che l'altro possono essere vuoti

Rappresentazione dei poliedri: esempio



$$P = \text{conv}((1,2), (2,1)) + \text{cone}((1,0), (0,1))$$



Condizioni di ottimalità

Teo: Caratterizzazione elementare dell'ottimalità

Dato il problema

$$(P) \quad \begin{cases} \max c^T x \\ x \in P = \{x \in \mathbb{R}^n : Ax \leq b\} \end{cases}$$

- Se $c \neq 0$, allora ogni soluzione ottima di (P) non è interna al poliedro P .
- Se il problema (P) ha due soluzioni ottime, allora ne ha infinite.
- Le soluzioni ottime locali di (P) sono anche ottime globali.

Teorema fondamentale della PL

Teo: Fondamentale della PL

Supponiamo che il poliedro P sia rappresentato come

$$P = \text{conv}\{v^1, \dots, v^m\} + \text{cono}\{e^1, \dots, e^p\}.$$

Se il problema (\mathcal{P}) ha valore ottimo finito, allora esiste $k \in \{1, \dots, m\}$ tale che v^k è una soluzione ottima di (\mathcal{P}) .

Teorema fondamentale della PL

Dimostrazione

(\mathcal{P}) è equivalente a:

$$\begin{aligned} & \max \sum_{i=1}^m \lambda_i c^T v^i + \sum_{j=1}^p \mu_j c^T e^j \\ & \sum_{i=1}^m \lambda_i = 1 \\ & \lambda \geq 0 \\ & \mu \geq 0. \end{aligned}$$

Teorema fondamentale della PL

Dimostrazione

(\mathcal{P}) ha ottimo finito \Rightarrow

$$c^T e^j \leq 0, \quad \forall j = 1, \dots, p,$$

altrimenti, se $c^T e^j > 0$

$\mu_j \rightarrow +\infty \Rightarrow$ funzione obiettivo tenderebbe a $+\infty$.

Teorema fondamentale della PL

Dimostrazione

Sia $v^k = \arg \max_{1 \leq i \leq m} c^T v^i$

$\forall x \in P$ otteniamo:

$$\begin{aligned} c^T x &= \sum_{i=1}^m \lambda_i c^T v^i + \sum_{j=1}^p \mu_j c^T e^j \\ &\leq \sum_{i=1}^m \lambda_i c^T v^i \leq \sum_{i=1}^m \lambda_i \max_{1 \leq i \leq m} c^T v^i \\ &= c^T v^k \sum_{i=1}^m \lambda_i \\ &= c^T v^k \end{aligned}$$

Teorema fondamentale della PL

Dimostrazione

Quindi $\max_{x \in P} c^T x \leq c^T v^k$.

$v^k \in P \Rightarrow$

$$c^T v^k \leq \max_{x \in P} c^T x$$

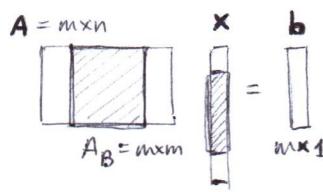
$\Rightarrow v^k$ è una soluzione ottima di (\mathcal{P}) . ■

Idea del metodo del simplex (Dantzig 1947)

- ▶ La soluzione ottima, se è finita, si trova su un vertice
- ▶ La ricerca si può limitare ai vertici della regione ammissibile
- ▶ L'idea del simplex è spostarsi da un vertice ad un altro migliorando via via la funzione obiettivo

Formalizziamo il simplex a partire da un problema scritto in forma standard:

- ▶ introduciamo il concetto di soluzione di base
- ▶ mostriamo relazione tra soluzioni di base e vertici



Soluzione di base

Soluzioni di base

Consideriamo un problema di programmazione lineare scritto in forma standard:

$$\min c^T x$$

$$Ax = b$$

$$x \geq 0$$

con A matrice $m \times n$, con $n > m$, di rango massimo.

Dalla matrice A è possibile estrarre una sottomatrice quadrata $m \times m$ di rango massimo, $A_B : A = (A_B A_N)$

Soluzioni di base

Soluzioni di base

Il sistema dei vincoli può essere così riscritto:

$$A_B x_B + A_N x_N = b, x_B, x_N \geq 0$$

dove

A_B è la sottomatrice quadrata estratta. Le sue colonne corrispondono a un sottoinsieme delle variabili;

x_B è l'insieme delle variabili associate alle colonne di A_B ;

A_N è la sottomatrice composta dalle colonne che non appartengono a A_B ;

x_N è l'insieme delle variabili associate alle colonne di A_N ;

c_B sono i costi relativi alle variabili associate ad A_B ;

c_N sono i costi relativi alle variabili associate ad A_N .

Soluzioni di base

È possibile calcolare una soluzione del problema imponendo $x_N = 0$ e $x_B = A_B^{-1}b$. arrevo le variabili fuori base

Def. Soluzione di base

Data una sottomatrice A_B della matrice dei coefficienti dei vincoli, invertibile di rango massimo, una soluzione del tipo $(A_B^{-1}b, 0)$ è una soluzione di base, la matrice A_B è una matrice di base, l'insieme degli indici delle variabili associate alla matrice di base è descritto dall'insieme B , la base.

Def. Soluzione ammissibile di base

Una soluzione di base con $x_B \geq 0$ è una soluzione ammissibile di base.

Le variabili associate a A_B sono chiamate variabili in base, quelle associate a A_N fuori base.

Relazione tra soluzioni di base e vertici

Possibili basi

$m = 4, n = 6$: le basi sono composte da 4 variabili
Es:

$$(x_L, x_P, x_3, x_4) = A_B^{-1}b = (8, 4, 2, 2)$$

$$AAA_B A_N = \begin{pmatrix} x_L & x_P & x_1 & x_2 & x_3 & x_4 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 2 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Rappresentazione grafica

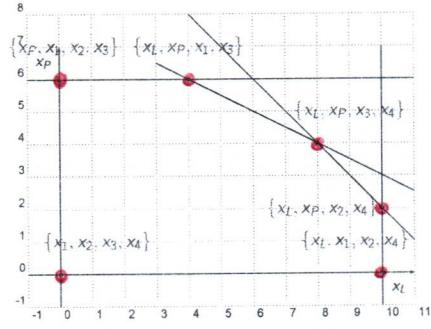


Figura: Regione ammissibile

Rappresentazione grafica

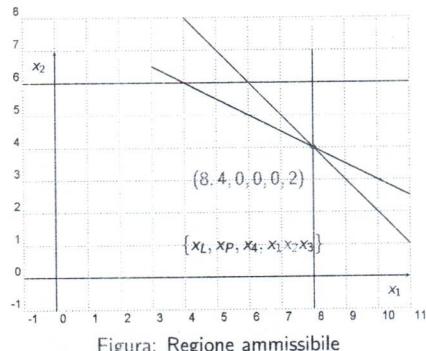


Figura: Regione ammissibile

Coltivatore: forma standard

$$\begin{array}{ll} \max 3x_L + 5x_P & \rightarrow -\min -3x_L - 5x_P \\ x_L + x_P \leq 12 & \rightarrow x_L + x_P + x_1 = 12 \\ x_L + 2x_P \leq 16 & \rightarrow x_L + 2x_P + x_2 = 16 \\ x_L \leq 10 & \rightarrow x_L + x_3 = 10 \\ x_P \leq 6 & \rightarrow x_P + x_4 = 6 \\ x_P, x_L \geq 0 & x_L, x_P, x_1, x_2, x_3, x_4 \geq 0 \end{array}$$

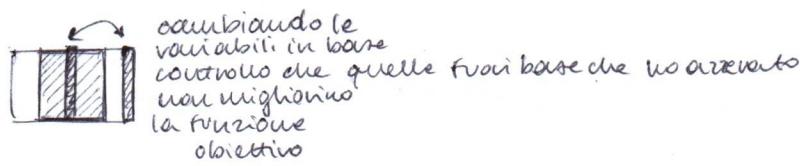
Relazione tra soluzioni di base e vertici

Possibili basi

$m = 4, n = 6$: le basi sono composte da 4 variabili
Es:

$$(x_L, x_P, x_3, x_4) = A_B^{-1}b = (8, 4, 2, 2)$$

$$A = [A_B A_N] = \begin{pmatrix} x_L & x_P & x_1 & x_2 & x_3 & x_4 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 2 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$



Esempio:

Modello

$$\begin{array}{ll} x_L + x_P \leq 12 & \rightarrow x_L + x_P + x_1 = 12 \\ x_L + 2x_P \leq 16 & \rightarrow x_L + 2x_P + x_2 = 16 \\ x_L \leq 8 & \rightarrow x_L + x_3 = 8 \\ x_P \leq 6 & \rightarrow x_P + x_4 = 6 \\ x_P, x_L \geq 0 & x_L, x_P, x_1, x_2, x_3, x_4 \geq 0 \end{array}$$

Soluzioni degeneri

Soluzione degenera

Consideriamo una soluzione di questa forma:

$$x_1, \dots, x_i, \dots, x_m, \dots, x_n$$

dove le prime m variabili sono in base, e supponiamo che la variabile x_i , con $i \in B$, sia nulla.

In questo caso alla soluzione è associata più di una base: le diverse basi si possono ottenere scambiando la colonna i con una colonna fuori base (a patto di ottenere con lo scambio una matrice invertibile).

Soluzioni degeneri

Def: soluzione di base degenera

Una soluzione di base in cui una variabile in base (o più di una) assume valore nullo è detta **soluzione degenera**.

Relazione tra soluzioni di base e vertici

Teorema

Consideriamo una matrice $m \times n$ di rango massimo (con $m < n$) A e la regione ammissibile P descritta dai vincoli

$$\begin{aligned} Ax &= b \\ x &\geq 0 \end{aligned} \quad (1)$$

Un punto x è un vertice di $P \iff$ è una soluzione di base per (1).

Relazione tra soluzioni di base e vertici

Dimostrazione

\Leftarrow Se x è una soluzione di base allora

$$x = (x_1, x_2, x_3, \dots, x_m, 0, \dots, 0)$$

e

$$x_1 \mathbf{a}_1 + x_2 \mathbf{a}_2 + \dots + x_m \mathbf{a}_m = b,$$

dove \mathbf{a}_i è l' i -esima colonna di A e le colonne $\mathbf{a}_1, \dots, \mathbf{a}_m$ sono linearmente indipendenti.

Se per assurdo x non è un vertice allora esistono $y, z \in P, y \neq z$ e $0 < \lambda < 1$ tali che

$$x = \lambda y + (1 - \lambda)z.$$

Relazione tra soluzioni di base e vertici

Dimostrazione

Poiché $x, y, z \geq 0$ allora le ultime $n - m$ componenti di y e z devono essere nulle e si ha

$$\begin{aligned} y_1 \mathbf{a}_1 + y_2 \mathbf{a}_2 + \dots + y_m \mathbf{a}_m &= b \\ z_1 \mathbf{a}_1 + z_2 \mathbf{a}_2 + \dots + z_m \mathbf{a}_m &= b \end{aligned}$$

Sottraendo le due espressioni si ottiene una combinazione lineare **nulla** delle colonne di A a coefficienti non tutti nulli, contraddicendo l'ipotesi che x sia una soluzione di base.

Relazione tra soluzioni di base e vertici

Dimostrazione

\Rightarrow Se x è un vertice di P , allora supponiamo che le prime k componenti di x siano non nulle. Allora si ha

$$x_1 \mathbf{a}_1 + x_2 \mathbf{a}_2 + \dots + x_k \mathbf{a}_k = b$$

Dimostriamo per assurdo che $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_k$ sono linearmente indipendenti e quindi x è una base.

Relazione tra soluzioni di base e vertici

Dimostrazione

Se $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_k$ non sono linearmente indipendenti (per assurdo) allora $\exists y_1, y_2, \dots, y_k$ non tutti nulli tale che

$$y_1 \mathbf{a}_1 + y_2 \mathbf{a}_2 + \dots + y_k \mathbf{a}_k = 0.$$

Definiamo il vettore di n elementi $y = (y_1, y_2, \dots, y_k, 0, \dots, 0)$. Possiamo trovare ϵ tale che:

$$\begin{aligned} x + \epsilon y &\geq 0, \quad x - \epsilon y \geq 0 \\ A(x + \epsilon y) &= b, \quad A(x - \epsilon y) = b \\ x + \epsilon y \in P, \quad x - \epsilon y \in P \end{aligned}$$

Relazione tra soluzioni di base e vertici

Dimostrazione

Ma allora

$$x = \frac{1}{2}(x + \epsilon y) + \frac{1}{2}(x - \epsilon y)$$

e x non è un vertice.

Relazione tra soluzioni di base e vertici

Implicazioni del Teorema fondamentale

Per il Teorema Fondamentale della Programmazione Lineare, è possibile limitare la ricerca della soluzione ottima del problema ai vertici del poliedro, quindi alle soluzioni di base.

teorema principale PL + teorema soluzione base e vertici

Il metodo del simplex

Idea generale

Il metodo del simplex si muove da una soluzione di base ammissibile ad un'altra finché non giunge nella soluzione ottima.

Si muove tra soluzioni di base **adiacenti**, ovvero che differiscono per una sola colonna.

Metodo del simplex

Criteri

Bisogna dunque definire:

- un criterio di ottimalità per riconoscere una soluzione ottima,
- un criterio per stabilire quale variabile (colonna) entra nella base,
- un criterio per stabilire quale variabile (colonna) esce dalla base.

Simplex: un esempio introduttivo

Coltivatore: forma standard

$$\begin{array}{ll} \max 3x_L + 5x_P & \rightarrow -\min -3x_L - 5x_P \\ x_L + x_P \leq 12 & \rightarrow x_L + x_P + x_1 = 12 \\ x_L + 2x_P \leq 16 & \rightarrow x_L + 2x_P + x_2 = 16 \\ x_L \leq 10 & \rightarrow x_L + x_3 = 10 \\ x_P \leq 6 & \rightarrow x_P + x_4 = 6 \\ x_P, x_L \geq 0 & x_L, x_P, x_1, x_2, x_3, x_4 \geq 0 \end{array}$$

Simplex: un esempio introduttivo

È possibile migliorare la soluzione trovata?

x_P ha costo negativo: $x_P > 0$ produce una diminuzione della funzione obiettivo $z = -5x_P$. Di quanto può crescere x_P ? Le variabili in base cambiano valore all'aumentare di x_P :

$$\begin{array}{lll} x_P = 1 & z = -5 & x_1 = 11, x_2 = 14, x_3 = 10, x_4 = 5 \\ x_P = 4 & z = -20 & x_1 = 8, x_2 = 6, x_3 = 10, x_4 = 2 \\ x_P = 7 & z = -35 & x_1 = 5, x_2 = 2, x_3 = 10, x_4 = -1 \end{array}$$

Nel primo e secondo caso la soluzione non è di base, nel terzo non è ammissibile

Simplex: un esempio introduttivo

È possibile migliorare la soluzione trovata?

Occorre:

- 1) scrivere le variabili in base x_P, x_1, x_2, x_3 in funzione di quelle fuori base x_L, x_4 :

$$\begin{array}{ll} x_P &= 6 - x_4 \\ x_1 &= 12 - x_L + x_4 \\ x_2 &= 4 - x_L + 2x_4 \\ x_3 &= 10 - x_L \end{array}$$

Simplex: un esempio introduttivo

È facile individuare una soluzione ammissibile:

$x_L = x_P = 0, x_1 = 12, x_2 = 16, x_3 = 10, x_4 = 6$, dove sono in base le variabili di scarto x_1, x_2, x_3, x_4 .

A questa soluzione corrisponde una funzione obiettivo

$$z = -3x_L - 5x_P = 0$$

Ciascuna delle variabili in base può essere scritta in funzione di quelle fuori base:

$$\begin{array}{ll} x_1 &= 12 - x_L - x_P \\ x_2 &= 16 - x_L - 2x_P \\ x_3 &= 10 - x_L \\ x_4 &= 6 - x_P \end{array}$$

Simplex: un esempio introduttivo

Qual è il massimo valore θ che può raggiungere x_P ?

$$x_P = \theta:$$

$$\begin{array}{ll} x_1 = 12 - \theta \geq 0 & \Rightarrow \theta \leq 12 \\ x_2 = 16 - \theta \geq 0 & \Rightarrow \theta \leq 8 \\ x_3 = 10 \geq 0 & \\ x_4 = 6\theta \geq 0 & \Rightarrow \theta \leq 6 \end{array}$$

Con $x_P = 6$ si ottiene una nuova soluzione di base $x_L = 0, x_P = 6, x_1 = 6, x_2 = 4, x_3 = 10, x_4 = 0$, con valore $z = -30$.

Simplex: un esempio introduttivo

- 2) valutare l'impatto sulla funzione obiettivo delle variabili fuori base:

$$\min -3x_L - 5x_P = -30 - 3x_L + 5x_4 = z$$

$$z = z_0 - 3x_L + 5x_4$$

valore della funzione obiettivo nella soluzione corrente
contributo delle variabili fuori base

Simplesso: un esempio introduttivo

Far entrare in base x_L produce una diminuzione: di quanto può aumentare x_L ? $x_L = \theta$:

$$\begin{aligned}x_P &= 6 - x_4 \geq 0 \\x_1 &= 6 - \theta \geq 0 \Rightarrow \theta \leq 6 \\x_2 &= 4 - \theta \geq 0 \Rightarrow \theta \leq 4 \\x_3 &= 10 - \theta \geq 0 \Rightarrow \theta \leq 10\end{aligned}$$

Nuova soluzione:

$x_L = 4$, $x_P = 6$, $x_1 = 2$, $x_2 = 0$, $x_3 = 6$, $x_4 = 0$, con valore $z = -42$

Il metodo del simplesso

È possibile scrivere le variabili in base in funzione di quelle fuori base:

$$Ax = A_B x_B + A_N x_N = b \Rightarrow x_B = A_B^{-1}b - A_B^{-1}A_N x_N.$$

La funzione obiettivo può essere riscritta:

$$\begin{aligned}z &= c^T x = c_B^T x_B + c_N^T x_N = c_B^T (A_B^{-1}b - A_B^{-1}A_N x_N) + c_N^T x_N \\&= c_B^T A_B^{-1}b + (c_N^T - c_B^T A_B^{-1}A_N)x_N \\Z &= z_0 + r_N^T x_N.\end{aligned}$$

dove z_0 è il valore della funzione obiettivo associato alla soluzione di base attuale.

Il metodo del simplesso

Nella prima iterazione dell'esempio del coltivatore: Es:

$$(x_1, x_2, x_3, x_4) = A_B^{-1}b = (12, 16, 10, 6)$$

$$A = [A_B \ A_N] = \left(\begin{array}{cccc|cc} x_L & x_P & x_1 & x_2 & x_3 & x_4 \\ \hline 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 2 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{array} \right)$$

I costi delle variabili in base sono tutti nulli:

$$c_N^T - c_B^T A_B^{-1}A_N = c_N^T$$

Nella seconda iterazione dell'esempio del coltivatore: Es:

$$(x_P, x_1, x_2, x_3) = A_B^{-1}b = (6, 6, 4, 10)$$

$$A = [A_B \ A_N] = \left(\begin{array}{cccc|cc} x_L & x_P & x_1 & x_2 & x_3 & x_4 \\ \hline 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 2 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{array} \right)$$

$$c_N^T - c_B^T A_B^{-1}A_N = c_N^T = (-3 \ 0)^T$$

Costi ridotti

Def: costi ridotti

Definiamo vettore dei costi ridotti il vettore:

$$r_N^T = c_N^T - c_B^T A_B^{-1}A_N.$$

I costi ridotti descrivono l'impatto sulla funzione obiettivo dell'aumento del valore delle variabili fuori base.

Se esiste una variabile x_i , con $i \notin B$, per cui $r_i < 0$, aumentando il valore della variabile x_i , cioè inserendola tra le variabili di base, il valore della funzione obiettivo diminuisce.

Scelta della variabile che entra in base

Casi

- Se i costi ridotti delle variabili fuori base sono tutti non negativi, la soluzione di base considerata è ottima
- Se esiste una variabile x_i , con $i \notin B$, con costo ridotto r_i negativo, la base B non è ottima.

La base viene modificata sostituendo una variabile attualmente in base con una attualmente fuori base.

Scelta della variabile che entra in base

Scelta della variabile che entra in base

Entra in base una delle variabili fuori base con costo ridotto negativo.

Per rimanere in una soluzione di base, se una variabile entra a far parte della soluzione, un'altra deve uscire.

Il numero di variabili in base deve essere m .

$$x_B = A_B^{-1}b - A_B^{-1}A_N x_N$$

Al crescere del valore della variabile x_i , con $i \notin B$ e $r_i < 0$, il valore delle variabili in base può decrescere.

La soluzione deve però rimanere ammissibile, ovvero $x_B \geq 0$. Dunque

$$x_j = (A_B^{-1}b - A_B^{-1}A_N x_N)_j = (A_B^{-1}b)_j - (A_B^{-1}A_N x_N)_j \geq 0, \forall j \in B.$$

Scelta della variabile che esce dalla base

Supponiamo che la variabile x_i , con $i \notin B$ assuma il valore $\theta > 0$.

Il nuovo valore della variabile in base x_j è

$$x_j = (A_B^{-1}b)_j - (A_B^{-1}A_N)_{ji}\theta.$$

Per garantire l'ammissibilità della nuova soluzione bisogna imporre, per ogni $j \in B$:

$$x_j = (A_B^{-1}b)_j - (A_B^{-1}A_N)_{ji}\theta \geq 0$$

Scelta della variabile che esce dalla base

Scelta della variabile che esce dalla base

Considerando la variabile che entra in base x_i e una variabile attualmente in base x_j , si possono verificare due casi:

- a) $(A_B^{-1}A_N)_{ji} > 0$: θ deve essere tale da garantire che la nuova soluzione sia ammissibile, ovvero

$$\theta \leq \frac{(A_B^{-1}b)_j}{(A_B^{-1}A_N)_{ji}}$$

- b) $(A_B^{-1}A_N)_{ji} \leq 0$: anche se x_i cresce x_j non decresce, nessun limite al valore di θ è necessario per garantire l'ammissibilità di x_j

Scelta della variabile che esce dalla base

Scelta della variabile che esce dalla base

- $\forall j \in B$, $(A_B^{-1}A_N)_{ji} \leq 0$
non vi è alcun limite al valore di θ :
la variabile x_i può assumere un valore grande a piacere, la funzione obiettivo può decrescere illimitatamente.
L'ottimo è illimitato.

- la variabile che esce dalla base è la variabile j^* tale che

$$j^* = \arg \min_{\substack{j \in B \\ (A_B^{-1}A_N)_{ji} > 0}} \left\{ \frac{(A_B^{-1}b)_j}{(A_B^{-1}A_N)_{ji}} \right\}.$$

La nuova base è data da $B = B \cup \{i\} \setminus \{j^*\}$.

Schema del metodo del simplexso

Schema del metodo del simplexso

1. Si calcola il valore delle variabili in base $x_B = A_B^{-1}b$, il valore della funzione obiettivo $z = c_B A_B^{-1}b$ e il vettore dei costi ridotti $r_N^T = c_N^T - c_B^T A_B^{-1}A_N$
2. Se i costi ridotti sono tutti positivi la soluzione è ottima e il simplexso termina
3. Altrimenti si seleziona una variabile fuori base x_i con costo ridotto negativo
4. Se $(A_B^{-1}A_N)_{ji} \leq 0 \forall j \in B$ l'ottimo è illimitato e il simplexso termina
5. Altrimenti si seleziona la variabile j^* in base:

$$j^* = \arg \min_{\substack{j \in B \\ (A_B^{-1}A_N)_{ji} > 0}} \left\{ \frac{(A_B^{-1}b)_j}{(A_B^{-1}A_N)_{ji}} \right\}$$
6. Si aggiorna la base sostituendo x_{j^*} con x_i e si torna al passo 1

Il metodo del simplexso: esempio

Coltivatore

$$\begin{array}{lcl}
 \max 3x_L + 5x_P & \rightarrow - \min -3x_L - 5x_P \\
 x_L + x_P & \leq & 12 \rightarrow x_L + x_P + x_1 = 12 \\
 x_L + 2x_P & \leq & 16 \rightarrow x_L + 2x_P + x_2 = 16 \\
 x_L & \leq & 10 \rightarrow x_L + x_3 = 10 \\
 x_P & \leq & 6 \rightarrow x_P + x_4 = 6 \\
 x_P, x_L & \geq & 0 \quad x_L, x_P, x_1, x_2, x_3, x_4 \geq 0
 \end{array}$$

Esempio

$m = 4, n = 6$: le basi sono composte da 4 variabili
Consideriamo la base data dalle variabili di scarto: $A_B = I$.
A una sottomatrice identità è sempre associata una base ammissibile: $x_B = Ib = b \geq 0$
Base iniziale:

$$\{x_1, x_2, x_3, x_4\} = A_B^{-1}b^T = (12, 16, 10, 6), f.o.b.(z) = 0$$

$$A = [A_B A_N] = \begin{pmatrix} x_L & x_P & x_1 & x_2 & x_3 & x_4 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 2 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Esempio

$$A_B = I, c_B^T = (0 \ 0 \ 0 \ 0), c_N^T = (-3 \ -5)$$

$$r_N^T = c_N^T - c_B^T A_B^{-1} A_N = c_N^T = (-3 \ -5).$$

La base non è ottima: scegliamo la variabile con costo ridotto più negativo x_P .

Esempio

Calcoliamo i rapporti relativi alla variabile entrante x_P ($q = 2$):

$$\begin{array}{ll} i = 1 (x_1) & \frac{(A_B^{-1} b)_1}{(A_B^{-1} A_N)_{12}} = \frac{12}{1} \\ i = 2 (x_2) & \frac{(A_B^{-1} b)_2}{(A_B^{-1} A_N)_{22}} = \frac{16}{2} \\ i = 3 (x_3) & (A_B^{-1} A_N)_{32} \leq 0 \\ i = 4 (x_4) & \frac{(A_B^{-1} b)_4}{(A_B^{-1} A_N)_{42}} = \frac{6}{1} \end{array} \quad \left| \begin{array}{l} x_L + x_P + x_1 = 12 \\ x_L + 2x_P + x_2 = 16 \\ x_L + x_4 = 6 \end{array} \right.$$

esce dalla base x_4 .

Esempio

Nuova base:

$$\{x_P, x_1, x_2, x_3\}$$

$$A = [A_B A_N] = \begin{pmatrix} x_L & x_P & x_1 & x_2 & x_3 & x_4 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 2 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Esempio

$$A_B^{-1} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -2 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Nuova base:

$$\{x_P, x_1, x_2, x_3\} = A_B^{-1} b = (6, 6, 4, 10), z = -30$$

$$c_B^T = (-5, 0, 0, 0), c_N^T = (-3, 0)$$

Esempio

$$A_B^{-1} A_N = \begin{pmatrix} 0 & 1 \\ 1 & -1 \\ 1 & -2 \\ 1 & 0 \end{pmatrix}$$

$$r_N^T = c_N^T - c_B^T A_B^{-1} A_N = (-3, 5)$$

entra in base x_L .

Esempio

Calcoliamo i rapporti relativi alla variabile entrante x_L ($q = 1$):

$$\begin{array}{ll} i = 1 (x_P) & (A_B^{-1} A_N)_{11} \leq 0 \\ i = 2 (x_1) & \frac{(A_B^{-1} b)_2}{(A_B^{-1} A_N)_{21}} = \frac{6}{1} \\ i = 3 (x_2) & \frac{(A_B^{-1} b)_3}{(A_B^{-1} A_N)_{31}} = \frac{4}{1} \\ i = 4 (x_3) & \frac{(A_B^{-1} b)_4}{(A_B^{-1} A_N)_{41}} = \frac{10}{1} \end{array}$$

esce dalla base x_2 .

Esempio

Nuova base:

$$\{x_P, x_L, x_1, x_3\}$$

$$A = [A_B A_N] = \begin{pmatrix} x_L & x_P & x_1 & x_2 & x_3 & x_4 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 2 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Esempio

$$A_B^{-1} = \begin{pmatrix} 0 & 1 & 0 & -2 \\ 0 & 0 & 0 & 1 \\ 1 & -1 & 0 & 1 \\ 0 & -1 & 1 & 2 \end{pmatrix}$$

Nuova base:

$$\{x_L, x_P, x_1, x_3\} = A_B^{-1} b = (4, 6, 2, 6), z = -42$$

Esempio

$$A_B^{-1} A_N = \begin{pmatrix} x_2 & x_4 \\ 0 & -2 \\ 0 & 1 \\ -1 & 1 \\ -1 & 2 \end{pmatrix}$$

$$r_N^T = c_N^T - c_B^T A_B^{-1} A_N = (3, -1)$$

entra in base x_4 .

Esempio

Calcoliamo i rapporti relativi alla variabile entrante x_4 ($q = 2$):

$$\begin{aligned} i = 1 (x_L) \quad & (A_B^{-1} A_N)_{12} \leq 0 \\ i = 2 (x_P) \quad & \frac{(A_B^{-1} b)_2}{(A_B^{-1} A_N)_{22}} = \frac{6}{1} \\ i = 3 (x_1) \quad & \frac{(A_B^{-1} b)_3}{(A_B^{-1} A_N)_{32}} = \frac{2}{1} \\ i = 4 (x_3) \quad & \frac{(A_B^{-1} b)_4}{(A_B^{-1} A_N)_{42}} = \frac{6}{2} \end{aligned}$$

esce dalla base x_1 .

Esempio

$$A_B^{-1} = \begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -2 & 1 & 1 & 0 \\ 1 & -1 & 0 & 1 \end{pmatrix}$$

Nuova base:

$$\{x_L, x_P, x_3, x_4\} = A_B^{-1} b = (8, 4, 2, 2), z = -44$$

$$r_N^T = c_N^T - c_B^T A_B^{-1} A_N = (1, 2)$$

\Rightarrow soluzione ottima

Regole anticiclo

Applicando il simplex, in presenza di soluzioni di base (vertici) degeneri, è possibile spostarsi da una base ad un'altra senza cambiare il valore delle variabili.
Questo avviene quando una variabile in base nulla viene sostituita da una variabile fuori base nulla.
Può accadere che il simplex cicli indefinitamente su un insieme di basi, associate allo stesso vertice degenero, senza cambiare il valore della funzione obiettivo.

Fase I del simplex

Ricerca della soluzione di base ammissibile di partenza

Il simplex richiede una soluzione di base ammissibile di partenza. In caso questa non sia disponibile, si applica la Fase I del simplex.

Problema artificiale

Si introducono m variabili artificiali, v , una per ogni vincolo e si risolve il problema artificiale:

$$\begin{aligned} \min v \\ Ax + v = b \\ x \geq 0 \\ v \geq 0 \end{aligned}$$

Esempio

Nuova base:

$$\{x_P, x_L, x_3, x_4\}$$

$$A = [A_B \ A_N] = \left(\begin{array}{cccccc} x_L & x_P & x_1 & x_2 & x_3 & x_4 \\ \hline 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 2 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{array} \right)$$

Esempio

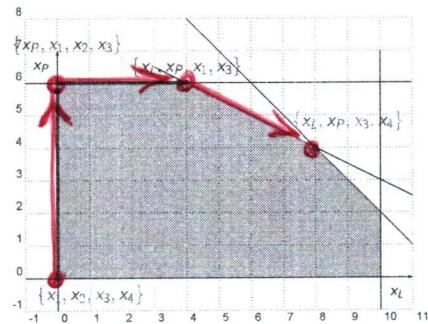


Figura: Regione ammissibile

Regola di Bland

Si può evitare con un'opportuna scelta delle variabili che entrano o escono dalla base.

Tra tutte le variabili candidate a entrare o uscire dalla base si deve scegliere sempre quella con indice minore.

Teorema (Bland 1977)

Il simplex termina se come variabile che entra o esce dalla base viene scelta quella di indice minore

Fase I del simplex

Fase I del simplex

- Se il valore ottimo del problema artificiale è nullo, dalla base ottima del problema artificiale si può ricavare una base ammissibile del problema originario.
- Se il valore ottimo del problema artificiale è positivo, il problema originario non ammette soluzioni ammissibili.

4. Teoria della dualità (5)

Fondamenti di Ricerca Operativa - A.A. 2018/2019

Giuliana Carello
DEIB, Politecnico di Milano

- Modellazione in Programmazione Lineare
- Proprietà della Programmazione Lineare (a variabili continue) e metodi di soluzione
 - Teoria della dualità
 - Relazione tra problema prima e duale e proprietà
 - Scarti complementari (criterio di ottimalità e interpretazione dei costi ridotti)
 - Simplesso duale
- Una classe particolare di problemi di Ottimizzazione: i problemi su grafo
- Metodi per la Programmazione Lineare Intera

Il problema duale: esempio

Consideriamo il problema¹

$$\begin{aligned} \min & 7x_1 + x_2 + 5x_3 \\ x_1 - x_2 + 3x_3 & \geq 10 \\ 5x_1 + 2x_2 - x_3 & \geq 6 \\ x_1, x_2, x_3 & \geq 0 \end{aligned}$$

e cerchiamo di trovare un limite inferiore al valore dell'ottimo confrontando funzione obiettivo e vincoli:

$$7x_1 + x_2 + 5x_3 \geq x_1 - x_2 + 3x_3 \geq 10$$

considero un vincolo per capire ±

¹Esempio tratto da V.Vazirani, Approximation Algorithms, 2003

Dualità: esempio

Possiamo combinare linearmente i vincoli con due moltiplicatori y_1 e y_2 :

$$7x_1 + x_2 + 5x_3 \geq y_1(x_1 - x_2 + 3x_3) + y_2(5x_1 + 2x_2 - x_3) \geq 10y_1 + 6y_2$$

affinché valga la maggiorazione deve essere:

$$\begin{aligned} y_1 + 5y_2 &\leq 7 \\ -y_1 + 2y_2 &\leq 1 \\ 3y_1 - y_2 &\leq 5 \end{aligned}$$

Dualità: esempio

Relazione tra primale e duale

Il problema duale consiste di trovare y_1 e y_2 che forniscono la miglior approssimazione della funzione obiettivo:

$$\begin{aligned} \max & 10y_1 + 6y_2 \\ y_1 + 5y_2 &\leq 7 \\ -y_1 + 2y_2 &\leq 1 \\ 3y_1 - y_2 &\leq 5 \\ y_1, y_2 &\geq 0 \end{aligned}$$

	x_1	x_2	x_3	max
y_1	1	-1	3	≥ 10
y_2	5	2	-1	≥ 6
	\leq	\leq	\leq	
min	7	1	5	

Copie di problemi primale e duale

Copie di problemi primale e duale

Coppia simmetrica

Problema primale	Problema duale
$\min c^T x$	$\max y^T b$
$Ax \geq b$	$y^T A \leq c^T$
$x \geq 0$	$y \geq 0$

Coppia asimmetrica

Problema primale	Problema duale
$\min c^T x$	$\max y^T b$
$Ax = b$	$y^T A \leq c^T$
$x \geq 0$	

Il primale è in forma standard, e il duale in forma generale

Tabella di conversione

Primale	Duale
min	max
variabile	vincolo
vincolo	variabile
costi c	termine noto b
termine noto b	costo c

Tabella di conversione

Primale	Duale
min	max
$A_i x \geq b_i$	$y_i \geq 0$
$A_i x \leq b_i$	$y_i \leq 0$
$A_i x = b_i$	y_i libera
$x_i \geq 0$	$y^T A^i \leq c_i$
$x_i \leq 0$	$y^T A^i \geq c_i$
x_i libera	$y^T A^i = c_i$

Proprietà del duale

Proprietà

Il duale del duale è il primale

Dimostrazione (coppia asimmetrica)

$$P = \min\{c^T x : Ax = b, x \geq 0\}$$

$$D = \max\{y^T b : y^T A \leq c^T\}$$

$$D = \min\{-(y^+ - y^-)^T b : (y^+ - y^-)^T A + \lambda = c^T, y^+, y^- \geq 0\}$$

il duale di D è

$$\max\{c^T w : Aw \leq -b, -Aw \leq b; w \leq 0\}$$

cambiando variabile $x = -w$:

$$\max\{-c^T x : Ax = b, x \geq 0\}$$

$$\min\{c^T x : Ax = b, x \geq 0\}$$

Proprietà del duale

Teo: Dualità debole

Data una coppia asimmetrica di problemi primale e duale:

$$(P) \quad \begin{array}{ll} \min c^T x & \max y^T b \\ Ax = b & y^T A \leq c^T \\ x \geq 0 & \end{array}$$

se (P) e (D) hanno soluzioni ammissibili \bar{x} e \bar{y} , allora

$$c^T \bar{x} \geq \bar{y}^T b$$

Proprietà del duale

Dimostrazione

$$\bar{x}$$
 è ammissibile $\Rightarrow A\bar{x} = b$

$$\text{quindi } \bar{y}^T b = \bar{y}^T A\bar{x}$$

$$\bar{y}$$
 è ammissibile $\Rightarrow \bar{y}^T A \leq c^T$

$$\text{quindi } \bar{y}^T b = \bar{y}^T A\bar{x} \leq c^T \bar{x}$$

Proprietà del duale

Corollario

Se \bar{x} e \bar{y} sono ammissibili per (P) e (D) , rispettivamente, e tali che $c^T \bar{x} = \bar{y}^T b$ allora \bar{x} e \bar{y} sono ottime

Proprietà del duale

Dimostrazione

$\forall x$ ammissibile per (P) si ha

$$c^T x \geq \bar{y}^T A x = \bar{y}^T b = c^T \bar{x} \Rightarrow c^T x \geq c^T \bar{x}$$

$\forall y$ ammissibile per (D) si ha

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

Proprietà del duale

Corollario

Se (P) è illimitato allora (D) non ha soluzioni ammissibili.

Proprietà del duale

		<i>P</i>		
<i>D</i>	finito	sì	illimitato	N.A.
	illimitato	no	no	sì
	N.A.		sì	

Proprietà del duale

Lema di Farkas

Data una matrice A di ordine $m \times n$ ed un vettore $c \in \mathbb{R}^n$, i due sistemi

$$\begin{cases} \eta^T A = c^T \\ \eta \geq 0 \end{cases} \quad (1)$$

$$\begin{cases} A\xi \leq 0 \\ c^T \xi > 0 \end{cases} \quad (2)$$

sono in alternativa, cioè (1) ammette soluzioni se e solo se (2) è impossibile.

Proprietà del duale

Conseguenza del lemma di Farkas

È possibile che il problema primale abbia soluzione ottima finita se il duale non ha soluzioni ammissibili?

$$(P) \quad \begin{array}{l} \max c^T x \\ Ax \leq b \\ x \geq 0 \end{array} \quad (D) \quad \begin{array}{l} \min y^T b \\ y^T A = c^T \\ y \geq 0 \end{array}$$

se (D) non ha soluzione allora $\nexists \eta : \eta^T A = c^T, \eta \geq 0$
per Farkas $\exists \xi : c^T \xi > 0, A\xi \leq 0$

Proprietà del duale

Conseguenza del lemma di Farkas

$$\exists \xi : c^T \xi > 0, A\xi \leq 0$$

allora considero una soluzione \bar{x} ammissibile per (P) e una nuova soluzione ottenuta come $\bar{x} + \xi$

$$c^T(\bar{x} + \xi) = c^T\bar{x} + c^T\xi > c^T\bar{x}$$

$$A(\bar{x} + \xi) = A\bar{x} + A\xi \leq b$$

Possiamo muoverci illimitatamente nella direzione ξ trovando soluzioni sempre migliori e sempre ammissibili:
 ξ è una direzione di recessione di crescita

$\Rightarrow (P)$ è illimitato

Proprietà del duale

		<i>P</i>		
<i>D</i>	finito	✗		
	illimitato			✗
	N.A.	✗		✗

Esercizio: trovare un esempio di primale e duale entrambi non ammissibili.

Proprietà del duale

Teo: Dualità forte

Se (P) e (D) ammettono soluzioni ammissibili allora

$$\min\{c^T x : Ax = b, x \geq 0\} = \max\{y^T b : y^T A \leq c^T\}$$

Scarti complementari

Teo: Scarti complementari

Dati

$$(P) \quad \begin{array}{l} \min c^T x \\ Ax = b \\ x \geq 0 \end{array} \quad (D) \quad \begin{array}{l} \max y^T b \\ y^T A \leq c^T \end{array}$$

e \bar{x} e \bar{y} soluzioni ammissibili per (P) e (D) rispettivamente, allora le tre condizioni seguenti sono equivalenti:

- i) \bar{x} e \bar{y} sono ottime
- ii) $c^T \bar{x} = \bar{y}^T b$
- iii) $(c^T - \bar{y}^T A)\bar{x} = 0$ (equazione agli scarti complementari)

Scarti complementari

Dimostrazione

i) e ii) sono equivalenti per la dualità forte

ii) \Rightarrow iii):

$$c^T \bar{x} = \bar{y}^T b \text{ e } A\bar{x} = b \Rightarrow c^T \bar{x} = \bar{y}^T A\bar{x} \Rightarrow (c^T - \bar{y}^T A)\bar{x} = 0$$

iii) \Rightarrow ii):

$$(c^T - \bar{y}^T A)\bar{x} = 0 \Rightarrow c^T \bar{x} = \bar{y}^T A\bar{x} \Rightarrow c^T \bar{x} = \bar{y}^T b$$

Scarti complementari

Def: Soluzioni complementari

Dati un problema primale (P) e un problema duale (D), due soluzioni \bar{x} e \bar{y} sono dette **complementari** se soddisfano le condizioni degli scarti complementari:

$$(c^T - \bar{y}^T A) \bar{x} = 0$$

Scarti complementari

Teorema

Dati due problemi primale e duale

$$\begin{array}{ll} \min c^T x & \max y^T b \\ (P) \quad Ax \geq b & (D) \quad y^T A \leq c^T \\ x \geq 0 & y \geq 0 \end{array}$$

e due soluzioni \bar{x} e \bar{y} ammissibili rispettivamente per (P) e (D), allora

$$\bar{x} \text{ e } \bar{y} \text{ sono ottime} \Rightarrow \begin{aligned} (c_j - y^T A^j) x_j &= 0 \\ y_i (b_i - A_i x) &= 0 \end{aligned}$$

Scarti complementari

Osservazione

In presenza di soluzioni complementari

- $x_i > 0 \Rightarrow y^T A^i - c_i = 0$
- $y^T A^i - c_i \neq 0 \Rightarrow x_i = 0$

Interpretazione dei costi ridotti

Soluzioni di base complementari

Consideriamo una soluzione di base associata a una sottomatrice A_B .

Una soluzione complementare deve soddisfare questo sistema: $y^T A_B = c_B^T$, poiché solo le variabili in base possono essere non nulle.

Una base indica un sottoinsieme di variabili nel primale e un sottoinsieme di vincoli nel duale che devono essere soddisfatti all'egualanza (**vincoli attivi**).

Perchè la soluzione complementare sia ammissibile, anche i vincoli duali associati alle variabili primarie fuori base devono essere soddisfatti.

La soluzione complementare $\bar{y}^T = c_B^T A_B^{-1}$ deve soddisfare i vincoli

$$y^T A_N \leq c_N^T.$$

Scarti complementari

Coppia simmetrica

Nel caso di coppia simmetrica

$$\begin{array}{ll} \min c^T x & \max y^T b \\ (P) \quad Ax \geq b & (D) \quad y^T A \leq c^T \\ x \geq 0 & y \geq 0 \end{array}$$

le condizioni degli scarti complementari sono:

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

Scarti complementari

Dimostrazione

\bar{x} e \bar{y} ammissibili $\Rightarrow \bar{y}^T b \leq \bar{y}^T A \bar{x} \leq c^T \bar{x}$

\bar{x} e \bar{y} ottime $\Rightarrow \bar{y}^T b = c^T \bar{x}$, da cui

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

Interpretazione dei costi ridotti

Condizioni di complementarietà

Una soluzione del primale \bar{x} e una soluzione del duale \bar{y} sono complementari se soddisfano le condizioni di complementarietà:

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

Condizioni di ottimalità

Se due soluzioni complementari \bar{x} e \bar{y} sono ammissibili, per il problema primale e duale rispettivamente, allora sono ottime.

Interpretazione dei costi ridotti

Condizioni di ottimalità

Data una soluzione di base primale \bar{x} , associata alla base B (e alla matrice di base A_B), e la sua soluzione duale complementare $\bar{y}^T = c_B^T A_B^{-1}$, si possono avere due casi:

- a) \bar{y} è ammissibile, ovvero $\bar{y}^T A_N = c_B^T A_B^{-1} A_N \leq c_N^T \Rightarrow r_N^T = c_N^T - c_B^T A_B^{-1} A_N \geq 0$. La soluzione \bar{x} è ottima;
- b) \bar{y} non è ammissibile, ovvero $\bar{y}^T A_N = c_B^T A_B^{-1} A_N \not\leq c_N^T \Rightarrow r_N^T = c_N^T - c_B^T A_B^{-1} A_N < 0$. La base non è ottima.

Simplesso primale e simplesso duale

- Il simplesso primale si muove verso l'ottimalità primale (ovvero l'ammissibilità duale) mantenendo l'ammissibilità primale
- Il simplesso duale si muove verso l'ottimalità duale mantenendo l'ammissibilità duale ovvero si muove verso l'ammissibilità primale mantenendo la soluzione primale super-ottima → tutti i costi ridotti sono positivi o nulli ma le variabili possono essere negative

N.B. possiamo definire i costi ridotti anche per le variabili in base:

$$r^T = c^T - c_B^T A_B^{-1} A$$

I costi ridotti delle variabili in base sono nulli: $r_B = 0$.
Per una soluzione ottima $r^T = c^T - c_B^T A_B^{-1} A \geq 0$.

Esempio

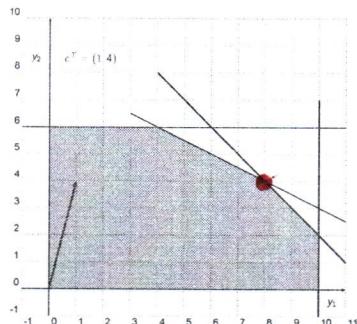


Figura: Esempio: funzione obiettivo $\max y_1 + 4y_2$

Abbiamo visto come il metodo del simplesso mantenga, ad ogni iterazione, due soluzioni primale e duale che sono, per costruzione, in scarti complementari e, quindi, diventano ottime nel momento in cui tutte e due sono ammissibili. In particolare, il metodo del simplesso parte da una soluzione primale ammissibile e cerca di rendere questa soluzione ottimale (costi ridotti non negativi), mantenendo, ad ogni iterazione, una soluzione ammissibile primale. Questo corrisponde, visto che i costi ridotti corrispondono ai vincoli duali, ad avere, ad ogni iterazione, una soluzione duale non ammissibile che, al termine diventa ammissibile. Esistono casi (come vedremo frequenti nelle applicazioni) in cui si ha una soluzione di base primale non ammissibile in scarti complementari con una soluzione ammissibile duale. Chiaramente, vista la non ammissibilità primale, non è possibile applicare direttamente il metodo del simplesso come visto finora. Tuttavia è possibile sfruttare la coppia di soluzioni disponibili per arrivare, in modo molto efficiente, ad una soluzione ottima, attraverso il metodo del simplesso duale. Il metodo del simplesso duale mantiene ad ogni iterazione una base ammissibile nel duale (ovvero una base per la quale i costi ridotti siano non-negativi) e termina quando determina una base che sia ammissibile anche nel primale. Tale metodo può essere interpretato come il metodo del simplesso eseguito sul problema duale invece che sul primale.

Figura: Esempio: funzione obiettivo $\max x_1 + 4x_2$

Simplesso duale

- Ci spostiamo da un vertice ad uno adiacente
- Selezionando, tra le direzioni ammissibile, ovvero le direzioni che garantiscono di rimanere dentro la regione ammissibile, una che abbia una componente positiva lungo il gradiente della funzione obiettivo (direzione ammissibile di crescita)
- Condizione di ottimalità:
Se il gradiente della funzione obiettivo appartiene al cono generato dai gradienti dei vincoli attivi non esistono direzioni ammissibili di crescita e il vertice è ottimo

Problema primale	Problema duale
$\min c^T x$	$\max y^T b$
$Ax = b$	$y^T A \leq c^T$
$x \geq 0$	

Data la base B :

Problema primale	Problema duale
$\min c^T x$	$\max y^T b$
$A_B x_B + A_N x_N = b$	$y^T A_B = c_B^T$
	$y^T A_N \leq c_N^T$
	$x \geq 0$

abbiamo però $x_p < 0, p \in B \rightarrow p$ esce dalla base.

Simplesso duale

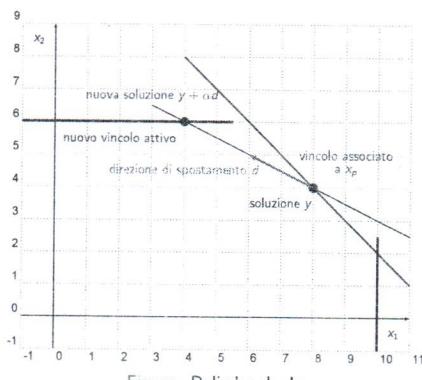


Figura: Poliedro duale

Simplesso duale

Per modificare la base, cambiamo l'insieme dei vincoli attivi spostandoci ad un altro vertice del poliedro duale.
Nella nuova soluzione un vincolo che non era attivo diventa attivo, mentre il vincolo associato a x_p non lo è più.
La direzione di spostamento deve essere

- di crescita (vogliamo muoverci verso l'ottimalità duale, ovvero l'ammissibilità primale)
- ammissibile (vogliamo mantenere la soluzione duale ammissibile)

Simplesso duale

Consideriamo la direzione data dal vettore opposto alla p -esima riga della matrice A_B^{-1} :

$$d^T = -(A_B^{-1})_p.$$

d è sia di crescita che ammissibile.

$d^T = -(A_B^{-1})_p$ è di crescita

$x_p = (A_B^{-1})_p b$ e $x_p < 0$: dunque $d^T b = -(A_B^{-1})_p b > 0$, cioè d è di crescita.

Simplesso duale

$d^T = -(A_B^{-1})_p$ è ammissibile

Consideriamo ora i vincoli non di base: quale variabile primale (vincolo duale) entra in base? I costi ridotti devono rimanere non negativi:

$$c_N^T - (y + \alpha d)^T A_N \geq 0 \rightarrow c_N^T - (c_B^T A_B^{-1} - \alpha (A_B^{-1})_p) A_N \geq 0$$

$$c_N^T - (c_B^T A_B^{-1} A_N) + \alpha (A_B^{-1})_p A_N = r_N^T + \alpha (A_B^{-1})_p A_N \geq 0$$

Simplesso duale

Quale variabile primale (vincolo duale) lascia la base?

2. Esiste $j \notin B : (A_B^{-1})_p A_N^j < 0$: allora α deve essere limitato per non rendere inammissibile il duale,
 $r_j + \alpha (A_B^{-1})_p A_N^j \geq 0 \iff$

$$\alpha \leq -\frac{r_j}{(A_B^{-1})_p A_N^j} \quad \forall j \notin B : (A_B^{-1})_p A_N^j < 0$$

Schema del simplesso duale

- Calcolare $x_B = A_B^{-1} b$, $z = c_B^T A_B^{-1} b$. Se $\forall q \in B, x_q \geq 0$ la base attuale è ammissibile e ottima (la procedura termina)
- Altrimenti selezionare $q \in B : x_q < 0$ (p -esimo elemento della base)
- Se $(A_B^{-1} A_N)_{pi} \geq 0, \forall i \notin B$ il duale è illimitato, il primale non è ammissibile (la procedura termina)
- Altrimenti entra in base al posto di x_p la variabile x_j con

$$j = \arg \min_{i \notin B} \left\{ -\frac{r_i}{(A_B^{-1} A_N)_{pi}} : (A_B^{-1} A_N)_{pi} < 0 \right\}$$
 e si torna a 1

Simplesso duale

$d^T = -(A_B^{-1})_p$ è ammissibile

La nuova soluzione duale sarà $y + \alpha d$, dove $\alpha \geq 0$ è il passo di spostamento. Consideriamo prima i vincoli di base:

- Se $i \in B \setminus \{p\}$: A_B^i indica la i -esima colonna di A_B

$$(y + \alpha d)^T A_B^i = y^T A_B^i + \alpha (-A_B^{-1})_p A_B^i = y^T A_B^i = c_i$$

- Se $i = p$:

$$(y + \alpha d)^T A_B^p = y^T A_B^p + \alpha (-A_B^{-1})_p A_B^p = y^T A_B^p - \alpha = c_p - \alpha \leq c_p$$

Simplesso duale

Quale variabile primale (vincolo duale) lascia la base?

Per ogni variabile x_j con $j \notin B$ si deve avere

$$r_j + \alpha (A_B^{-1})_p A_N^j \geq 0.$$

Possiamo avere due casi:

- $(A_B^{-1})_p A_N^j \geq 0 \quad \forall j \notin B \rightarrow$ nessun costo ridotto diventa negativo, il duale è sempre ammissibile, è possibile muoversi illimitatamente lungo la direzione scelta (d è una direzione di recessione) → il duale è illimitato e il primale è non ammissibile

Schema del simplesso duale

Input Soluzione di base super-ottima ma non ammissibile B
Output Soluzione ammissibile ottima

N.B. Il simplesso duale è sostituibile dalla Fase I del simplesso

Simplesso duale: esempio

Esempio

Risolvere con il simplesso duale il seguente problema:

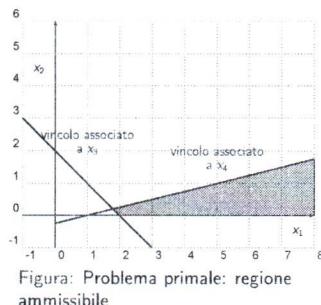
$$\min 5x_1$$

$$x_1 + x_2 \geq 2$$

$$x_1 - 4x_2 \geq 1$$

$$x_1, x_2 \geq 0$$

Simplex duale



Esempio: prima iterazione

Nella matrice dei vincoli c'è una sottomatrice $-I$, associata a x_3, x_4 :

$$A = \left(\begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ 1 & 1 & -1 & 0 \\ 1 & -4 & 0 & -1 \end{array} \right)$$

$$A_B = -I = A_B^{-1}, x_B^T = (-2, -1), r_N^T = c_N^T = (5, 0), z = 0$$

Abbiamo una soluzione ammissibile duale ($y = (0, 0)$) ma non una soluzione ammissibile primale

Esempio: seconda iterazione

$$A_B = \left(\begin{array}{cc} x_2 & x_4 \\ 1 & 0 \\ -4 & -1 \end{array} \right)$$

$$A_B^{-1} = \left(\begin{array}{cc} 1 & 0 \\ -4 & -1 \end{array} \right)$$

$$x_B^T = (x_2, x_4) = (2, -9), r_N^T = c_N^T = (5, 0), z = 0$$

Abbiamo una soluzione ammissibile duale ($y^T = (0, 0)$) ma non una soluzione ammissibile primale. Esce x_4 .

Esempio: terza iterazione

$$A_B = \left(\begin{array}{cc} x_1 & x_2 \\ 1 & 1 \\ 1 & -4 \end{array} \right)$$

$$A_B^{-1} = \left(\begin{array}{cc} \frac{4}{5} & \frac{1}{5} \\ \frac{1}{5} & -\frac{1}{5} \end{array} \right)$$

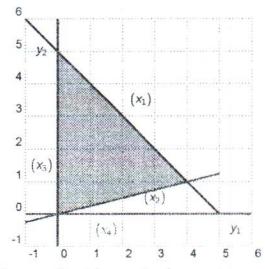
$$x_B^T = (x_1, x_2) = \left(\frac{9}{5}, \frac{1}{5} \right), z = 9$$

Abbiamo una soluzione ammissibile duale ($y^T = c_B^T A_B^{-1} = (4, 1)$) e una soluzione ammissibile primale.

Simplex duale

Problema duale:

$$\begin{aligned} \max 2y_1 + y_2 \\ y_1 + y_2 \leq 5 \\ y_1 - 4y_2 \leq 0 \\ -y_1 \leq 0 \\ -y_2 \leq 0 \end{aligned}$$



Esempio: prima iterazione

$x_p = x_3$:

$$A_B^{-1} A_N = \left(\begin{array}{cc|c} (x_1) & (x_2) & \\ -1 & -1 & (x_3) \\ -1 & 4 & (x_4) \end{array} \right)$$

$$j = \arg \min \{-\frac{5}{1}, -\frac{0}{1}\} = 0$$

Esce x_3 entra x_2 . Nuova base $\{x_2, x_4\}$

N.B.: $d^T = (1, 0), \alpha = 0$: soluzione duale degenere

Esempio: seconda iterazione

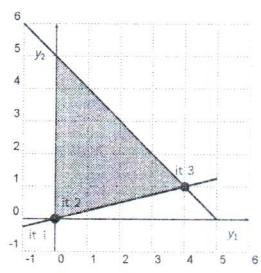
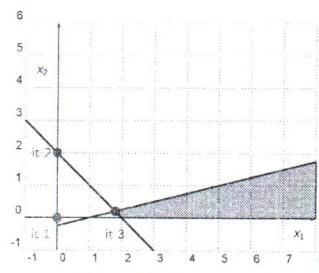
$x_p = x_4$:

$$A_B^{-1} A_N = \left(\begin{array}{cc|c} (x_1) & (x_3) & \\ 1 & -1 & (x_2) \\ -5 & 4 & (x_4) \end{array} \right)$$

Entra x_1 . Nuova base $\{x_1, x_2\}$

N.B.: $d^T = (4, 1), \alpha = 1$

Simplex duale



Osservazione sugli scarti complementari

Consideriamo il problema

$$\begin{aligned} \max 2y_1 + y_2 \\ y_1 + y_2 &\leq 12 \\ y_1 + 2y_2 &\leq 16 \\ y_1 &\leq 8 \\ y_2 &\leq 6 \\ -y_1 &\leq 0 \\ -y_2 &\leq 0 \end{aligned}$$

e applichiamo le condizioni di complementarità per valutare se la soluzione $(8, 4)$ è ottima.

Osservazione sugli scarti complementari

Le equazioni agli scarti complementari sono:

$$\begin{aligned} x_1(y_1 + y_2 - 12) &= 0 \\ x_2(y_1 + 2y_2 - 16) &= 0 \\ x_3(y_1 - 8) &= 0 \\ x_4(y_2 - 6) &= 0 \\ x_5(-y_1) &= 0 \\ x_6(-y_2) &= 0 \end{aligned}$$

da cui si ottiene $x_4, x_5, x_6 = 0$.

Osservazione sugli scarti complementari

Il duale del problema è:

$$\begin{aligned} \min 12x_1 + 16x_2 + 8x_3 + 6x_4 \\ x_1 + x_2 + x_3 - x_5 &= 2 \\ x_1 + 2x_2 + x_4 - x_6 &= 1 \\ x_1, x_2, x_3, x_4, x_5, x_6 &\geq 0 \end{aligned}$$

Osservazione sugli scarti complementari

Cosa succede dal punto di vista dei coni generati dai gradienti dei vincoli attivi (soddisfatti all'egualianza)?

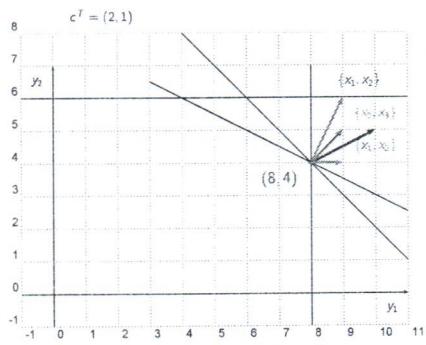


Figura: Coni generati dai gradienti dei vincoli

Osservazione sugli scarti complementari

Possiamo fissare arbitrariamente un'altra variabile duale a 0, ottenendo 3 soluzioni duali complementari.

- $\{x_1, x_2\} = (3, -1)$
- $\{x_1, x_3\} = (1, 1)$
- $\{x_2, x_4\} = (\frac{1}{2}, \frac{3}{2})$

la prima soluzione complementare non è ammissibile, ma la soluzione primale è ottima, come mostrato dalle altre due soluzioni complementari.

Stiamo lavorando su soluzioni di base sia duali (sottoinsiemi di 2 variabili) sia primali (sottoinsiemi di 2 vincoli linearmente indipendenti)

A una stessa soluzione corrispondono più basi: a seconda di quale sottoinsieme dei vincoli attivi (o di variabili non nulle) si sceglie la base è ottima o meno.

Osservazione sugli scarti complementari

- A seconda di quali vincoli attivi si scelgono per comporre la base, il gradiente della funzione obiettivo appartiene o meno al cono generato dai loro gradienti
- Le variabili x di base rappresentano i coefficienti della combinazione conica
- Quando il gradiente della funzione obiettivo è combinazione conica dei vincoli attivi, non esiste una direzione ammissibile di crescita (Farkas)

5. Grafi (2)

Introduzione, algoritmo di visita
e albero di copertura di costo minimo
Fondamenti di Ricerca Operativa - A.A. 2018/2019

Giuliana Carello
DEIB, Politecnico di Milano

- Modellazione in Programmazione Lineare
- Proprietà della Programmazione Lineare (a variabili continue) e metodi di soluzione
- Una classe particolare di problemi di Ottimizzazione: i problemi su grafo
 - Albero di copertura ottimo
 - Albero dei cammini minimi
 - Flusso massimo
- Metodi per la Programmazione Lineare Intera

Grafo

Grafo direzionato

Si dice **grafo** una coppia di insiemi (N, A)

- N : insieme dei **nodi**
- $A \subseteq N \times N$: sottoinsieme delle coppie ordinate dei nodi (rappresenta una relazione tra nodi)
- $(i, j) \in A$ è detto **arco**

Un grafo direzionato ha al più $n(n - 1)$ archi, dove $n = |N|$.

Grafo

Dato $G = (N, A)$

- i nodi i e j sono gli **estremi** dell'arco
- i è la **coda** dell'arco, mentre j è la **testa**

Dato un nodo $i \in N$

- **Insieme dei successori di i :** $FN(i) = \{j \in N : (i, j) \in A\}$
- **Insieme dei predecessori di i :** $PN(i) = \{j \in N : (j, i) \in A\}$
- **Insieme degli adiacenti di i :** $AD(i) = FN(i) + PN(i)$
- **Stella uscente:** $FS(i) = \{(i, j) \in A\}$
- **Stella entrante:** $BS(i) = \{(j, i) \in A\}$

Grafo

Dato $G = (N, A)$

- un **cammino** sul grafo è una sequenza di archi presi secondo il loro verso di percorrenza che collega due nodi s e t
 - un **cammino semplice** è un cammino senza archi ripetuti
 - un **cammino elementare** è un cammino senza nodi ripetuti
- un **ciclo** è un cammino chiuso in cui è ripetuto solo il primo nodo ($s = t$)

Grafo

Grafo non direzionato

- Se le coppie non sono ordinate il grafo è detto **non direzionato** (o **simmetrico**): $G = (N, E)$
- in questo caso si parla di **lati**
- $\{i, j\} \in E$ è detto **lato** e i e j sono i suoi **estremi**
- due nodi sono **adiacenti** se sono collegati da un lato
- un lato è **incidente** nei suoi estremi

Si estendono le definizioni di cammino e ciclo

Un grafo non direzionato $G = (N, E)$ può essere rappresentato come un grafo direzionato $G = (N, A)$ tale che, per ogni lato di E , A contiene la corrispondente coppia di archi:

$$\forall e = \{i, j\} \in E \quad (i, j) \in A, (j, i) \in A$$

Un grafo non direzionato ha al più $\frac{n(n-1)}{2}$ lati, dove $n = |N|$.

Problemi su grafo

- Visita di un grafo (verifica di connettività)
- Albero di copertura di costo minimo
- Albero dei cammini minimi
- Flusso massimo
- Flusso di costo minimo

Connettività di un grafo

Def.: Nodi connessi

Dato un grafo $G = (N, A)$, due nodi $i, j \in N$ sono **connessi** se esiste un cammino che li collega

Def.: Grafo connesso

Un grafo $G = (N, A)$ è **connesso** se ogni coppia di nodi $i, j \in N$ è connessa

Connettività di un grafo

Problema

Dato un grafo orientato $G = (N, A)$ e una coppia di nodi $s, t \in N$, esiste un cammino che collega s a t ?
(il grafo è connesso?)

Osservazioni

- Per mostrare che il cammino esiste è sufficiente descriverlo
- Per mostrare che non esiste dobbiamo enumerare tutti i cammini oppure sfruttare una proprietà dei grafi

archi di taglio:
 $(2, 4), (3, 5)$
 $(= \text{archi in avanti},$
 $\not\exists \text{ archi all'indietro})$

Proprietà

Def.: Archi (o lati) del taglio

Si dicono *archi del taglio* gli archi che collegano i due sottoinsiemi della partizione:

$$\{(i, j) : i \in N_S, j \in N_T\} \cup \{(i, j) : i \in N_T, j \in N_S\}$$

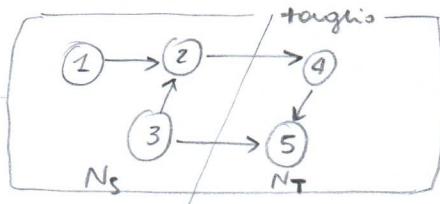
- Archi in avanti:* $\{(i, j) : i \in N_S, j \in N_T\}$
- Archi all'indietro:* $\{(i, j) : i \in N_T, j \in N_S\}$

Proprietà

Def.: Taglio di un grafo

Dato un grafo $G = (N, A)$, si definisce *taglio* una partizione dell'insieme dei nodi

$$N_T \subseteq N, N_S \subseteq N : N_T \cap N_S = \emptyset, N_T \cup N_S = N.$$



Proprietà

Proprietà

Dato un grafo orientato $G = (N, A)$ e due nodi $s, t \in N$, una sola di queste due affermazioni è vera:

- a) esiste un cammino che collega s a t
- b) esiste un taglio (N_S, N_T) : $s \in N_S, t \in N_T$ e tutti gli archi (i, j) del taglio sono all'indietro, $i \in N_T, j \in N_S$.

Per dimostrare questa proprietà usiamo una dimostrazione costruttiva che fa uso dell'*algoritmo di visita di un grafo*



Algoritmo di visita di un grafo

Input

il grafo $G = (N, A)$ e due nodi $s, t \in N$

Output il cammino che li collega, o il taglio che li separa

L' algoritmo usa:

- una lista Q per contenere i nodi visitati
- un parametro P per ogni nodo che contiene il nodo che lo precede nel cammino, il *predecessore*

Schema dell'algoritmo

- (inizializzazione) $P[i] = 0, \forall i \in N, P[s] = s; Q = \{s\}$
- si estrae un nodo $i \in Q$
- se $i = t$ il cammino è stato trovato e l'algoritmo termina
- altrimenti si scorrono tutti gli archi della stella uscente di i :
 $\forall (i, j) \in FS(i)$
 - se $P[j] = 0$ allora $P[j] = i, Q = Q \cup \{j\}$
- se $Q = \emptyset$ i nodi non sono connessi e l'algoritmo termina, altrimenti si torna al passo 2

Esempio

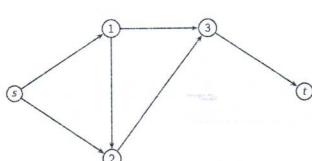


Figura: Connessione di un grafo

Applicazione dell'algoritmo

- $P[i] = 0, \forall i \in N, P[s] = s, FS(s) = \{(s, 1), (s, 2)\}$
- $(s, 1) : P[1] = 0 \rightarrow P[1] = s, Q = Q \cup \{1\}$
- $(s, 2) : P[2] = 0 \rightarrow P[2] = s, Q = Q \cup \{2\}$
- $Q = \{1, 2\}$, estraggo $i = 1, Q = \{2\}, FS(1) = \{(1, 2), (1, 3)\}$
- $(1, 2) : P[2] \neq 0$
- $(1, 3) : P[3] = 0 \rightarrow P[3] = 1, Q = Q \cup \{3\}$
- $Q = \{2, 3\}$, estraggo $i = 2, Q = \{3\}, FS(2) = \{(2, 3)\}$
- $(2, 3) : P[3] \neq 0$
- $Q = \{3\}$, estraggo $i = 3, Q = \emptyset, FS(3) = \{(3, t)\}$
- $(3, t) : P[t] = 0 \rightarrow P[t] = 3, Q = Q \cup \{t\}$
- $Q = \{t\}$, estraggo $i = t \rightarrow \text{STOP}$

Il cammino si ricostruisce a ritroso: $t, P[t] = 3, P[3] = 1, P[1] = s$

Esempio

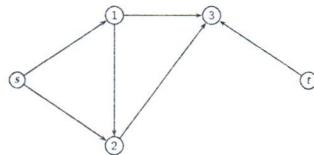


Figura: Connessione di un grafo

Applicazione dell'algoritmo

- $P[i] = 0, \forall i \in N, P[s] = s, Q = s, FS(s) = \{(s, 1), (s, 2)\}$
- $(s, 1) : P[1] = 0 \rightarrow P[1] = s, Q = Q \cup \{1\}$
- $(s, 2) : P[2] = 0 \rightarrow P[2] = s, Q = Q \cup \{2\}$
- $Q = \{1, 2\}$, estraggo $i = 1, Q = \{2\}, FS(1) = \{(1, 2), (1, 3)\}$
- $(1, 2) : P[2] \neq 0$
- $(1, 3) : P[3] = 0 \rightarrow P[3] = 1, Q = Q \cup \{3\}$
- $Q = \{2, 3\}$, estraggo $i = 2, Q = \{3\}, FS(2) = \{(2, 3)\}$
- $(2, 3) : P[3] \neq 0$
- $Q = \{3\}$, estraggo $i = 3, Q = \emptyset, FS(3) = \emptyset$
- $Q = \emptyset \rightarrow \text{STOP}$

Non esiste cammino da s a t , $P[t] = 0$

Proprietà

Proprietà

Dato un grafo orientato $G = (N, A)$ e due nodi $s, t \in N$, una sola di queste due affermazioni è vera:

- esiste un cammino che collega s a t
- esiste un taglio $(N_S, N_T) : s \in N_S, t \in N_T$ e tutti gli archi (i, j) del taglio sono all'indietro, $i \in N_T, j \in N_S$.

Proprietà

Dimostrazione

Applicando l'algoritmo di visita si possono verificare due casi:

- $P[t] \neq 0$: il nodo t è raggiungibile → caso a)
- $P[t] = 0$: il nodo t non è raggiungibile. N è diviso in due sottoinsiemi:
 - N_S è l'insieme dei nodi raggiunti nella visita
 $N_S = \{i \in N : P[i] \neq 0\}$
 - N_T è l'insieme dei nodi che non sono raggiunti nella visita
 $N_T = \{i \in N : P[i] = 0\}$

(N_S, N_T) è una partizione. Per l'algoritmo di visita non esiste un arco $(i, j) : i \in N_S, j \in N_T$, altrimenti anche il nodo j sarebbe stato visitato e avrebbe $P[j] \neq 0$

Costo computazionale dell'algoritmo

Quante operazioni *elementari* (somma, sottrazione, confronto, ...) esegue l'algoritmo nella peggiore delle ipotesi?

Complessità asintotica

$f(n), g(n)$: funzioni del numero di operazioni

Si dice che $f(n)$ è dell'ordine di $g(n)$ e si scrive $f(n) = O(g(n))$
se

$\exists c > 0, \exists n_0 : f(n) \leq cg(n), n > n_0$

Nel caso peggiore devo estrarre tutti i nodi prima di estrarre t
→ $|N|$ estrazioni

Per ogni nodo estratto visito tutta la stessa uscente ⇒ $|A|$ visite
Ordine di $|N||A|$ operazioni $O(|N||A|)$

In realtà ogni arco viene controllato al più una volta ⇒ $O(|A|)$

Albero di copertura di costo minimo

:= *albero senza cicli*

= *minore sottografo senza cicli*

Definizione

Dato un grafo non direzionale $G = (N, E)$, in cui a ogni lato $e \in E$ è associato un costo, o peso, $w_e \geq 0$, trovare il sottoinsieme di lati T di costo minimo che collega tutti i nodi. Il costo del sottoinsieme è dato dalla somma dei costi dei lati che lo compongono $\sum_{e \in T} w_e$

Albero di copertura di costo minimo

Caratteristiche della soluzione

- Sottoinsieme dei lati → sottografo:
 $G' = (N', E') : N' \subseteq N, E' \subseteq E$, dove ogni $\{i, j\} \in E'$ collega due nodi in N'
- che tocca tutti i nodi → connesso: $N' = N$
- senza cicli → aciclico

Un sottografo connesso aciclico è un *albero di copertura*

Proprietà

Osservazione 1

Un albero con n nodi ha $n - 1$ lati

Osservazione 2

Su un albero, ogni coppia di nodi è collegata da un unico cammino

Osservazione 4

Eliminando da un albero T un lato si ottiene un taglio

Osservazione 3

Aggiungendo a un albero un lato che non gli appartiene si genera un unico ciclo

Proprietà

Osservazione 5 (proprietà di scambio)

Si consideri un albero di copertura T su un grafo $G = (N, E)$ e un lato $e \notin T$. $T \cup \{e\}$ contiene un ciclo C . Sostituendo e a ogni lato del ciclo C si ottiene un altro albero di copertura: $\forall f \in C \setminus \{e\}, T \cup \{e\} \setminus \{f\}$ è un albero di copertura.

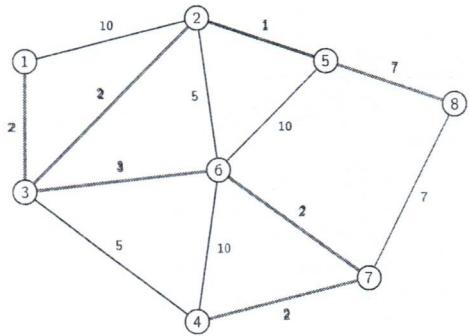
Condizioni di ottimalità

Condizione di ottimalità sui tagli

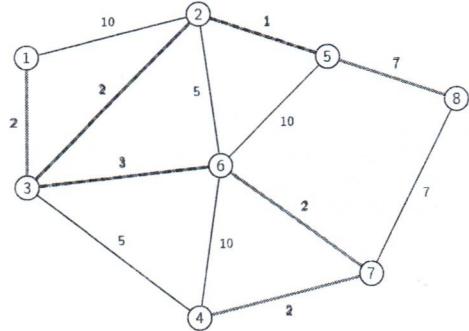
Un albero di copertura T^* è un albero di copertura di costo minimo se soddisfa la seguente condizione:

per ogni lato dell'albero $e \in T^*$ si ha che $w_e \leq w_i$, per ogni lato i che appartiene al taglio generato eliminando e dall'albero

Condizioni di ottimalità



Condizione di ottimalità



Algoritmo di Kruskal

Input il grafo $G = (N, E)$, il costo w_e di ogni lato
Output un albero di costo minimo T

Condizioni di ottimalità

Condizione di ottimalità sui cammini

Un albero di copertura T^* è un albero di copertura di costo minimo se soddisfa la seguente condizione:

per ogni lato che non appartiene all'albero $i \notin T^*$, con $i = \{i, j\}$, si ha che $w_e \leq w_i$, per ogni lato e che appartiene all'albero T^* e al cammino che sull'albero T^* collega i a j .

Algoritmo di Kruskal: idea generale dell'algoritmo

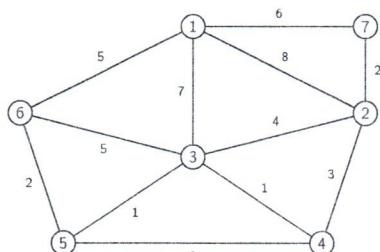
- ▶ La soluzione viene costruita passo passo, aggiungendo un lato per volta
- ▶ si aggiungono i lati dal meno costoso al più costoso (albero di costo minimo)
- ▶ non si inseriscono i lati che creano cicli (aciclicità)
- ▶ la soluzione è completa quando tutti i nodi sono collegati o sono stati inseriti $|N| - 1$ lati

Algoritmo di Kruskal

Schema dell'algoritmo

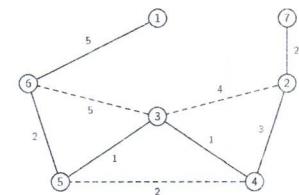
1. (inizializzazione) $T = \emptyset$
2. si ordinano i lati per costo non decrescente
3. si seleziona e , il prossimo lato nell'ordine
4. se $T \cup \{e\}$ contiene un ciclo si torna al passo 3
5. altrimenti si aggiunge e all'albero, $T = T \cup \{e\}$
6. se $|T| = |N| - 1$ l'algoritmo termina
7. altrimenti si torna al passo 3

Algoritmo di Kruskal: esempio



Algoritmo di Kruskal: esempio

- ▶ {3, 4}
- ▶ {3, 5}
- ▶ {5, 6}
- ▶ {4, 5}: crea ciclo
- ▶ {2, 7}
- ▶ {2, 4}
- ▶ {2, 3}: crea ciclo
- ▶ {3, 6}: crea ciclo
- ▶ {1, 6}: tutti connessi: STOP



Algoritmo di Kruskal

Come si controlla se $T \cup \{e\}$ contiene un ciclo? Usando una lista per ogni sottoinsieme di nodi già collegati dall'insieme T e assegnando a ciascuna lista un'etichetta: se i due estremi di un lato sono associati alla stessa etichetta, allora aggiungere il lato a T crea un ciclo.

Quando si aggiunge un lato, si uniscono le due liste a cui appartengono i suoi estremi

Algoritmo di Kruskal

Tempo computazionale

1. (inizializzazione) $T = \emptyset$
2. si ordinano i lati per costo non decrescente $\rightarrow O(|E| \log |E|)$
3. si seleziona e , il prossimo lato nell'ordine
4. se $T \cup \{e\}$ contiene un ciclo si torna al passo 3
 $\rightarrow O(|N| \log |N|)$
5. altrimenti si aggiunge e all'albero, $T = T \cup \{e\} \rightarrow O(|E|)$
6. se $|T| = |N| - 1$ l'algoritmo termina
7. altrimenti si torna al passo 3

Osservazioni

- ▶ L'algoritmo di Kruskal è un algoritmo di tipo *greedy*:
 - ▶ crea la soluzione aggiungendo un elemento per volta
 - ▶ non ridiscute decisioni già prese
 - ▶ impiega un tempo polinomiale in funzione delle dimensioni del problema (numero di nodi e di lati, in questo caso)

Correttezza dell'algoritmo di Kruskal

Teo.: correttezza dell'algoritmo di Kruskal

L'algoritmo di Kruskal fornisce una soluzione ottima del problema dell'albero di copertura di costo minimo, ovvero genera un albero di copertura di costo minimo.

Dimostrazione

Sia T_K l'albero generato dall'algoritmo di Kruskal e T un albero di copertura qualunque. Consideriamo un generico lato $\{i, j\} \in T_K$. Eliminando $\{i, j\}$ da T_K si ottiene un sottografo non connesso e si trova un taglio (N_i, N_j) . Nell'albero T esiste un lato $\{p, q\}$ che connette N_i e N_j .

Correttezza dell'algoritmo di Kruskal

Dimostrazione

Se $\{p, q\} \neq \{i, j\}$ allora $T_K \cup \{p, q\}$ contiene un ciclo \mathcal{C} e $\{i, j\} \in \mathcal{C}$.

Per come è costruito T_K si ha che $w_{\{ij\}} \leq w_{\{pq\}}$.

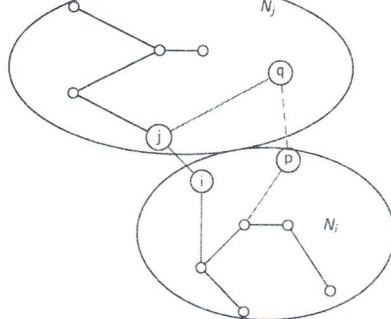
Se $\{p, q\} = \{i, j\}$ allora $w_{\{ij\}} = w_{\{pq\}}$.

Ripetendo il ragionamento per ogni lato di T_K si ottiene che

$$w(T_K) = \sum_{\{ij\} \in T_K} w_{\{ij\}} \leq \sum_{\{pq\} \in T} w_{\{pq\}} \leq w(T)$$

quindi T_K è ottimo.

Correttezza dell'algoritmo di Kruskal



Percorso minimo

Modelli di Programmazione Lineare - Modelli di problemi su grafo

Fondamenti di Ricerca Operativa - A.A. 2017/2018

Giuliana Carello
DEIB, Politecnico di Milano

Un autista di camion deve trasportare un carico da una città s a una città t . Sapendo che la rete stradale è descritta da un grafo $G = (N, A)$ in cui a ciascun arco (i, j) è associata una distanza c_{ij} , come è possibile rappresentare in programmazione lineare il problema di trovare il percorso più breve?

Percorso minimo - modello

Variabili

Quali archi sono usati nel percorso? $\rightarrow x_{ij} \in \{0, 1\}, \forall (i, j) \in A$

$$x_{ij} = \begin{cases} 1, & \text{l'arco } (i, j) \text{ appartiene al cammino minimo;} \\ 0, & \text{altrimenti.} \end{cases}$$

Funzione obiettivo

$$\min \sum_{(i,j) \in A} c_{ij} x_{ij}$$

Percorso minimo - modello

Vincoli

- Il cammino deve partire da s : $\sum_{(s,j) \in A} x_{sj} = 1$
- Il cammino deve raggiungere t : $\sum_{(i,t) \in A} x_{it} = 1$
- Il cammino non deve fermarsi ne' essere originato nei nodi intermedi: $\sum_{(i,j) \in A} x_{ij} = \sum_{(j,i) \in A} x_{ji} = 0, \forall i \neq s, t$

Percorso minimo - modello

Vincoli

Possiamo sostituire i tre vincoli con un vincolo di bilanciamento del flusso:

$$\sum_{(i,j) \in A} x_{ij} - \sum_{(j,i) \in A} x_{ji} = \begin{cases} 1, & i = s; \\ -1, & i = t; \\ 0, & i \neq s, t. \end{cases} \quad \forall i \in N$$

Percorso minimo - modello

Cosa succede usando variabili continue?

- $x_{ij} \in \{0, 1\} \rightarrow x_{ij} \geq 0$? (il cammino si può biforcire)
- La soluzione ottima non cambia:
 - Non è vantaggioso che $x_{ij} > 1$
 - Se esiste una soluzione a costo minimo in cui il cammino si biforca, allora esiste una soluzione a costo equivalente in cui il cammino è unico
- Possiamo scrivere il modello con variabili continue non negative
- Possiamo rappresentare il problema come un problema di flusso di costo minimo, in cui il percorso è rappresentato da una unità di flusso che deve andare da s a t

Albero dei cammini minimi

Problema

È dato un grafo $G = (N, A)$, a ogni arco del quale è associato un costo c_{ij} . Dato un nodo radice $r \in N$ si vogliono trovare i cammini minimi dal nodo r a tutti gli altri nodi del grafo.

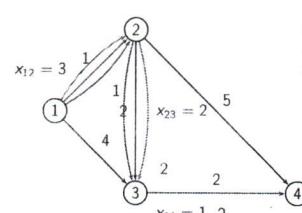
Variabili

- Rappresentiamo ognuno degli $|N| - 1$ cammini con una unità di flusso
- Dobbiamo decidere quanti cammini usano ogni arco:
 $\rightarrow x_{ij} \geq 0, \forall (i, j) \in A$
- x_{ij} rappresenta il numero di cammini che usano l'arco (i, j)

Albero dei cammini minimi - modello

Funzione obiettivo

$$\sum_{(i,j) \in A} c_{ij} x_{ij}$$



Cammino da 1 a 2 = 1
Cammino da 1 a 3 = 3
Cammino da 1 a 4 = 5

$$\sum_{(i,j) \in A} c_{ij} x_{ij} = 9$$

$$x_{12} = 3 \quad x_{23} = 2 \quad x_{34} = 1 \quad x_{31} = 2$$

Albero dei cammini minimi - modello

Vincoli

- Dal nodo radice deve partire un cammino ($\rightarrow |N| - 1$ unità di flusso) per ogni nodo
- In ogni nodo diverso dalla radice deve arrivare un cammino (\rightarrow un'unità di flusso)

$$\sum_{(i,j) \in A} x_{ij} - \sum_{(j,i) \in A} x_{ji} = \begin{cases} |N| - 1, & i = r; \\ -1, & i \neq r; \end{cases}, \forall i \in N$$

N.B. Si definisce *albero* un sottografo (ovvero un sottoinsieme di archi) che non contiene cicli

Albero dei cammini minimi - grafo non direzionato

Grafo non direzionato

La descrizione del problema e il modello cambiano se si cercano i cammini minimi su un grafo non direzionato $G = (N, E)$?

- Problema e modello non cambiano
- È possibile rappresentare un grafo non direzionato come grafo direzionato simmetrico $G' = (N, A)$:

- Se esiste il lato $\{i, j\} \in E$, di costo c_{ij} , allora nel grafo direzionato G' esistono sia l'arco (i, j) che l'arco (j, i) , e hanno lo stesso costo.

Flusso massimo

Come cambia il problema se tralasciamo i costi, assegnamo un capacità a ogni arco e, scelti due nodi s e t , vogliamo massimizzare il flusso che parte da s e raggiunge t senza violare le capacità sugli archi?

Flusso massimo - modello

Variabili

Quantità di flusso che passa sull'arco (i, j) : $x_{ij} \geq 0$
Quantità di flusso che parte da s e arriva a t : $v \geq 0$

Funzione obiettivo

$$\max v$$

Modello

Vincoli

$$x_{ij} \leq u_{ij}, \quad \forall (i, j) \in A$$

$$\sum_{(i,j) \in A} x_{ij} - \sum_{(j,i) \in A} x_{ji} = \begin{cases} v, & i = s; \\ -v, & i = t; \\ 0, & i \in N \setminus \{s, t\}. \end{cases}$$

Albero dei cammini minimi

6. Grafi

Albero dei cammini minimi

Fondamenti di Ricerca Operativa - A.A. 2018/2019

Giuliana Carello
DEIB, Politecnico di Milano

Definizione

Dato un grafo, che può essere sia direzionale che non direzionato, $G = (N, A)$, in cui a ogni arco $(i, j) \in A$ è associato un costo, o peso, c_{ij} , e un nodo sorgente $r \in N$ trovare il percorso più breve dal nodo r a tutti gli altri nodi del grafo.

Albero dei cammini minimi

Modello

$$\begin{aligned} \min & \sum_{(i,j) \in A} c_{ij} x_{ij} \\ \sum_{(j,i) \in A} x_{ji} - \sum_{(i,j) \in A} x_{ij} &= \begin{cases} -(|N|-1), i = r \\ 1, i \neq r \end{cases}, \forall i \in N \\ x_{ij} \geq 0, & \forall (i,j) \in A \end{aligned}$$

Albero dei cammini minimi

Caratteristiche della soluzione (caso non direzionato)

L'insieme degli archi che appartengono ad almeno ad un cammino deve:

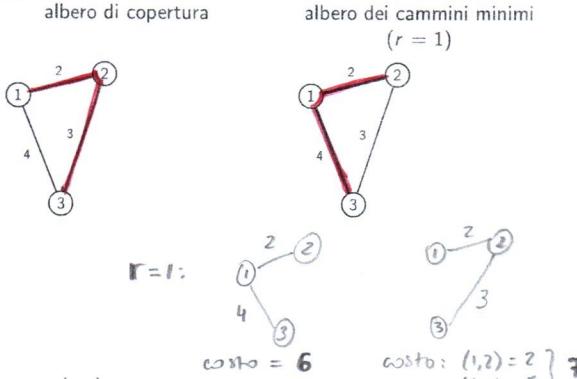
- collegare tutti i nodi
- non avere cicli (basta un percorso da r a ciascuno degli altri nodi)

Una soluzione del problema dei cammini minimi è rappresentata da un albero di copertura

Albero dei cammini minimi

Problema duale

Albero di copertura di costo minimo e albero dei cammini minimi



Modello

$$\begin{aligned} \min & \sum_{(i,j) \in A} c_{ij} x_{ij} \\ \sum_{(j,i) \in A} x_{ji} - \sum_{(i,j) \in A} x_{ij} &= \begin{cases} -(|N|-1), i = r \\ 1, i \neq r \end{cases}, \forall i \in N \quad (\pi_i) \\ x_{ij} \geq 0, & \forall (i,j) \in A \end{aligned}$$

Problema duale

Condizione di ottimalità

Modello duale

$$\begin{aligned} \max & -(|N|-1)\pi_r + \sum_{i \in N: i \neq r} \pi_i \\ \pi_j - \pi_i & \leq c_{ij}, \forall (i,j) \in A \end{aligned}$$

Condizione di ottimalità

Una soluzione duale ammissibile assegna ad ogni nodo $i \in N$ un'etichetta π_i tale che:

$$\pi_j \leq \pi_i + c_{ij}$$

per ogni arco del grafo.

L'etichetta rappresenta la lunghezza del cammino da r a ciascun nodo. All'ottimo l'etichetta del nodo $i \in N$ rappresenta il più breve tra tutti i percorsi che passano da tutti i predecessori di i .

Casi

- Caso di grafo aciclico
- Caso di grafo con cicli, ma costi non negativi
- Caso generale (presenza di cicli e costi anche negativi)



Caso aciclico

- Prima si fissa l'ordinamento topologico, descritto da un'etichetta $\phi_i, \forall i \in N$
- si assegna $d[r] = 0$
- si scorrono i nodi secondo l'indice ϕ ; e si valuta l'impatto della loro stessa uscente
- quando si controlla il nodo i si sono già controllati tutti i possibili cammini fino a i

Caso aciclico

- I sottocammini di un cammino minimo sono a loro volta cammini minimi
- in assenza di cicli i nodi del grafo si possono ordinare topologicamente
- ad ogni nodo è associata un'etichetta $d[i]$ che rappresenta la lunghezza del cammino
- dopo aver assegnato un valore all'etichetta $d[i]$ di tutti i predecessori di un nodo, è possibile fissare l'etichetta del nodo, confrontando la lunghezza dei diversi percorsi

Ordinamento topologico

I nodi sono ordinati topologicamente se l'indice di un nodo è sempre inferiore all'indice di tutti i suoi successori:

$$(i, j) \in A \implies i < j$$

Enumerazione topologica

Input Il grafo $G = (N, A)$

Output Etichette ϕ_i , che rappresentano l'ordinamento topologico

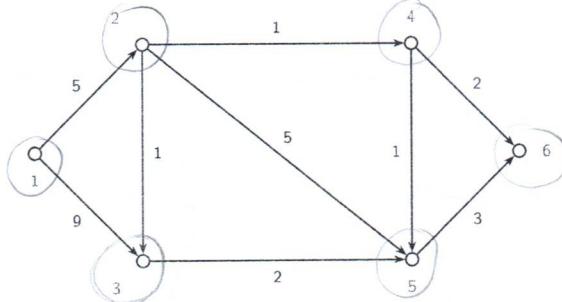
L'algoritmo ha una struttura ricorsiva: si applica a una grafo via via modificato $G' = (N', A')$, mantenendo il prossimo valore dell'etichetta da assegnare nel parametro x

Enumerazione topologica

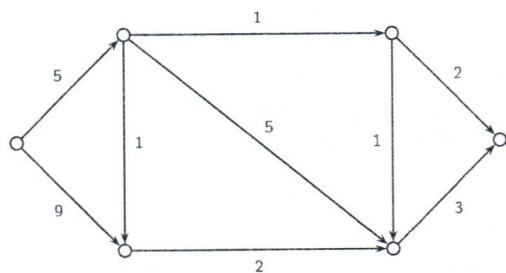
Schema dell'algoritmo

1. $G'(N', A') = G(N, A), x = 1$
2. se $|N'| = 0$ l'algoritmo termina
3. altrimenti si cerca un nodo $i \in N'$ tale che $PN(i) = \emptyset$
 - se non esiste nessun nodo $i \in N'$ tale che $PN(i) = \emptyset$ l'algoritmo termina: il grafo non è aciclico
 - altrimenti si assegna l'etichetta a a i e si aggiornano il grafo e il parametro: $\phi(i) = x, x = x + 1, N' = N' \setminus \{i\}, A' = A' \setminus FS(i)$ e si torna a 2

Esempio



Esempio



Algoritmo SPT-aciclico

Input Un grafo $G = (N, A)$, dove i nodi sono ordinati secondo l'ordinamento topologico, i costi associati a ogni arco e un nodo radice r

Output Il cammino minimo da r a ogni altro nodo

L'algoritmo usa due etichette per ogni nodo:

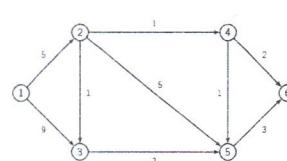
- $d[i]$ che all'ottimo rappresenta la lunghezza del cammino minimo da r a i
- $P[i]$ che rappresenta il predecessore di i nel cammino minimo da r a i

Algoritmo SPT-aciclico

Schema dell'algoritmo

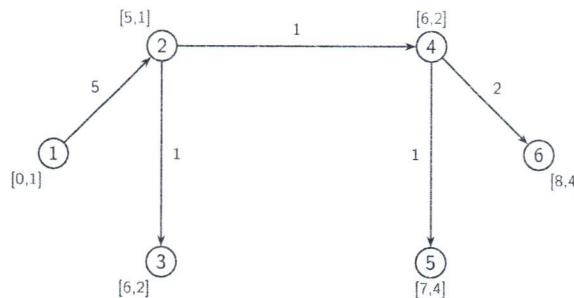
- (inizializzazione) $P[i] = r, d[i] = \infty, \forall i \in N$
- si considera i il prossimo nodo nell'ordinamento topologico
- si scorrono gli archi $(i, j) \in FS(i)$:
 - se $d[i] + c_{ij} < d[j]$ allora $d[j] = d[i] + c_{ij}, P[j] = i$
- se ci sono ancora nodi da esplorare, si torna al passo 2

Esempio



i	$d[i]$	$P[i]$	aggiornamenti
1	0	1	$d[2] = 5, P[2] = 1;$ $d[3] = 9, P[3] = 1$
2	5	1	$d[3] = 6, P[3] = 2;$ $d[4] = 6, P[4] = 2;$ $d[5] = 10, P[5] = 2$
3	6	2	$d[5] = 8, P[5] = 3$
4	6	2	$d[5] = 7, P[5] = 4;$ $d[6] = 8, P[6] = 4$
5	7	4	
6	8	4	

Esempio



Osservazioni

Caso c_{ij} anche negativo

L'algoritmo può essere applicato anche in caso di costi negativi.
E in caso di grafo non direzionato?

Complessità dell'algoritmo

Ogni arco viene controllato al più una volta → il costo dell'algoritmo è $O(|A|) +$ costo dell'ordinamento topologico.

Caso con cicli, $c_{ij} \geq 0$

Idea dell'algoritmo

Presenza di cicli

In presenza di cicli non è possibile ordinare topologicamente i nodi

Caratteristiche dell'algoritmo

- ▶ Aggiornare le etichette
- ▶ Visitare i nodi uno per volta e fissare la lunghezza del percorso minimo
- ▶ Non ritornare sui nodi già visitati

Input Un grafo $G = (N, A)$, i costi associati a ogni arco e un nodo radice r

Output Il cammino minimo da r a ogni altro nodo

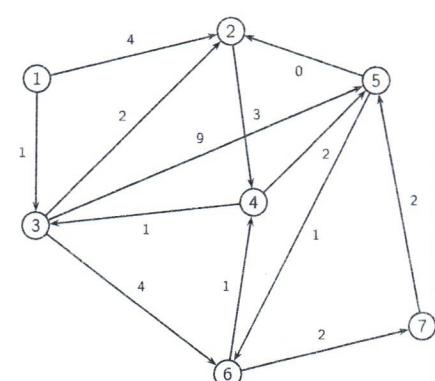
- ▶ Oltre a $d[i]$ e $P[i]$ l'algoritmo usa un insieme Q , l'insieme dei nodi da visitare (e di cui bisogna fissare il cammino minimo)
- ▶ Seleziona da Q un nodo di cui fissa il cammino minimo
- ▶ Seleziona il nodo con $d[i]$ minima: il cammino non può essere migliorato esplorando la stessa uscente degli altri nodi in Q ($c_{ij} \geq 0$)
- ▶ Ogni nodo entra in Q una sola volta

Algoritmo di Dijkstra (1959)

Schema dell'algoritmo

- (inizializzazione) $P[r] = r, d[r] = 0, Q = \{r\}$
- si seleziona $i \in Q : i = \arg \min_{j \in Q} \{d[j]\}$:
 - $Q = Q \setminus \{i\}$
 - $\forall (i, j) \in FS(i)$ se $d[i] + c_{ij} < d[j]$ allora $d[j] = d[i] + c_{ij}, P[j] = i$, se $j \notin Q, Q = Q \cup \{j\}$
- se $Q = \emptyset$ l'algoritmo termina, altrimenti si torna a 2

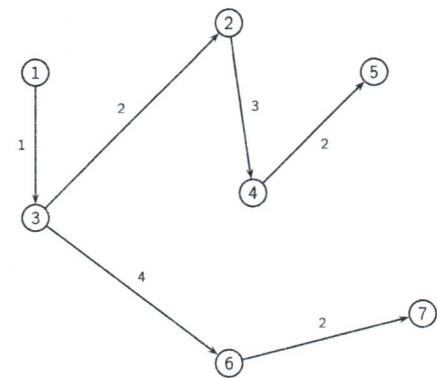
Esempio



Esempio

d[1],P[1]	0,1					
d[2],P[2]	M,1	4,1	3,3			
d[3],P[3]	M,1	1,1				
d[4],P[4]	M,1	M,1	M,1	6,2	6,2	
d[5],P[5]	M,1	M,1	10,3	10,3	10,3	8,4
d[6],P[6]	M,1	M,1	5,3	5,3		
d[7],P[7]	M,1	M,1	M,1	7,6	7,6	
Q	1	2,3	2,5,6	5,6,4	5,4,7	5,7
i	1	3	2	6	4	7
						5

Esempio



Algoritmo di Dijkstra - complessità

Schema dell'algoritmo

1. (inizializzazione)
 $P[i] = r, d[i] = \infty, \forall i \in N, P[r] = r, d[r] = 0, Q = \{r\}$
2. si seleziona $i \in Q : i = \arg \min_{j \in Q} \{d[j]\} \rightarrow O(|N|^2)$
 - 2.1 $Q = Q \setminus \{i\}$
 - 2.2 $\forall (i, j) \in FS(i)$ se $d[i] + c_{ij} < d[j]$ allora
 $d[j] = d[i] + c_{ij}, P[j] = i$
se $j \notin Q, Q = Q \cup \{j\} \rightarrow O(|A|)$
3. se $Q = \emptyset$ l'algoritmo termina, altrimenti si torna a 2

Caso generale, con cicli e $c_{ij} \leq 0$

Se il grafo è ciclico e con costi anche negativi, è possibile che

$$d[j] < d[i] + c_{ij}$$

anche se $d[i] > d[j]$.

Dijkstra non può essere applicato

Idea dell'algoritmo

Controllare la condizione di ottimalità $d[j] \leq d[i] + c_{ij}$ per ogni arco, e modificare il valore di $d[j]$, se necessario, finché non è soddisfatta da tutti gli archi.

Algoritmo di Bellman-Ford

Struttura dell'algoritmo

1. Ripetere finchè c'è stato un aggiornamento:
 - 1.1 $\forall (i, j) \in A$
 - se $d[i] + c_{ij} < d[j]$ allora $d[j] = d[i] + c_{ij}, P[j] = i$

Algoritmo di Bellman-Ford

Presenza di cicli negativi

In questo caso è possibile che nel grafo siano presenti cicli di lunghezza negativa

Percorrendo infinite volte il ciclo si riduce il valore della funzione obiettivo

Siamo interessati a rilevare la presenza di cicli di lunghezza negativa

Il problema di trovare il cammino minimo semplice è difficile

Algoritmo di Bellman-Ford

Come rilevare la presenza di cicli negativi

1. Ripetere finchè c'è stato un aggiornamento:
 - 1.1 $\forall (i, j) \in A$
 - se $d[i] + c_{ij} < d[j]$ allora $d[j] = d[i] + c_{ij}, P[j] = i$

Ogni iterazione in cui si controllano tutti gli archi è detta fase

Algoritmo di Bellman-Ford

Come rilevare i cicli negativi

Nella fase k si trovano i cammini minimi con al più k archi.
Un cammino semplice può contenere al più $|N| - 1$ archi.

Se nella fase $|N| - 1$ c'è ancora un aggiornamento allora esiste un cammino minimo che comprende $|N|$ archi \Rightarrow il grafo contiene un ciclo negativo.

Algoritmo di Bellman-Ford

Input Un grafo $G = (N, A)$, i costi associati a ogni arco e un nodo radice r

Output Il cammino minimo da r a ogni altro nodo o un ciclo negativo

Implementazione

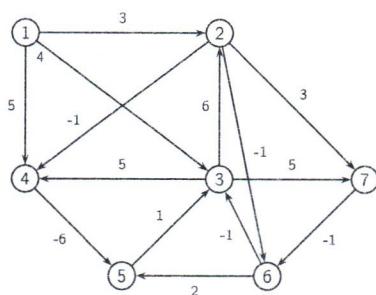
- Si controllano solo gli archi (i, j) per i quali si è modificata $d[i]$
- Per tenere conto dei nodi di cui si deve controllare la stessa uscente si usa una lista Q
- Se un nodo entra in Q $|N|$ volte si rileva la presenza di un ciclo negativo (parametro $k[i]$)

Algoritmo di Bellman-Ford

Schema dell'algoritmo

1. (inizializzazione) $\forall i \in N, d[i] = \infty, P[i] = r, k[i] = 0, d[r] = 0, P[r] = r, Q = \{r\}, k[r] = 1$
2. si seleziona $i \in Q, Q = Q \setminus \{i\}$
3. $\forall (i, j) \in FS(i)$
 - 3.1 se $d[i] + c_{ij} < d[j]$ allora
 - $d[j] = d[i] + c_{ij}, P[j] = i$
 - se $j \notin Q$ allora $Q = Q \cup \{j\}, k[j] = k[j] + 1$
 - se $k[j] = |N|$ è presente un ciclo negativo, l'algoritmo termina
4. se $Q = \emptyset$ l'algoritmo termina restituendo i cammini minimi
5. altrimenti torna a 2

Esempio



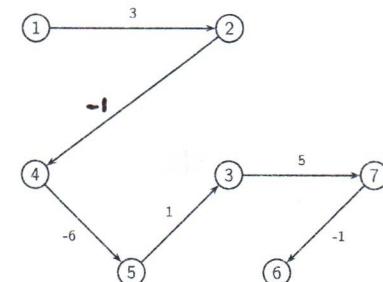
Esempio

$d[1], P[1], k[1]$	0,1,1	0,1,1	0,1,1	0,1,1	0,1,1
$d[2], P[2], k[2]$	3,1,1	3,1,1	3,1,1	3,1,1	3,1,1
$d[3], P[3], k[3]$	-1,6,2	-3,5,2	-3,5,2	-3,5,2	-3,5,2
$d[4], P[4], k[4]$	2,2,1	2,2,1	2,2,1	2,2,1	2,2,1
$d[5], P[5], k[5]$	-4,4,1	-4,4,1	-4,4,1	-4,4,1	-4,4,1
$d[6], P[6], k[6]$	2,2,1	2,2,1	2,2,1	1,7,2	1,7,2
$d[7], P[7], k[7]$	6,2,1	6,2,1	2,3,2	2,3,2	2,3,2
Q	5,3	3	7	6	
i	5	3	7	6	

Esempio

$d[1], P[1], k[1]$	0,1,1	0,1,1	0,1,1	0,1,1	0,1,1	0,1,1
$d[2], P[2], k[2]$	M,1,0	3,1,1	3,1,1	3,1,1	3,1,1	3,1,1
$d[3], P[3], k[3]$	M,1,0	4,1,1	4,1,1	4,1,1	4,1,1	-1,6,2
$d[4], P[4], k[4]$	M,1,0	5,1,1	2,2,1	2,2,1	2,2,1	* 2,2,1
$d[5], P[5], k[5]$	M,1,0	M,1,0	M,1,0	M,1,0	-4,4,1	-4,4,1
$d[6], P[6], k[6]$	M,1,0	M,1,0	2,2,1	2,2,1	2,2,1	2,2,1
$d[7], P[7], k[7]$	M,1,0	M,1,0	6,2,1	6,2,1	6,2,1	6,2,1
Q	1	2,3,4	3,4,6,7	4,6,7	6,7,5	7,5,3
i	1	2	3	4	6	7

Esempio



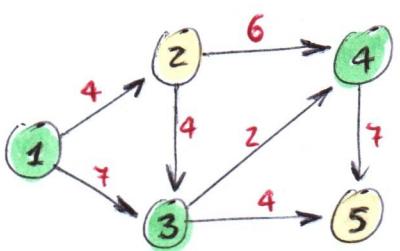
Osservazioni

Complessità computazionale

Nel caso peggiore esegue tutte le fasi controllando per ciascuna tutti gli archi

$\rightarrow O(|N||A|)$

$$N_S = \{1, 3, 4\} \quad N_T = \{2, 5\}$$



Per la capacità di taglio mi intervengono gli archi:

$$\text{Diagram: } \text{green circle} \rightarrow \text{yellow circle} : \begin{array}{ll} (1, 2) & u_{12} = 4 \\ (3, 5) & u_{35} = 4 \\ (4, 5) & u_{45} = 7 \end{array}$$

$$U = 15$$

Flusso massimo = trovare ⁱⁿ una rete di flusso con una rete sorgente ed un solo pozzo un flusso ammissibile che ne massimizza

7. Grafi - Flusso massimo

Fondamenti di Ricerca Operativa - A.A. 2018/2019

Giuliana Carello
DEIB, Politecnico di Milano

Flusso massimo

Definizione

Dato un grafo orientato $G = (N, A)$, in cui ad ogni arco $(i, j) \in A$ è associata una capacità u_{ij} , e una coppia di nodi sorgente $s \in N$ e destinazione $t \in N$, trovare la massima quantità di flusso che può essere inviata da s a t senza violare i limiti imposti dalle capacità su ogni arco.

Tagli e capacità

Modello

$$\begin{aligned} \max v \\ \sum_{(i,j) \in A} x_{ij} - \sum_{(j,i) \in A} x_{ji} &= \begin{cases} v, & i = s \\ -v, & i = t \\ 0, & i \neq s, t \end{cases}, \forall i \in N \\ x_{ij} &\leq u_{ij}, \quad \forall (i,j) \in A \\ x_{ij} &\geq 0, \quad \forall (i,j) \in A \end{aligned}$$

Def.: Taglio di un grafo

Dato un grafo $G = (N, A)$, si definisce taglio una partizione dell'insieme dei nodi

$$N_T \subseteq N, N_S \subseteq N : N_T \cap N_S = \emptyset, N_T \cup N_S = N.$$

(non devono essere collegati)

Tagli e capacità

Def.: Archi (o lati) del taglio

Si dicono archi del taglio gli archi che collegano i due sottoinsiemi della partizione:

$$\{(i, j) : i \in N_S, j \in N_T\} \cup \{(i, j) : i \in N_T, j \in N_S\}$$

- Archi in avanti $A^+(N_S, N_T) = \{(i, j) : i \in N_S, j \in N_T\}$
- Archi all'indietro $A^-(N_S, N_T) = \{(i, j) : i \in N_T, j \in N_S\}$

Def.: Capacità del taglio

Si definisce capacità di un taglio N_S, N_T la somma delle capacità degli archi in avanti del taglio:

$$U(N_S, N_T) = \sum_{(i,j) \in A^+(N_S, N_T)} u_{ij}$$

Capacità del taglio

Taglio:

$$N_S = \{1, 3, 4\}, N_T = \{2, 5\}$$

Archi del taglio:

- Archi in avanti: $A^+(N_S, N_T) = \{(1, 2), (3, 5), (4, 5)\}$
- Archi all'indietro: $A^-(N_S, N_T) = \{(2, 3), (2, 4)\}$

Capacità del taglio:

$$u_{12} + u_{35} + u_{45} = 15$$

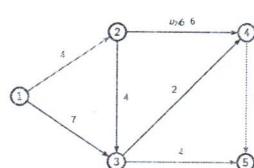


Figura: Capacità del taglio

Flusso dal nodo $s = 1$ al nodo $t = 5$.

Flusso attraverso il taglio

Flusso attraverso il taglio

Il flusso attraverso il taglio è pari alla somma dei flussi sugli archi in avanti diminuito del flusso sugli archi all'indietro:

$$X(N_S, N_T) = \sum_{(i,j) \in A^+(N_S, N_T)} x_{ij} - \sum_{(i,j) \in A^-(N_S, N_T)} x_{ij}$$

Il flusso da s a t è pari al flusso che attraversa un generico taglio

$$v = X(N_S, N_T)$$

Esempio

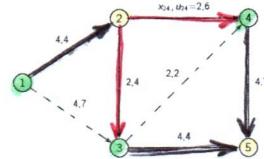
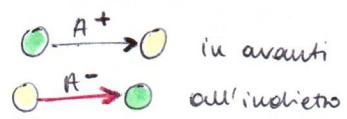


Figura: Flusso del taglio

$$\begin{aligned} \text{Flusso} &= x_{12} + x_{35} + x_{45} - x_{23} - x_{24} = 12 - 4 = 8 \\ v &= x_{12} + x_{13} = x_{45} + x_{35} = 8 \end{aligned}$$

Relazione tra flusso e capacità di un taglio

$$X(N_S, N_T) = \sum_{(i,j) \in A^+(N_S, N_T)} x_{ij} - \sum_{(i,j) \in A^-(N_S, N_T)} x_{ij}$$

$$\sum_{(i,j) \in A^+(N_S, N_T)} x_{ij} \leq \sum_{(i,j) \in A^+(N_S, N_T)} u_{ij}$$

$$\sum_{(i,j) \in A^-(N_S, N_T)} x_{ij} \geq 0$$

Quindi

$$X(N_S, N_T) = \sum_{(i,j) \in A^+(N_S, N_T)} x_{ij} - \sum_{(i,j) \in A^-(N_S, N_T)} x_{ij} \leq \sum_{(i,j) \in A^+(N_S, N_T)} u_{ij}$$

Relazione tra flusso e capacità di un taglio

Osservazione

Il flusso da s a t è sempre minore o uguale alla capacità di un generico taglio

$$X(N_S, N_T) \leq U(N_S, N_T)$$

Condizione di ottimalità

Se per un taglio N_S, N_T si ha $X(N_S, N_T) = U(N_S, N_T)$ il flusso è massimo (ottimo).

Problema duale

Problema duale

$$\begin{aligned} \max v \\ \sum_{(i,j) \in A} x_{ij} - \sum_{(j,i) \in A} x_{ji} - \left\{ \begin{array}{ll} v, i = s \\ -v, i = t \\ 0, i \neq s, t \end{array} \right. = 0, \forall i \in N \quad (\sigma_i) \\ x_{ij} \leq u_{ij}, \quad \forall (i,j) \in A \quad (\pi_{ij}) \\ x_{ij} \geq 0, \quad \forall (i,j) \in A \end{aligned}$$

$$\begin{aligned} \min \sum_{(i,j) \in A} u_{ij} \pi_{ij} \\ \sigma_i - \sigma_j + \pi_{ij} \geq 0, \quad \forall (i,j) \in A \quad (x_{ij}) \\ -\sigma_s + \sigma_t \geq 1 \quad (v) \\ \pi_{ij} \geq 0, \quad \forall (i,j) \in A \end{aligned}$$

Soluzione del duale

Teorema

Ogni taglio (N_S, N_T) determina una soluzione ammissibile del duale con valore della funzione obiettivo $U(N_S, N_T)$:

$$\pi_{ij} = \begin{cases} 1, (i,j) \in A^+(N_S, N_T) \\ 0, \text{ altrimenti} \end{cases}$$

$$\sigma_i = \begin{cases} 0, i \in N_S \\ 1, i \in N_T \end{cases}$$

Teorema max flow - min cut

Teorema (max flow - min cut)

Il valore v di ogni flusso da s a t non è superiore alla capacità di ogni taglio (N_S, N_T) . Il valore del flusso massimo è pari alla capacità del taglio di capacità minima. Una soluzione del primale \bar{x}_{ij} è ottima se

- $\bar{x}_{ij} = 0, \forall (i,j) \in A^-(N_S, N_T)$
- $\bar{x}_{ij} = u_{ij}, \forall (i,j) \in A^+(N_S, N_T)$

Idea dell'algoritmo

- Partire da una soluzione ammissibile (ne esiste sempre una?)
- Modificare il flusso mantenendolo ammissibile → *grafo residuale o incrementale*
- In corrispondenza della soluzione ottima, costruire un taglio di capacità minima

Grafo residuale

Grafo residuale
Il grafo residuale rappresenta le variazioni che un flusso ammissibile \bar{x} può subire rimanendo ammissibile

$$G(N, A) \Rightarrow G_R(N, A(\bar{x}))$$

Quali sono le variazioni possibili?

- se l'arco non è *saturo*, cioè se $\bar{x}_{ij} < u_{ij}$, il flusso può essere aumentato
- se l'arco non è *scarico*, cioè se $\bar{x}_{ij} > 0$, il flusso può essere diminuito

Grafo residuale

Archi del grafo residuale

Rappresentiamo le variazioni con gli archi del grafo residuale:

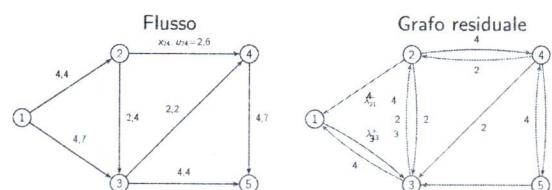
- se $\bar{x}_{ij} < u_{ij}$ si aggiunge l'arco (i, j) : $(i, j) \in A(\bar{x})$
- se $\bar{x}_{ij} > 0$ si aggiunge l'arco (j, i) : $(j, i) \in A(\bar{x})$
- gli archi del primo tipo rappresentano la possibilità di aumentare il flusso: sono archi *concordi* rispetto all'arco del grafo originario $A_C(\bar{x})$
- gli archi del secondo tipo rappresentano la possibilità di diminuire il flusso: sono archi *discordi* rispetto all'arco del grafo originario $A_D(\bar{x})$

Grafo residuale

Entità delle possibili variazioni

- il massimo aumento consentito su un arco non saturo è $u_{ij} - \bar{x}_{ij}$: si associa ad ogni arco $(i, j) \in A_C(\bar{x})$ l'etichetta $\lambda_{ij}^+ = u_{ij} - \bar{x}_{ij}$
- la massima diminuzione consentita su un arco non scarico è \bar{x}_{ij} : si associa ad ogni arco $(i, j) \in A_D(\bar{x})$ l'etichetta $\lambda_{ij}^- = \bar{x}_{ij}$

Esempio



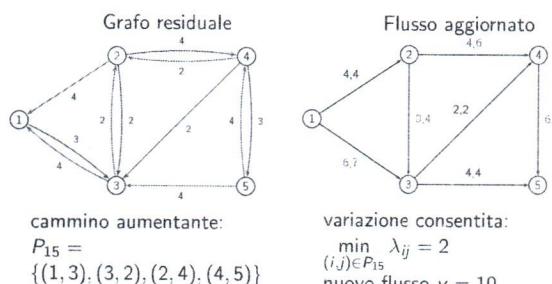
Ricerca di una soluzione migliorante

Gli archi del grafo residuale descrivono quali variazioni sono ammissibili su ogni arco. Come scegliere quali delle possibili variazioni devono essere effettuate, mantenendo anche il corretto bilanciamento ai nodi?

Cammino aumentante

Si cerca sul grafo residuale un cammino da s a t (algoritmo di visita). Se tale cammino, detto *cammino aumentante*, esiste la soluzione corrente non è ottima. Si può trovare una soluzione migliore variando il flusso sugli archi secondo il cammino aumentante trovato.

Esempio



Variazione del flusso

Come variare il flusso

- Scelta della massima variazione:

$$\theta = \min_{(i,j) \in p} \lambda_{ij} = \min_{(i,j) \in p} \{(u_{ij} - \bar{x}_{ij}) : (i, j) \in A_C(\bar{x}), (\bar{x}_{ji}) : (i, j) \in A_D(\bar{x})\}$$

- aggiornamento dei flussi

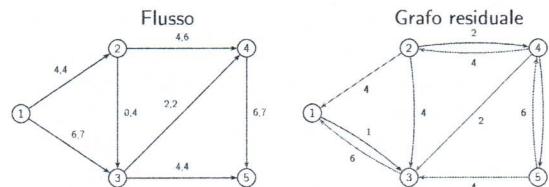
$$x' = \begin{cases} \bar{x}_{ij}, & \text{se } (i, j) \notin p \\ \bar{x}_{ij} + \theta, & \text{se } (i, j) \in p \cap A_C(\bar{x}) \\ \bar{x}_{ji} - \theta, & \text{se } (i, j) \in p \cap A_D(\bar{x}) \end{cases}$$

- aggiornamento del flusso massimo: $v = v + \theta$

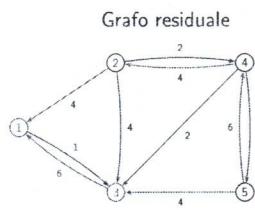
Non esiste una soluzione migliorante

Esempio

Non esiste un cammino aumentante
Se non esiste un cammino aumentante la soluzione corrente è ottima. L'algoritmo di visita, applicato al grafo residuale, restituisce un taglio. Il taglio trovato è un taglio di capacità minima. L'algoritmo che si basa sui cammini aumentanti dimostra così l'ottimalità della soluzione trovata.



Esempio



Taglio
 Il taglio è di capacità minima è
 $N_S = \{1, 3\}$, $N_T = \{2, 4, 5\}$
 $U(N_S, N_T) = 10$
 $\bar{x}_{ij} = 0, \forall (i, j) \in A^-(N_S, N_T)$
 $\bar{x}_{ij} = u_{ij}, \forall (i, j) \in A^+(N_S, N_T)$

Osservazione

Caratteristica della soluzione ottima

$\bar{x}_{ij} = 0, \forall (i, j) \in A^-(N_S, N_T)$: in caso contrario ci sarebbe un arco da j a i sul grafo residuale, e i sarebbe raggiungibile nel grafo residuale

$\bar{x}_{ij} = u_{ij}, \forall (i, j) \in A^+(N_S, N_T)$: in caso contrario ci sarebbe un arco da i a j sul grafo residuale, e j sarebbe raggiungibile nel grafo residuale

$$X(N_S, N_T) = \sum_{(i,j) \in A^+(N_S, N_T)} x_{ij} - \sum_{(i,j) \in A^-(N_S, N_T)} x_{ij} \\ = \sum_{(i,j) \in A^+(N_S, N_T)} u_{ij}$$

Algoritmo dei cammini aumentanti (Ford-Fulkerson 1957)

Input Il grafo $G = (N, A)$, la coppia di nodi sorgente e destinazione

$s, t \in N$, la capacità associata a ogni arco u_{ij}

Output Il flusso massimo, i flussi sugli archi, un taglio di capacità minima

Algoritmo dei cammini aumentanti (Ford-Fulkerson 1957)

Schema dell'algoritmo

1. Si costruisce il grafo residuale G_R (algoritmo di visita)
2. Si cerca un cammino aumentante sul grafo residuale
3. Se il cammino aumentante esiste si aggiorna il flusso e si torna al passo 1
4. Altrimenti l'algoritmo termina

Complessità

Complessità

Ad ogni iterazione posso aumentare di una unità \Rightarrow il numero di iterazioni dipende dal valore dell'ottimo \Rightarrow dipende dalle capacità degli archi, non dalle dimensioni del grafo

Algoritmo pseudopolinomiale

Complessità

Complessità

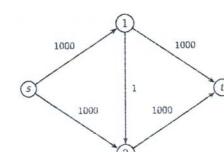


Figura: Caso patologico

Versione polinomiale

Scegliendo sempre il cammino aumentante con il minor numero di archi si garantisce la polinomialità dell'algoritmo

8. Programmazione Lineare Intera

Metodo dei piani di taglio

Fondamenti di Ricerca Operativa - A.A. 2018/2019

Giuliana Carello
DEIB, Politecnico di Milano

Problemi a variabili intere

- ▶ Problemi a variabili intere o binarie \Rightarrow il simplex non garantisce di trovare una soluzione con variabili intere o binarie
- ▶ Problemi per cui non esistono algoritmi polinomiali (come ad es. per cammini minimi o flusso massimo)

Problemi *difficili*

Esempio

Metodi esatti

Strategie

- ▶ Metodi esatti: garantiscono di trovare l'ottimo, ma possono richiedere tempi elevati
- ▶ Metodi euristici: impiegano tempo limitato, ma non garantiscono di trovare l'ottimo

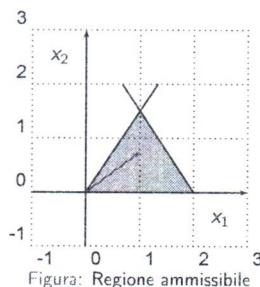
Metodi esatti

- ▶ Piani di taglio: si basa sul raffinamento della formulazione
- ▶ Branch and bound: si basa sull'enumerazione implicita delle soluzioni

$$\begin{aligned} \max & 4x_1 + 3x_2 \\ 3x_1 + 2x_2 & \leq 6 \\ -3x_1 + 2x_2 & \leq 0 \\ x_1, x_2 & \geq 0, \quad x_1, x_2 \in \mathbb{Z} \end{aligned}$$

Esempio

Esempio



Osservazioni

- ▶ Ottimo del problema a variabili continue: $\left(1, \frac{3}{2}\right), z = \frac{17}{2}$
- ▶ Punto a variabili intere più vicino all'ottimo continuo $(1, 1), z = 7$
- ▶ Ottimo a variabili intere $(2, 0), z = 8$

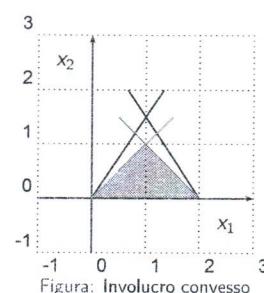
Esempio

Esempio

Esempio

Formulazione alternativa

$$\begin{aligned} \max & 4x_1 + 3x_2 \\ x_1 - x_2 & \geq 0 \\ x_1 + x_2 & \leq 2 \\ x_1, x_2 & \geq 0, \quad x_1, x_2 \in \mathbb{Z} \end{aligned}$$



Involucro convesso

Involucro convesso

Dato un insieme di punti S , l'involucro convesso ($\text{conv}(S)$) è il più piccolo poliedro che contiene S .

- La regione ammissibile del problema a variabili continue cambia, ma i punti a coordinate intere sono gli stessi
- Tutti i vertici del poliedro sono interi \Rightarrow risolvendo il problema a variabili continue con il simplex si trova l'ottimo intero
- La formulazione *ideale* descrive l'*involucro convesso* delle soluzioni a variabili intere

Involucro convesso

$$\min cx$$

$$\begin{array}{l} Ax \leq b \\ x \geq 0, \quad x \in \mathbb{Z}^n \end{array}$$

$$\min cx$$

$$x \in \text{conv}(\{Ax \leq b, \quad x \geq 0, x \in \mathbb{Z}^n\})$$

Se si risolvesse il problema a variabili continue su $\text{conv}(S)$, si troverebbe l'ottimo intero.

Non è sempre facile descrivere $\text{conv}(S)$: cerchiamo strategia alternativa

Metodo dei piani di taglio

Idea del metodo

- Si cerca una formulazione che descriva bene la regione ammissibile vicino all'ottimo intero
- Si risolve il problema con variabili continue, il problema rilassato o rilassamento continuo P_R :

Problema originale P

$$\begin{array}{ll} \min cx \\ Ax \leq b \\ x \geq 0, \quad x \in \mathbb{Z}^n \end{array}$$

Problema rilassato P_R

$$\begin{array}{ll} \min cx \\ Ax \leq b \\ x \geq 0 \end{array}$$

Se l'ottimo non è a variabili intere si aggiungono dei vincoli per eliminarlo dalla regione ammissibile

Che vincoli si aggiungono?

Si elimina il
vincolo di
interetza

Piani di taglio

Def. Piano di taglio

Dato un problema in programmazione lineare intera

$$\min cx$$

$$\begin{array}{ll} Ax \leq b \\ x \geq 0, \quad x \in \mathbb{Z}^n \end{array}$$

e una soluzione \bar{x} del problema rilassato (non intera), un piano di taglio è una diseguaglianza valida $gx \leq \gamma$ tale che

$$g\bar{x} > \gamma$$

= butto fuori
dall'insieme delle sol.

ammissibili il pt. non intero
ottimo (non tocca gli interi (dis. valida))

= data una formulazione iniziale, aggiungere
iterativamente nuovi vincoli finché non si ha
una soluzione ottima intera

Come generare i tagli?

Schema del metodo dei piani di taglio

Input: Un problema a variabili intere (c, A, b)

Output: Una soluzione ottima intera

1. Si risolve il problema rilassato P_R ottenendo la soluzione \bar{x}

2. se \bar{x} è intera allora la procedura termina

3. altrimenti

3.1 Si genera un taglio per \bar{x}

3.2 Si aggiunge il taglio ai vincoli del problema e si torna al passo 1

Tagli di Gomory

Tagli di Gomory

Sono dati un problema P e il corrispondente problema rilassato P_R

Problema P

$$\min cx$$

$$\begin{array}{ll} Ax \leq b \\ x \geq 0, \quad x \in \mathbb{Z}^n \end{array}$$

Problema rilassato P_R

$$\min cx$$

$$\begin{array}{ll} Ax \leq b \\ x \geq 0 \end{array}$$

e una soluzione ottima di base di P_R , $\bar{x} = (x_B, x_N)$.

- I tagli devono essere generati in modo automatico
- Possiamo usare tagli specifici per un problema o tagli "generici"
- Tagli di Gomory, applicabili a un problema scritto in forma standard e a una soluzione ottima continua di base
- Basati sulle diseguaglianze di Chvátal: si ottengono diseguaglianze valide combinando linearmente i vincoli

Tagli di Gomory

Tagli di Gomory

I vincoli possono essere riscritti in modo che in ogni vincolo compaia una sola variabile di base:

$$x_B + A_B^{-1} A_N x_N = A_B^{-1} b$$

$$x_B + \bar{A} x_N = \bar{b}$$

dove $A_B^{-1} b = \bar{b}$ e $A_B^{-1} A_N = \bar{A}$. Se tutte le variabili in base sono intere (b è intero) allora la soluzione trovata è ottima anche per il problema intero.

Tagli di Gomory

Tagli di Gomory

Altrimenti esiste un indice h tale che \bar{x}_h è frazionaria. Supponiamo che x_h sia la t -esima variabile della base: $\bar{x}_h = \bar{b}_t$, frazionario.

Riscriviamo il vincolo associato alla t -esima variabile in base:

$$x_h + \sum_{j \in N} \bar{a}_{tj} x_j = \bar{b}_t,$$

dove N è l'insieme degli indici delle variabili fuori base.

Ricaviamo una diseguaglianza valida.

Tagli di Gomory

Tagli di Gomory

$$x_h + \sum_{j \in N} \lfloor \bar{a}_{tj} \rfloor x_j \leq x_h + \sum_{j \in N} \bar{a}_{tj} x_j = \bar{b}_t$$

quindi

$$x_h + \sum_{j \in N} \lfloor \bar{a}_{tj} \rfloor x_j \leq \bar{b}_t$$

Per una soluzione intera x_h è intera e $\sum_{j \in N} \lfloor \bar{a}_{tj} \rfloor x_j$ è intera, mentre \bar{b}_t è frazionario, quindi

$$x_h + \sum_{j \in N} \lfloor \bar{a}_{tj} \rfloor x_j \leq \lfloor \bar{b}_t \rfloor$$

Tagli di Gomory

Tagli di Gomory

Il vincolo

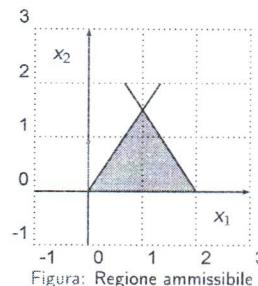
$$x_h + \sum_{j \in N} \lfloor \bar{a}_{tj} \rfloor x_j \leq \lfloor \bar{b}_t \rfloor$$

► è soddisfatto da tutte le soluzioni intere \Rightarrow è una diseguaglianza valida

► è violato dalla soluzione ottima continua in cui $\bar{x}_h = \bar{b}_t > \lfloor \bar{b}_t \rfloor$
 \Rightarrow è un taglio

Esempio

$$\begin{aligned} \max x_2 \\ 3x_1 + 2x_2 &\leq 6 \\ -3x_1 + 2x_2 &\leq 0 \\ x_1, x_2 \geq 0, x_1, x_2 \in \mathbb{Z} \end{aligned}$$



Problema rilassato

P_R

Soluzione di base ottima

$$x_1 = 1, x_2 = \frac{3}{2}, z = -\frac{3}{2}$$

$$\min -x_2$$

$$3x_1 + 2x_2 + x_3 = 6$$

$$-3x_1 + 2x_2 + x_4 = 0$$

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

$$x_B = (x_1, x_2), x_N = (x_3, x_4)$$

$$A_B = \begin{pmatrix} 3 & 2 \\ -3 & 2 \end{pmatrix}$$

$$A_B^{-1} = \begin{pmatrix} \frac{1}{6} & -\frac{1}{6} \\ \frac{1}{4} & \frac{1}{4} \end{pmatrix}$$

Taglio di Gomory

$$\begin{aligned} x_1 + \frac{1}{6}x_3 - \frac{1}{6}x_4 &= 1 \\ x_2 + \frac{1}{4}x_3 + \frac{1}{4}x_4 &= \frac{3}{2} \end{aligned}$$

Si ricava il taglio dal secondo vincolo:

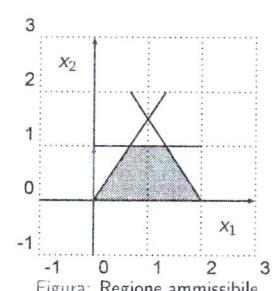
$$x_2 + \left\lfloor \frac{1}{4} \right\rfloor x_3 + \left\lfloor \frac{1}{4} \right\rfloor x_4 \leq \left\lfloor \frac{3}{2} \right\rfloor$$

da cui

$$x_2 \leq 1$$

Nuovo problema

$$\begin{aligned} \min -x_2 \\ 3x_1 + 2x_2 + x_3 &= 6 \\ -3x_1 + 2x_2 + x_4 &= 0 \\ x_2 + x_5 &= 1 \\ x_1, x_2, x_3, x_4, x_5 &\geq 0 \end{aligned}$$



La base $\{x_1, x_2, x_5\} = \{1, \frac{3}{2}, -\frac{1}{2}\}$ non è ammissibile, ma tutti i costi ridotti sono non negativi \Rightarrow applichiamo il simplex duale.

Nuovo ottimo continuo

Taglio di Gomory

Vincolo associato a x_1 :

Soluzione ottima continua: $x_1 = \frac{2}{3}, x_2 = 1, x_3 = 2$ Matrice di base:

$$A_B = \begin{pmatrix} 3 & 2 & 1 \\ -3 & 2 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \quad A_B^{-1} = \begin{pmatrix} 0 & -\frac{1}{3} & \frac{2}{3} \\ 0 & 0 & 1 \\ 1 & 1 & 4 \end{pmatrix}$$

$$x_1 - \frac{1}{3}x_4 + \frac{2}{3}x_5 = \frac{2}{3}$$

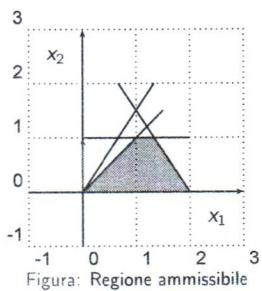
da cui si ricava il taglio di Gomory

$$x_1 - x_4 \leq 0$$

Ricordando che $x_4 = 3x_1 - 2x_2$ si ottiene

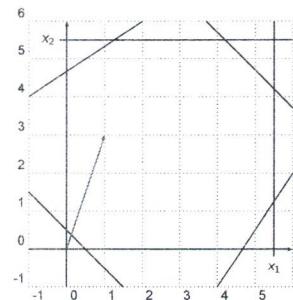
$$-x_1 + x_2 \leq 0$$

Nuovo problema



Nuovo ottimo continuo $x_1 = x_2 = 1$: è anche ottimo intero

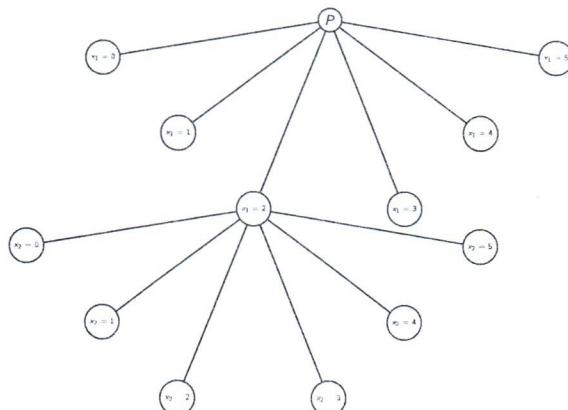
Enumerazione esplicita



- ▶ È possibile enumerare tutte le soluzioni a variabili intere
- ▶ Si usa un albero di ricerca, che permette di generare tutte le soluzioni

Figura: Regione ammissibile - funzione obiettivo $\max x_1 + 3x_2$

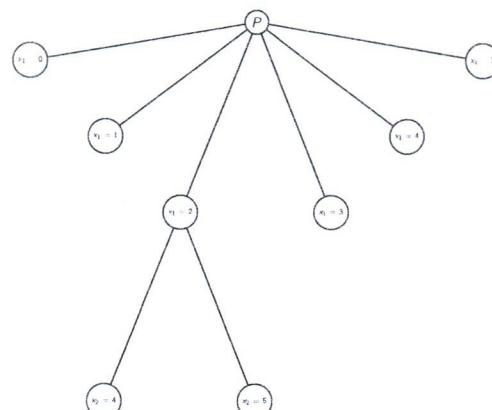
Albero di ricerca



Enumerazione implicita

- ▶ L'enumerazione esplicita può richiedere un grande sforzo computazionale
- ▶ Possiamo esaminare (e scartare) le soluzioni a gruppi, anziché singolarmente
- ⇒ Enumerazione implicita

Albero di ricerca potato



Branch and bound

Branch and bound

Il metodo del branch and bound si basa su questo principio:

- ▶ Si partiziona la regione ammissibile (*branching*), generando un albero di ricerca
- ▶ Si valuta ogni regione ottenuta, stimandone il valore ottimo (*bounding*)
- ▶ Si scartano le regioni che non contengono l'ottimo (*potatura o fathoming*)

Dobbiamo definire

- ▶ Come generare le regioni
- ▶ Come scartare le regioni non ottime (corrisponde a chiudere nodi dell'albero di ricerca) → criteri di fathoming
- ▶ Come stimare il valore dell'ottimo in una regione
- ▶ In che ordine visitare i nodi dell'albero di ricerca

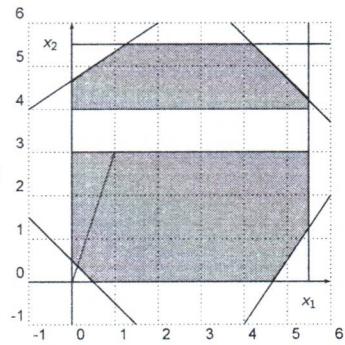


Branch

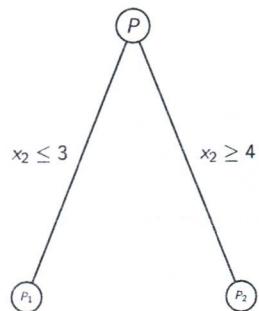
Come generare le regioni

- Le regioni si generano aggiungendo vincoli
- Il risultato deve essere una partizione della regione ammissibile intera, per non perdere nessuna soluzione

Branch

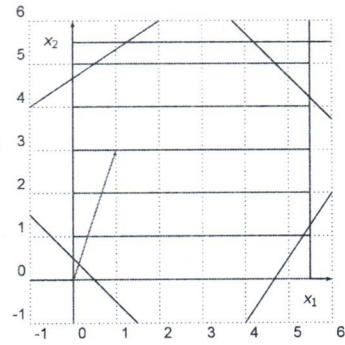


Albero corrispondente

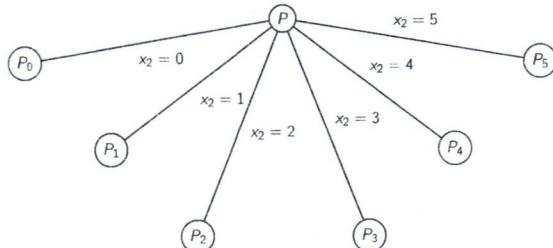


Branch

- Branch binario**
Ogni regione si suddivide in due regioni, aggiungendo due vincoli
Es. $x_2 \leq 3$ e $x_2 \geq 4$

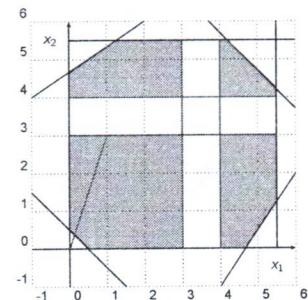


Albero corrispondente

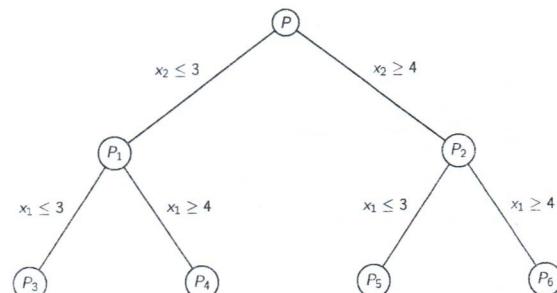


Branch

- Scendendo nell'albero le regioni si partizionano a loro volta
- Si aggiungono ulteriori vincoli ad esempio
 $x_2 \leq 3$ e $x_2 \geq 4$
 $x_1 \leq 3$ e $x_1 \geq 4$



Albero corrispondente



Branch

Osservazioni
I vincoli di branch non si applicano sempre ad una sola variabile fissandone il valore, possono essere vincoli di varia forma che coinvolgono più variabili

Stime

Stime della soluzione ottima

- ▶ Dobbiamo calcolare una stima del valore ottimo per ogni regione
- ▶ La stima deve essere sempre migliore, o uguale, all'ottimo

Stime

Funzione obiettivo di massimo

LB

UB

soluzione intera ammissibile \leq ottimo \leq stima

indichiamo la stima con **UB - upper bound** e una soluzione intera ammissibile con **LB - lower bound**

Funzione obiettivo di minimo

LB

UB

stima \leq ottimo \leq soluzione intera ammissibile

indichiamo la stima con **LB - lower bound** e una soluzione intera ammissibile con **UB - upper bound**

Stime

Per calcolare la stima usiamo un *rilassamento* ovvero risolviamo un altro problema che sia più veloce da risolvere del problema in esame e che dia informazioni sul valore ottimo

Rilassamento

Dato un problema $P = \max\{cx : x \in F\}$, il problema $P' = \max\{c'x : x \in F'\}$ è un rilassamento di P se

- ▶ $F \subseteq F'$
- ▶ $c'x \geq cx, \forall x \in F$

- ▶ Rilassamento per eliminazione di vincoli
- ▶ Rilassamento continuo: elimino i vincoli di interezza
 $x \in \{0, 1\} \Rightarrow 0 \leq x \leq 1$
 $x \in \mathbb{Z}_+ \Rightarrow x \geq 0$

Criteri di fathoming o potatura

1. Assenza di soluzione migliorante

Un nodo dell'albero di ricerca può essere chiuso, e di conseguenza la regione non ulteriormente investigata, se la stima dell'ottimo nella regione è peggiore della migliore soluzione intera nota (z^*).

Massimo

Si chiude il nodo P_i se $UB_i \leq z^*$

Minimo

Si chiude il nodo P_i se $LB_i \geq z^*$

Criteri di potatura

- ▶ Rilassamento continuo del sottoproblema P_4 $x_1 = 5.5, x_2 = 3, UB_4 = 14.5$
- ▶ Esiste una soluzione intera ammissibile di valore 16
- ▶ Il nodo P_4 può essere chiuso

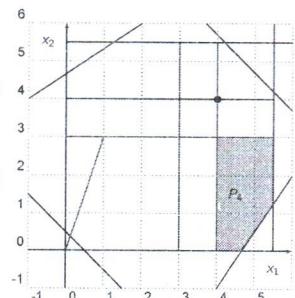


Figura: Regione ammissibile - funzione obiettivo $\max x_1 + 3x_2$

Criteri di potatura

2. Ottimo della regione

Se la stima coincide con una soluzione intera ammissibile, per la regione è stato trovato l'ottimo. Non è necessario esplorare ulteriormente la regione.

Criteri di potatura

- ▶ Rilassamento continuo del sottoproblema P_3 $x_1 = 3, x_2 = 3, UB_3 = 12$
- ▶ Abbiamo trovato la miglior soluzione intera della regione associata a P_3
- ▶ Il nodo P_3 può essere chiuso

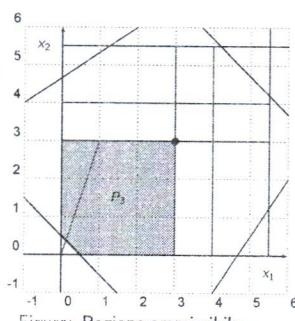


Figura: Regione ammissibile - funzione obiettivo $\max x_1 + 3x_2$

3. Non ammissibilità

Un nodo dell'albero di ricerca può essere chiuso, e di conseguenza la regione non ulteriormente investigata, se la regione risulta non ammissibile, a causa dei vincoli aggiuntivi.

Ad esempio $x_1 \geq 5, x_2 \geq 5$

Strategie di esplorazione

In che ordine esploriamo i nodi, partizionandone la regione ammissibile?

- Visita in profondità
- Visita del nodo più promettente
- Visita in ampiezza

Strategie di esplorazione

1. Depth first - esplorazione in profondità

- Per ogni nodo, esplora il primo nodo figlio generato dal nodo stesso, in modo ricorsivo:
 - trova rapidamente una soluzione intera ammissibile
 - occupa poca memoria
 - non tiene conto della qualità della soluzione trovata

Strategie di esplorazione

Schema del Branch & Bound

2. Best first - esplorazione del nodo migliore

- Esplora il nodo più promettente (ad es. con la stima migliore, o con la miglior combinazione di LB e UB)
 - occupa molto memoria
 - tiene conto della bontà delle soluzioni

Input Un problema a variabili intere (c, A, b)

Output Una soluzione ottima intera z^*

3. Breadth first - esplorazione in ampiezza

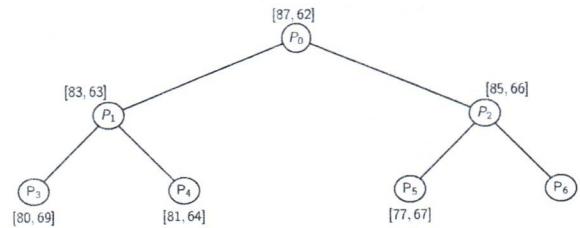
- Esplora tutti i nodi dello stesso livello prima di passare al livello successivo
 - non è usata, non garantisce buoni comportamenti

Schema del Branch & Bound

1. Se tutti i nodi sono stati esplorati e chiusi, il Branch & Bound termina e restituisce z^*
2. altrimenti seleziona un nodo da esplorare, calcola la stima b e controlla i criteri di potatura:
 - 2.1 se il problema non è ammissibile, chiude il nodo
 - 2.2 se la stima b è peggiore della migliore soluzione intera trovata z^* , chiude il nodo
 - 2.3 se la stima b corrisponde ad una soluzione ammissibile intera, chiude il nodo, e, se b è migliore di z^* , aggiorna z^*
3. se il nodo non è stato chiuso, genera i suoi figli facendo il branch e torna a 1

Esempio

Si consideri il seguente albero di branch and bound parziale per un problema di **minimizzazione**. Di tutti i nodi tranne il nodo 6 sono noti UB e LB.



Esempio

Esempio

1. In che intervallo è compreso l'ottimo?

La migliore soluzione z^* è 77. Rimangono aperti 4 nodi: P_3, P_4, P_5, P_6 . Da P_3 possiamo ottenere 69, da P_4 64, da P_5 67 e da P_6 66: dunque l'ottimo non può essere minore di 64. Perciò $64 \leq \text{ottimo} \leq 77$

2. Per quali valori di UB_6 e LB_6 è possibile chiudere P_3 ? Perché?
Per chiudere P_3 devo avere $z^* = UB_6 \leq 69$, ma $UB_6 \geq LB_6 \geq LB_2 = 66$. Dunque per $66 \leq UB_6 \leq 69$.

N.B. Scendendo la stima peggiora,
la soluzione intera ammissibile migliora

Esempio

3. Per quali valori di UB_6 e LB_6 è possibile chiudere tutti i nodi?
 Motivare la risposta.
 Non è possibile. Infatti per chiudere P_4 UB_6 dovrebbe essere minore o uguale a 64, ma questo non è possibile.

Rilassamento continuo con algoritmo ad-hoc

- Il rilassamento continuo è spesso usato per ottenere la stima, anche se non è l'unica possibilità
- Per qualche problema si può utilizzare un algoritmo ad hoc per calcolare l'ottimo del rilassamento continuo, ad esempio per il problema dello zaino

Esempio: zaino binario

Dati

- Un insieme di oggetti I
- peso $w_i \quad \forall i \in I$
- profitto $p_i \quad \forall i \in I$
- La capacità dello zaino B

Problema dello zaino binario

Selezionare un sottoinsieme di oggetti tali che:

- la somma dei loro pesi non superi la capacità dello zaino
- la somma dei loro profitti sia massima

Esempio: zaino binario

Modello

$$\max \sum_{i \in I} p_i x_i$$

$$\sum_{i \in I} w_i x_i \leq B$$

$$x_i \in \{0, 1\} \quad \forall i \in I$$

$$x_i = \begin{cases} 1 & \text{oggetto preso} \\ 0 & \text{oggetto non preso} \end{cases}$$

Esempio: zaino binario

Upper bound

Rilassamento continuo:

$$\max \sum_{i \in I} p_i x_i$$

$$\sum_{i \in I} w_i x_i \leq B$$

$$0 \leq x_i \leq 1 \quad \forall i \in I$$

Soluzione del rilassamento continuo \bar{x}

1. Ordinare gli oggetti per rapporto $\frac{p_i}{w_i}$ non crescente
2. Sia \tilde{B} la capacità residua
3. Sia i^* il prossimo oggetto secondo l'ordinamento
 - 3.1 Se $w_{i^*} \leq \tilde{B}$ allora $\bar{x}_{i^*} = 1$, $\tilde{B} = \tilde{B} - w_{i^*}$
 - 3.2 Se $w_{i^*} > \tilde{B}$ e $\tilde{B} > 0$ allora \bar{x}_{i^*} è frazionaria e $\bar{x}_{i^*} = \frac{\tilde{B}}{w_{i^*}}$, $\tilde{B} = 0$
 - 3.3 Se $\tilde{B} = 0$ allora $\bar{x}_i = 0, \forall i > i^*$

Branch and bound per zaino binario

- Solo una delle variabili è frazionaria
- si può applicare il branching sull'unica variabile frazionaria
- in alternativa si può applicare il branching alla variabile con rapporto $\frac{p_i}{w_i}$ più alto

Esercizio

Risolvere il seguente problema dello zaino 0/1 usando la tecnica del branch and bound.

$$\begin{aligned} \max \quad & 10x_1 + 12x_2 + 5x_3 + 7x_4 + 9x_5 \\ \text{s.t.} \quad & 5x_1 + 8x_2 + 6x_3 + 2x_4 + 7x_5 \leq 14 \\ & x_1, \dots, x_5 \in \{0, 1\} \end{aligned}$$

Per calcolare un lower bound si approssimi per difetto la variabile frazionaria nella soluzione calcolata dal rilassamento continuo.

Come primo passo costruiamo la lista delle variabili ordinate per rapporti profitto-peso decrescenti.

var.	x_4	x_1	x_2	x_5	x_3
p_i	7	10	12	9	5
w_i	2	5	8	7	6
p_i/w_i	3.5	2.0	1.5	1.2	0.8

Nodo 1

Al nodo 1 si è imposto $x_2 = 1$. La soluzione ottima del rilassamento continuo si ottiene per $x_4 = 1, x_1 = 1, x_2 = \frac{7}{8}, x_5 = x_3 = 0$. Quindi

$$UB_1 = LB_1 = 7 + 10 + 9 = 26 \Rightarrow \text{nuova } z^* = 26.$$

Nodo 4

Qui si impone $x_2 = x_1 = 1$. I due oggetti occupano uno spazio complessivo di $8 + 5 = 13$ unità. Nessuno dei rimanenti oggetti può essere inserito per intero nello spazio residuo di $14 - 13 = 2$ unità. Pertanto non si possono generare altre soluzioni ammissibili intere in questo nodo. Questa considerazione ci permette di chiuderlo.

Nodo 2

Al nodo 2 si è imposto $x_2 = 1$. La soluzione ottima del rilassamento continuo si ottiene per $x_2 = 1, x_4 = 1, x_1 = \frac{4}{5}, x_5 = x_3 = 0$, quindi

$$UB_2 = 12 + 7 + \left\lfloor 10 \cdot \frac{4}{5} \right\rfloor = 27, \quad LB_2 = 19 \Rightarrow \text{nuova } z^* = 19.$$

Nodo 5

Qui si è imposto $x_2 = 1, x_1 = 0$. La soluzione ottima del rilassamento continuo si ottiene per $x_2 = x_4 = 1, x_5 = \frac{4}{7}, x_1 = x_3 = 0$. Quindi

$$UB_5 = 12 + 7 + \left\lfloor 9 \cdot \frac{4}{7} \right\rfloor = 24 < z^*.$$

Il nodo viene chiuso.

La ricerca ha termine in quanto abbiamo chiuso tutti i nodi. L'ottimo risulta essere $z^* = 26$, con $x_1 = x_4 = x_5 = 1, x_2 = x_3 = 0$.

Albero di branch and bound

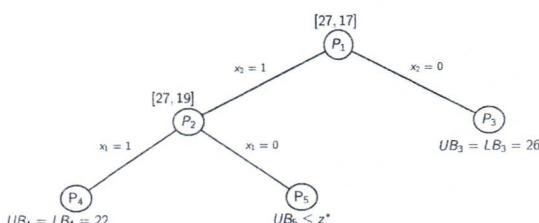


Figura: Albero di branch and bound

Esempio: zaino binario

Possiamo migliorare il calcolo del lower bound se non ci fermiamo una volta assegnato il valore 0 alla variabile frazionaria:

Lower bound x

Si controllano tutti gli oggetti secondo l'ordinamento e si fissano a 1 le loro variabili, se possibile:

1. Ordinare gli oggetti per rapporto $\frac{p_i}{w_i}$ non crescente
2. Sia \tilde{B} la capacità residua
3. Sia i^* il prossimo oggetto secondo l'ordinamento
 - 3.1 Se $w_{i^*} \leq \tilde{B}$ allora $x_{i^*} = 1, \tilde{B} = \tilde{B} - w_{i^*}$
 - 3.2 Se $w_{i^*} \geq \tilde{B}$ allora $x_{i^*} = 0$

- Come cambia l'albero con il lower bound migliorato?
- E facendo branch sulla variabile con rapporto più elevato?

Problemi facili e difficili

10. Cenni di complessità computazionale

Fondamenti di Ricerca Operativa - A.A. 2018/2019

Giuliana Carello
DEIB, Politecnico di Milano

Problemi facili e difficili

- ▶ Problemi facili: problemi risolubili con algoritmi polinomiali (albero di copertura, albero dei cammini minimi, ecc.)
- ▶ Problemi difficili: problemi per cui non conosciamo algoritmi polinomiali.

Lo studio della difficoltà dei problemi si basa sulla *Complessità Computazionale*:

- ▶ dimensione del problema
- ▶ complessità degli algoritmi

Dimensione dell'istanza

Istanza

Esempio specifico di un problema

Dimensione dell'istanza

Numero di bit necessari a codificare l'istanza

Cammini minimi

- ▶ Numero di nodi $N \rightarrow \lg(N)$
- ▶ Numero di archi $A \rightarrow \lg(A)$
- ▶ Elenco degli archi (coppie (i,j)) $\rightarrow 2A\lg(N)$
- ▶ Costi degli archi $\rightarrow \text{Alg}(C_{\max})$, dove C_{\max} è il costo più elevato

Complessità degli algoritmi

Complessità asintotica

$f(n), g(n)$: funzioni del numero di operazioni

Si dice che $f(n)$ è dell'ordine di $g(n)$ e si scrive $f(n) = O(g(n))$
se

$$\exists c > 0, \exists n_0 : f(n) \leq cg(n), n > n_0$$

- ▶ Algoritmi polinomiali $O(n^d)$
- ▶ Algoritmi esponenziali $O(2^n)$

Caso peggiore

La complessità degli algoritmi si calcola nel caso peggiore

Problemi facili e difficili

Problemi facili

- ▶ Albero di copertura, albero dei cammini minimi
- ▶ Programmazione Lineare?

Problemi difficili

- ▶ Zaino
- ▶ Problemi a variabili intere o binarie generici

Problema di riconoscimento

Analizziamo la complessità d un problema lavorando sulla versione di riconoscimento

La versione di *riconoscimento* (o *decisionale*) di un problema di ottimizzazione ha risposta SI/NO.

Versione di riconoscimento
Problema \mathcal{P}

$$\begin{array}{ll} \max cx \\ Ax & \leq b \\ x & \geq 0, x \in \mathbb{Z}^n \end{array}$$

Versione di riconoscimento

data una soglia k

$\exists \bar{x}$:

$$\begin{array}{l} A\bar{x} \leq b, \bar{x} \geq 0, x \in \mathbb{Z}^n \\ c\bar{x} \geq k? \end{array}$$

Esempi

Cammino minimo

Dato un grafo $G = (N, A)$, con costo c_{ij} per ogni arco, e due nodi $s, t \in N$, esiste un percorso da s a t di lunghezza inferiore od uguale a una soglia prefissata k ?

Zaino

Dato un insieme di oggetti I , a ciascuno dei quali è associato un profitto p_i ed un peso w_i , e la capienza massima dello zaino, B , esiste un insieme di oggetti, il cui peso complessivo non superi la capienza B , e di profitto complessivo non inferiore a una soglia prefissata k ?

Problemi polinomiali

Classe \mathcal{P}

Esiste un algoritmo che, per ogni istanza del problema di riconoscimento, dice in tempo polinomiale se la risposta è SI o NO.
Il problema di ottimizzazione si risolve all'ottimo in tempo polinomiale

Cammino minimo

Posso trovare il cammino più breve in tempo polinomiale e confrontarlo con la soglia k .

Problemi NP

Classe NP

Per ogni istanza, data una soluzione, è possibile verificare in tempo polinomiale se tale soluzione è una risposta SI al problema decisionale.

Più formalmente

Un problema \mathcal{P} è in NP se esiste un algoritmo \mathcal{A} tale che se l'istanza \mathcal{I} ha risposta SI allora esiste un certificato (una soluzione) $\gamma(\mathcal{I})$ e \mathcal{A} controlla in tempo polinomiale che $\gamma(\mathcal{I})$ sia una risposta SI.

Esempi

Cammino minimo

La soluzione è data da un insieme di archi. Per certificarla:

- ▶ Si verifica che la sequenza sia un cammino
- ▶ Si calcola la lunghezza del cammino, sommando tutti i pesi degli archi del cammino
- ▶ Si confronta il valore ottenuto con la soglia

Il controllo si esegue in un numero polinomiale di passi \Rightarrow il cammino minimo è un problema NP

Problemi NP-completi

- ▶ Alla classe NP-completi appartengono i problemi difficili
- ▶ Non conosciamo per essi un algoritmo di risoluzione polinomiale che garantisca di trovare la soluzione ottima per ogni istanza
- ▶ Per dare una definizione formale introduciamo il concetto di riducibilità

Classe NP-completi

Definizione

Un problema \mathcal{P} appartiene alla classe NP-completi (è NP-completo) se

- i) $\mathcal{P} \in NP$
- ii) $\mathcal{P}_1 \leq \mathcal{P}, \forall \mathcal{P}_1 \in NP$

Ogni problema in NP si può ridurre a \mathcal{P} .
 \mathcal{P} è almeno tanto difficile quanto ogni altro problema in NP.

Problemi NP

Nota bene!

- ▶ NP significa *Non-deterministico polinomiale*
- ▶ $P \subseteq NP$
- ▶ Se sia vero che $P \subset NP$ oppure $P = NP$ è un grande problema aperto

Esempi

Zaino

La soluzione è data dal valore delle variabili x_i associate ad ogni oggetto. Per certificarla:

- ▶ Si calcola il peso complessivo W . Scorro tutti gli oggetti, se $x_i = 1$ allora $W = W + w_i$
- ▶ Si verifica che $W \leq B$
- ▶ Si calcola il profitto complessivo P . Scorro tutti gli oggetti, se $x_i = 1$ allora $P = P + p_i$
- ▶ Si confronta P con la soglia

Il controllo si esegue in un numero polinomiale di passi \Rightarrow il problema dello zaino è un problema NP

Concetto di riduzione

Definizione

Dati due problemi \mathcal{P}_1 e \mathcal{P}_2 , si dice che \mathcal{P}_1 si riduce in tempo polinomiale a \mathcal{P}_2 (e si scrive $\mathcal{P}_1 \leq \mathcal{P}_2$) se per ogni istanza \mathcal{I}_1 di \mathcal{P}_1 possiamo costruire in tempo polinomiale un'istanza \mathcal{I}_2 di \mathcal{P}_2 tale che risolta \mathcal{I}_2 possiamo ricavare dalla sua soluzione una soluzione di \mathcal{I}_1 in tempo polinomiale

Se esiste un algoritmo \mathcal{A}_2 per risolvere \mathcal{P}_2 è possibile creare un algoritmo \mathcal{A}_1 per risolvere \mathcal{P}_1 che usa \mathcal{A}_2 . Se \mathcal{A}_2 è polinomiale, anche \mathcal{A}_1 lo è.

- ▶ se $\mathcal{P}_1 \leq \mathcal{P}_2$ allora \mathcal{P}_2 è almeno tanto difficile quanto \mathcal{P}_1
- ▶ se $\mathcal{P}_1 \leq \mathcal{P}_2$ e $\mathcal{P}_2 \in P$ allora $\mathcal{P}_1 \in P$

Classe NP-completi

Come si dimostra che un problema è NP-completo

- ▶ Si dimostra che $\mathcal{P} \in NP$
- ▶ Si dimostra che $\mathcal{P}_1 \leq \mathcal{P}$, con \mathcal{P}_1 NP-completo

La riducibilità è transitiva:

$$\mathcal{P}' \leq \mathcal{P}_1 \text{ e } \mathcal{P}_1 \leq \mathcal{P} \Rightarrow \mathcal{P}' \leq \mathcal{P}$$

Se \mathcal{P}_1 è NP-completo, $\mathcal{P}' \leq \mathcal{P}_1, \forall \mathcal{P}' \in NP$. Poiché la riducibilità è transitiva, $\mathcal{P}' \leq \mathcal{P}, \forall \mathcal{P}' \in NP$.

Dunque \mathcal{P} è NP-completo.

Problema della Soddisfabilità

Il primo problema che è stato dimostrato appartenere alla classe degli *NP*-completi è il *problema della Soddisfabilità* (Cook, 1971)

Soddisfabilità

Date m clausole booleane C_1, C_2, \dots, C_m , ciascuna del tipo $y_1 \vee y_2 \vee \bar{y}_3$, esiste un valore di verità da assegnare alle variabili booleane che renda tutte la clausole vere?

Esempio

Un problema di PLI è *NP*-completo

La versione di riconoscimento di un generico problema di Programmazione Lineare Intera

$$\begin{array}{ll} \exists \bar{x}: & \\ A\bar{x} & \leq b \\ \bar{x} & \geq 0, x \in \mathbb{Z}^n \\ c\bar{x} & \leq k \end{array}$$

è *NP*-completo.

Esempio

Esempio

Un problema di PLI è *NP*

Data una soluzione \bar{x} , per verificare che sia un certificato SI bisogna:

- ▶ Controllare l'ammissibilità di tutti i vincoli
- ▶ Calcolare il valore della funzione obiettivo e confrontarlo con k

Ogni problema *NP* è riducibile a PLI

Si dimostra riducendo il problema della Soddisfabilità a PLI.

$SAT \propto PLI$

Esempio

$SAT \propto PLI$

Problema SAT	Problema PLI
n variabili booleane y_i	n variabili binarie x_i
m clausole	m vincoli
$y_1 \vee y_2 \vee \bar{y}_3$	$x_1 + x_2 + (1 - x_3) \geq 1$
per garantire la verità della clausola	$x_1 + x_2 + (1 - x_3) \geq 1$
Se il problema PLI ha soluzione allora esiste una soluzione del problema SAT.	
La costruzione avviene in tempo polinomiale.	

Appendice A

Algoritmi e complessità

In questa appendice vogliamo brevemente richiamare alcuni concetti fondamentali della teoria della complessità computazionale, utili per meglio comprendere la diversa “difficoltà” della soluzione dei problemi di ottimizzazione.

A.1 Modelli computazionali

Una volta che un problema P sia stato formulato, deve essere risolto: siamo quindi interessati alla messa a punto di strumenti di calcolo che, data una qualsiasi istanza p , siano in grado di fornirne una soluzione in un tempo finito. Tali strumenti di calcolo si chiamano *algoritmi*. Un algoritmo che risolve P può essere definito come una sequenza finita di istruzioni che, applicata ad una qualsiasi istanza p di P , si arresta dopo un numero finito di passi (ovvero di computazioni elementari), fornendo una soluzione di p oppure indicando che p non ha soluzioni ammissibili. Per poter studiare gli algoritmi dal punto di vista della loro efficienza, o complessità computazionale, è necessario definire un modello computazionale: classici modelli computazionali sono la Macchina di Turing (storicamente il primo proposto), la R.A.M. (Random Access Machine), la Macchina a Registri (MR), etc.

La Macchina a Registri è un buon compromesso tra semplicità e versatilità: una MR consiste di un numero (finito ma non limitato) di registri, ciascuno dei quali può contenere un singolo numero intero, e di un programma, ossia di una sequenza finita di istruzioni del tipo

- incrementa il registro k e salta all’istruzione j ;
- decrementa il registro k e salta all’istruzione j ;
- se il registro k contiene 0 salta all’istruzione j , altrimenti salta all’istruzione h .

Si tratta di una macchina *sequenziale e deterministica*, poiché il comportamento futuro della macchina è univocamente determinato dalla sua configurazione presente. Una MR è un buon modello astratto di un calcolatore elettronico, ed è quindi in grado di compiere tutte le computazioni possibili in un qualunque sistema di calcolo attualmente noto (se si eccettuano i computer quantistici, la cui implementabilità è comunque ancora da dimostrare). D’altra parte, si può dimostrare che la classe delle funzioni computabili da una MR è equivalente alla classe delle funzioni computabili da una Macchina di Turing, il che, secondo la *Tesi di Church*, implica che le MR siano presumibilmente in grado di calcolare qualsiasi funzione effettivamente computabile con procedimenti algoritmici.

A.2 Misure di complessità

Dato un problema P , una sua istanza p , e un algoritmo A che risolve P , indichiamo con costo (o complessità) di A applicato a p una misura delle risorse utilizzate dalle computazioni che A esegue su una macchina MR per determinare la soluzione di p . Le risorse, in principio, sono di due tipi, *memoria occupata e tempo di calcolo*: nell’ipotesi che tutte le operazioni elementari abbiano la stessa durata, il tempo di calcolo può essere espresso come numero di operazioni elementari effettuate dall’algoritmo. Poiché molto spesso la risorsa più critica è il tempo di calcolo, nel seguito useremo soprattutto questa come misura della complessità degli algoritmi.

Dato un algoritmo, è opportuno disporre di una misura di complessità che consenta una valutazione sintetica della sua bontà ed eventualmente un suo agevole confronto con algoritmi alternativi. Conoscere la complessità di A per ognuna delle istanze di P non è possibile (l’insieme delle istanze di un problema è normalmente infinito), né sarebbe di utilità pratica: si cerca allora di esprimere la complessità come una funzione $g(n)$ della dimensione, n , dell’istanza cui viene applicato l’algoritmo. Poiché, per ogni dimensione, si hanno in generale molte istanze di quella dimensione, si sceglie $g(n)$ come il costo necessario per risolvere la più difficile tra le istanze di dimensione n : si parla allora di *complessità nel caso peggiore*.

Bisogna naturalmente definire in modo preciso il significato di dimensione di una istanza: chiameremo dimensione di p una misura del numero di bit necessari per rappresentare, con una codifica “ragionevolmente” compatta, i dati che definiscono p , cioè una misura della lunghezza del suo input. Per esempio, in un grafo con n nodi e m archi i nodi possono essere rappresentati dagli interi tra 1 ed n e gli archi per mezzo di una lista contenente m coppie di interi (l’arco che

collega i nodi i e j è rappresentato dalla coppia (i, j)): trascurando le costanti moltiplicative, potremo allora assumere come misura della dimensione della codifica del grafo, al variare del numero dei nodi e degli archi, la funzione $m \log n$, dato che interi positivi e non superiori ad n possono essere rappresentati con $\log n$ bit¹. Nel seguito, per semplicità, oltre alle costanti moltiplicative trascureremo anche le funzioni sublineari, come la funzione logaritmo; diremo allora che m è la lunghezza dell'input per un grafo con m archi. Nelle ipotesi fatte, la misura della lunghezza dell'input non varia se usiamo una codifica in base $b > 2$: se invece si usasse una codifica unaria, la lunghezza dell'input nell'esempio in questione diventerebbe nm , aumentando considerevolmente.

A questo punto la funzione $g(n)$, introdotta precedentemente, risulta definita in modo sufficientemente rigoroso: in pratica essa continua però ad essere di difficile uso come misura della complessità, dato che risulta difficile, se non praticamente impossibile, la valutazione di $g(n)$ per ogni dato valore di n . Questo problema si risolve sostituendo alla $g(n)$ il suo ordine di grandezza: si parlerà allora di *complessità asintotica*. Data una funzione $g(x)$, diremo che:

1. $g(x) \in O(f(x))$ se esistono due costanti c_1 e c_2 per cui, per ogni x , è $g(x) \leq c_1 f(x) + c_2$;
2. $g(x) \in \Omega(f(x))$ se $f(x) \in O(g(x))$;
3. $g(x) \in \Theta(f(x))$ se $g(x)$ è allo stesso tempo $O(f(x))$ e $\Omega(f(x))$.

Sia $g(x)$ il numero di operazioni elementari che vengono effettuate dall'algoritmo A applicato alla più difficile istanza, tra tutte quelle che hanno lunghezza di input x , di un dato problema P : diremo che la complessità di A è un $O(f(x))$ se $g(x)$ è un $O(f(x))$; analogamente, diremo che la complessità di A è un $\Omega(f(x))$ o un $\Theta(f(x))$ se $g(x)$ è un $\Omega(f(x))$ o un $\Theta(f(x))$.

A.3 Problemi trattabili e problemi intrattabili

Chiameremo *trattabili* i problemi per cui esistono algoritmi la cui complessità sia un $O(p(x))$, con $p(x)$ un polinomio in x , e *intrattabili* i problemi per cui un tale algoritmo non esiste: le seguenti tabelle chiariscono il perché di questa distinzione. In questa tabella vengono forniti, per diverse funzioni di complessità f , i tempi di esecuzione (in secondi, ove non diversamente specificato) per alcuni valori di n su un calcolatore che richiede $1e^{-6}$ secondi per effettuare un'operazione elementare.

f	10	20	40	60
n	$1e^{-5}$	$2e^{-5}$	$4e^{-5}$	$6e^{-5}$
n^3	$1e^{-3}$	$8e^{-3}$	$7e^{-2}$	$2e^{-1}$
n^5	$1e^{-1}$	3.2	1.7 min.	13 min.
2^n	$1e^{-3}$	1	13 giorni	36600 anni
3^n	$6e^{-2}$	1 ora	$4e^5$ anni	$1e^{13}$ anni

In questa tabella vengono invece indicati i miglioramenti ottenibili, in termini di dimensioni delle istanze risolvibili, per diverse funzioni di complessità, al migliorare della tecnologia dei calcolatori: con x_i abbiamo indicato la dimensione di un'istanza risolvibile oggi in un minuto per la i -esima funzione di complessità.

f	Computer odierno	100 volte più veloce	10000 volte più veloce
n	x_1	$100x_1$	$10000x_1$
n^3	x_2	$4.6x_2$	$21.5x_2$
n^5	x_3	$2.5x_3$	$6.3x_3$
2^n	x_4	$x_4 + 6.6$	$x_4 + 13.2$
3^n	x_5	$x_5 + 4.2$	$x_5 + 8.4$

Molti problemi di rilevante importanza pratica sono trattabili: sono problemi per i quali disponiamo di efficienti algoritmi di complessità polinomiale. Per potere effettuare una più rigorosa classificazione dei diversi problemi, facciamo riferimento a problemi in forma decisionale.

A.3.1 Le classi \mathcal{P} e \mathcal{NP}

Una prima importante classe di problemi è la classe \mathcal{NP} , costituita da tutti i problemi decisionali il cui problema di certificato associato può essere risolto in tempo polinomiale. In altri termini, i problemi in \mathcal{NP} sono quelli per cui è possibile verificare efficientemente una risposta "sì", perché è possibile decidere in tempo polinomiale se una soluzione x è ammessa per il problema. Ad esempio, il problema della soddisfattibilità proposizionale (SAT), introdotto in 1.2.3.1, è un problema in \mathcal{NP} : dato un qualunque assegnamento di valori di verità, è possibile verificare se tale assegnamento rende vera la formula in tempo lineare nella sua dimensione.

Equivalentemente, si può definire \mathcal{NP} come la classe di tutti i problemi decisionali risolvibili in tempo polinomiale da una MR nondeterministica (\mathcal{NP} va infatti inteso come Polinomiale nel calcolo Nondeterministico). Una MR nondeterministica è il modello di calcolo (astratto) in cui una MR, qualora si trovi ad affrontare un'operazione di salto condizionale,

¹In generale, tranne quando sarà detto il contrario, se i dati di un problema sono costituiti da n numeri considereremo come limitato da $\log n$ il numero di bits necessari per la codifica in binario dei numeri stessi.

può eseguire *contemporaneamente* entrambi rami dell'operazione, e questo ricorsivamente per un qualsiasi numero di operazioni. In altre parole, i problemi in \mathcal{NP} sono quelli per cui esiste una computazione di lunghezza polinomiale che può portare a costruire una soluzione ammisiibile, se esiste, ma questa computazione può essere "nascosta" entro un insieme esponenziale di computazioni analoghe tra le quali, in generale, non si sa come discriminare.

Ad esempio, per SAT si può immaginare una MR nondeterministica che costruisca l'assegnamento di valori di verità ai letterali con una computazione in cui, sequenzialmente, viene assegnato il valore di verità a ciascun letterale, in un certo ordine prestabilito, in base ad una certa condizione logica. Questo identifica un *albero di computazione* che descrive tutte le possibili esecuzioni della MR, e le cui foglie sono tutti i 2^n possibili assegnamenti di valori di verità agli n letterali della formula. Se la formula è soddisfattibile, allora esiste un cammino nell'albero di computazione, di lunghezza lineare nel numero di letterali, che porta a costruire esattamente un certificato del problema, ossia un assegnamento di valori di verità che soddisfa la formula.

Un sottoinsieme della classe \mathcal{NP} è la classe \mathcal{P} , costituita da tutti i (*problematici polinomiali*), quei problemi decisionali per i quali esistono algoritmi di complessità polinomiale che li risolvono. Una domanda particolarmente importante è se esistano problemi in \mathcal{NP} che non appartengano anche a \mathcal{P} . A questa domanda non si è a tutt'oggi stati in grado di rispondere, ma si ritiene fortemente probabile sia effettivamente $\mathcal{P} \neq \mathcal{NP}$; una breve giustificazione di questo sarà data nel paragrafo successivo.

A.3.2 Problemi \mathcal{NP} -completi e problemi \mathcal{NP} -ardui

Molti problemi, anche se apparentemente notevolmente diversi, possono tuttavia essere ricondotti l'uno all'altro; dati due problemi decisionali, P e Q , diciamo che P si riduce in tempo polinomiale a Q , e scriviamo $P \leq Q$, se, supponendo l'esistenza di un algoritmo A_Q che risolva Q in tempo costante (indipendente dalla lunghezza dell'input), esiste un algoritmo che risolve P in tempo polinomiale utilizzando come sottoprogramma A_Q : parliamo in tal caso di *riduzione polinomiale* di P a Q . Ad esempio, SAT si riduce polinomialmente alla PLI, come mostrato nel paragrafo 1.2.3.1. Quindi, se esistesse un algoritmo polinomiale per la PLI allora esisterebbe un algoritmo polinomiale per SAT: data la formula in forma normale congiuntiva, basterebbe produrre (in tempo polinomiale) il problema di PLI corrispondente, applicare l'algoritmo a tale problema e rispondere "sì" o "no" a seconda della risposta ottenuta. Possiamo quindi affermare che $SAT \leq PLI$. È facile verificare che la relazione \leq ha le seguenti proprietà:

1. è riflessiva: $A \leq A$;
2. è transitiva: $A \leq B$ e $B \leq C \Rightarrow A \leq C$;
3. se $A \leq B$ e $A \notin \mathcal{P}$, allora $B \notin \mathcal{P}$;
4. se $A \leq B$ e $B \in \mathcal{P}$, allora $A \in \mathcal{P}$.

Possiamo definire adesso la classe dei problemi \mathcal{NP} -completi: un problema A è detto \mathcal{NP} -completo se $A \in \mathcal{NP}$ e se per ogni $B \in \mathcal{NP}$ si ha che $B \leq A$. La classe dei problemi \mathcal{NP} -completi costituisce un sottoinsieme di \mathcal{NP} di particolare importanza; un fondamentale teorema, dovuto a Cook (1971), garantisce che ogni problema $P \in \mathcal{NP}$ si riduce polinomialmente a SAT, ossia che tale classe non è vuota. I problemi \mathcal{NP} -completi hanno la proprietà che se esiste per uno di essi un algoritmo polinomiale, allora necessariamente tutti i problemi in \mathcal{NP} sono risolvibili polinomialmente, e quindi è $\mathcal{P} = \mathcal{NP}$; in un certo senso, tali problemi sono i "più difficili" tra i problemi in \mathcal{NP} . Un problema che abbia come caso particolare un problema \mathcal{NP} -completo ha la proprietà di essere almeno tanto difficile quanto i problemi \mathcal{NP} -completi (a meno di una funzione moltiplicativa polinomiale): un problema di questo tipo si dice \mathcal{NP} -arduo. Si noti che un problema \mathcal{NP} -arduo può anche non appartenere a \mathcal{NP} . Ad esempio, sono \mathcal{NP} -ardui i problemi di ottimizzazione la cui versione decisionale è un problema \mathcal{NP} -completo: infatti, come si è già visto, è sempre possibile ricondurre un problema decisionale ad un problema di ottimizzazione con una opportuna scelta della funzione obiettivo. Ad esempio, il problema della PLI è \mathcal{NP} -arduo, poiché ad esso si riduce SAT, che è \mathcal{NP} -completo.

Fino ad oggi, sono stati trovati moltissimi problemi \mathcal{NP} -ardui; si può affermare che la grande maggioranza dei problemi combinatori siano \mathcal{NP} -ardui. Nonostante tutti gli sforzi dei ricercatori, non è stato possibile determinare per nessuno di essi un algoritmo polinomiale (il che avrebbe fornito algoritmi polinomiali per tutti i problemi della classe); questo fa ritenere che non esistano algoritmi polinomiali per i problemi \mathcal{NP} -completi, ossia che sia $\mathcal{P} \neq \mathcal{NP}$. Ad oggi non si conosce nessuna dimostrazione formale di questa ipotesi, ma essa è suffragata da molti indizi. Ad esempio, \mathcal{P} ed \mathcal{NP} sono solo i primi elementi di una *gerarchia polinomiale* di classi di problemi, sempre "più difficili", che colllasserebbero tutte sulla sola classe \mathcal{P} qualora fosse $\mathcal{P} = \mathcal{NP}$, il che fa ritenere questa eventualità altamente improbabile.

Per riassumere, possiamo dire che, allo stato delle conoscenze attuali, tutti i problemi \mathcal{NP} -ardui sono presumibilmente intrattabili. Naturalmente, ciò non significa che non sia in molti casi possibile costruire algoritmi in grado di risolvere efficientemente istanze di problemi \mathcal{NP} -ardui di dimensione significativa (quella richiesta dalle applicazioni reali): questo non rientra comunque nel campo della teoria della complessità computazionale ma piuttosto nel campo dell'Ottimizzazione Combinatoria (si veda il Capitolo 4 e seguenti).

A.3.3 Complessità ed approssimazione

Esistono molti altri risultati interessanti della teoria della complessità computazionale che non rientrano nello scopo di queste note. Risultati molto importanti permettono, ad esempio, di caratterizzare classi di problemi per cui non solo il problema originario, ma anche ottenere una soluzione approssimata sia un problema intrattabile.

Ricordiamo che un algoritmo euristico si dice ε -*approssimato* se produce una soluzione ε -ottima per ogni istanza. Uno *schemma di approssimazione* è un algoritmo che, oltre all'istanza p di P , prende in input anche un parametro $\varepsilon > 0$ e produce una soluzione \tilde{x} che è ε -ottima; sostanzialmente uno schema di approssimazione è famiglia infinita di algoritmi ε -approssimati, uno per ogni valore di ε . Tipicamente, la complessità di un tale algoritmo sarà anche una funzione dell'approssimazione ε voluta (crescente con $1/\varepsilon$): uno schema di approssimazione è detto:

- *polinomiale*, se la sua complessità è polinomiale nella dimensione dell'istanza p ;
- *pienamente polinomiale*, se la sua complessità è polinomiale nella dimensione di p ed in $1/\varepsilon$.

Accenniamo brevemente qui ad un noto risultato che riguarda la difficoltà di costruire schemi di approssimazione pienamente polinomiali per un problema di ottimizzazione.

Esistono problemi \mathcal{NP} -completi che sono risolvibili polinomialmente nella lunghezza dell'input se codificati in unario, mentre non lo sono con codifiche più compatte: un esempio è il problema dello zaino (1.2.2.1). Tali problemi sono in un certo senso più facili di quelli che richiedono algoritmi esponenziali indipendentemente dalla codifica dell'input. Viene detto *pseudopolinomiale* un algoritmo che risolva uno di tali problemi in tempo polinomiale nella lunghezza dell'input codificato in unario; un problema è quindi detto *pseudopolinomiale* se può essere risolto per mezzo di un algoritmo pseudopolinomiale. Viene chiamato \mathcal{NP} -completo *in senso forte* un problema \mathcal{NP} -completo per cui non esistono algoritmi pseudopolinomiali; analogamente, viene chiamato \mathcal{NP} -arduo *in senso forte* un problema di ottimizzazione che abbia come versione decisionale un problema \mathcal{NP} -completo in senso forte. I problemi \mathcal{NP} -completi in senso forte sono tipicamente quelli che “non contengono numeri”: ad esempio, SAT è un problema \mathcal{NP} -completo in senso forte. Un diverso esempio è il problema del commesso viaggiatore (TSP) (si veda il §1.2.2.3), che resta \mathcal{NP} -arduo anche se i costi degli archi sono limitati a due soli possibili valori, ad esempio “0” e “1”. Infatti, dato un algoritmo per TSP con costi 0/1 è possibile risolvere (in modo ovvio) il problema del ciclo Hamiltoniano su un grafo, che è notoriamente \mathcal{NP} -completo. È possibile dimostrare che l'esistenza di algoritmi pseudopolinomiali per un problema è equivalente all'esistenza di *schemi di approssimazione pienamente polinomiali* per il problema; in altri termini, nessun problema \mathcal{NP} -arduo in senso forte (la grande maggioranza) ammette schemi di approssimazione pienamente polinomiali (a meno che $\mathcal{P} = \mathcal{NP}$). Di conseguenza, per la maggior parte dei problemi \mathcal{NP} -ardui è difficile non solo risolvere il problema originario, ma anche una sua approssimazione arbitraria.

Risultati di questo tipo sono stati dimostrati anche per approssimazioni con errore relativo fissato. Ad esempio, è stata definita un'operazione di riduzione simile a α che preserva l'approssimabilità, ossia tale che se esiste un algoritmo ε -approssimato per un certo problema A ed un altro problema B si riduce ad A , allora esiste un algoritmo ε -approssimato (possibilmente per un diverso ε) anche per B . Sfruttando alcuni risultati che mostrano come per certi problemi non possano esistere algoritmi ε -approssimati con ε più piccolo di una specifica soglia (a meno che $\mathcal{P} = \mathcal{NP}$), si dimostra che per una grande classe di problemi di ottimizzazione non possono esistere algoritmi di approssimazione polinomiali con precisione arbitrariamente piccola.