UNIVERSITÀ DI PISA

Dipartimento di Ingegneria
Corso di Laurea Triennale in Ingegneria Informatica

Appunti

Ricerca operativa

Professori:
Prof. Pappalardo

Autore:
Enea Passardi

Anno Accademico 2024/2025

Indice

Unimap

1. **Lun 23/09/2024 10:30-12:30 (2:0 h)** lezione: Prerequisiti del corso. Problemi di Ricerca Operativa. Variabili decisionali, funzione obiettivo e vincoli. Modello matematico. La Programmazione Lineare (PL) in formato primale standard. Un problema di produzione ottima. Soluzione ammissibile, soluzione ottima. Regione ammissibile. (MASSIMO PAPPALARDO)

2. **Mar 24/09/2024 08:30-10:30 (2:0 h)** lezione: Risoluzione geometrica di un problema di PL in 2 variabili. Curve di isocosto. Definizione di poliedro. Poliedri vuoti, poliedri illimitati, problemi illimitati. Numero di soluzioni ottime di un problema di PL. Cono di competenza. (MASSIMO PAPPALARDO)

3. **Mer 25/09/2024 08:30-10:30 (2:0 h)** lezione: Il problema dell'assegnamento di costo minimo. Caso cooperativo e caso non cooperativo. Il modello matematico. Trasformazione in formato primale standard. Le variabili di scarto. (MASSIMO PAPPALARDO)

4. **Gio 26/09/2024 08:30-10:30 (2:0 h)** lezione: Combinazioni convesse e combinazioni coniche. Teorema di Weyl (rappresentazione dei poliedri). Teorema fondamentale della PL. Problemi illimitati superiormente. Esempi. (MASSIMO PAPPALARDO)

5. **Lun 30/09/2024 10:30-12:30 (2:0 h)** lezione: Definizione di vertice. Il concetto di base e di soluzione di base. Soluzioni di base ammissibili. Soluzioni di base degeneri. Teorema di caratterizzazione dei vertici. Esempi ed esercizi. La funzione linprog. (MASSIMO PAPPALARDO)

6. **Mar 01/10/2024 08:30-10:30 (2:0 h)** lezione: Soluzioni di base primali degeneri. Ancora sulle trasformazioni equivalenti: variabili di scarto e loro caratteristiche, imporre la positività delle variabili. Il formato duale standard. (MASSIMO PAPPALARDO)

7. **Mer 02/10/2024 08:30-10:30 (2:0 h)** lezione: Soluzioni di base di un poliedro in formato duale standard. Ammissibili e non ammissibili. Degeneri e non degeneri. Esercizi ed esempi. Problema del trasporto. Modello matematico. Esempi. (MASSIMO PAPPALARDO)

8. **Gio 03/10/2024 08:30-10:30 (2:0 h)** lezione: Teoria della dualità. Il problema duale. Il teorema della dualità forte. Soluzioni di base complementari. Il test di ottimalità per i vertici. Esempi ed esercizi. (MASSIMO PAPPALARDO)

9. **Lun 07/10/2024 10:30-12:30 (2:0 h)** lezione: L'algoritmo del simplesso primale. Le regole del cambio di base. Esempi es esercizi. Interpretazione geometrica dell'algoritmo. Le regole anticiclo. Il caso degenere. (MASSIMO PAPPALARDO)

10. **Mar 08/10/2024 08:30-10:30 (2:0 h)** lezione: Algoritmo del simplesso duale quando si è in possesso di una base duale ammissibile e primale non ammissibile. Il caso degenere. Esempi ed esercizi. (MASSIMO PAPPALARDO)

11. **Mer 09/10/2024 08:30-10:30 (2:0 h)** lezione: Il duale ausiliario. Poliedri vuoti. Base di partenza del simplesso. Il caso degenere. Esempi e controesempi. Esercizi. (MASSIMO PAPPALARDO)

12. **Gio 10/10/2024 08:30-10:30 (2:0 h)** lezione: Il problema dello zaino binario e dello zaino intero. Il modello matematico. Il concetto di rendimento tramite il duale del rilassato continuo. Problemi di PLI in formato standard. Le valutazioni inferiori e le superiori. Il ruolo del rilassamento continuo. Tecniche "greedy" per la ricerca di soluzioni ammissibili. Casi in cui si può usare l'arrotondamento per eccesso o per difetto. (MASSIMO PAPPALARDO)

13. **Lun 14/10/2024 10:30-12:30 (2:0 h)** lezione: Esercitazione su problemi di "zaino". Ancora sull'interpretazione geometrica del simplesso primale e di quello duale. Esercitazione scritta di ricapitolazione. (MASSIMO PAPPALARDO)

14. **Mar 15/10/2024 08:30-10:30 (2:0 h)** lezione: Tecniche "greedy" per la ricerca di soluzioni ammissibili. Casi in cui si può usare l'arrotondamento per eccesso o per difetto. Il caso del problema di assegnamento: matrici unimodulari. Vertici interi. Teorema dell'interezza per il problema del trasporto. Teorema di equivalenza tra PL e PLI. Il ruolo di convS quando S è l'insieme delle soluzioni ammissibili di un problema di PLI limitato (MASSIMO PAPPALARDO)

15. **Mer 16/10/2024 08:30-10:30 (2:0 h)** lezione: Il problema del "bin-packing". Il modello matematico di PLI. Valutazioni inferiori e superiori. Algoritmi del next fit decreasing, first fit decreasing and best fit decreasing. I comandi di intlinprog. Esempi ed esercizi. (MASSIMO PAPPALARDO)

16. **Gio 17/10/2024 08:30-10:30 (2:0 h)** lezione: Disuguaglianze valide e piani di taglio. I piani di taglio di Gomory. Esempi ed esercizi. (MASSIMO PAPPALARDO)

17. **Lun 21/10/2024 08:30-10:30 (2:0 h)** lezione: Piani di taglio per lo zaino intero. Piani di taglio per problemi di produzione di massimo e di minimo. Esercizi svolti. (MASSIMO PAPPALARDO)

18. **Mar 22/10/2024 08:30-10:30 (2:0 h)** lezione: Il problema del TSP asimmetrico. Assegnamenti e cicli hamiltoniani. La valutazione inferiore. L'algoritmo delle toppe per la valutazione superiore. Esempi. (MASSIMO PAPPALARDO)

19. **Mer 23/10/2024 08:30-10:30 (2:0 h)** lezione: Tecniche per costruire soluzioni ammissibili di tipo "greedy". Il k-albero di costo minimo come valutazione inferiore. L'algoritmo del nodo più vicino per la valutazione superiore. (MASSIMO PAPPALARDO)

20. **Gio 24/10/2024 08:30-10:30 (2:0 h)** lezione: Il problema del TSP simmetrico. I vincoli di grado. Il modello matematico. Il metodo del "Branch and Bound" per problemi di PLI booleana. L'albero di enumerazione totale. Le regole di taglio per problemi di massimo. (MASSIMO PAPPALARDO)

21. **Mar 29/10/2024 08:30-10:30 (2:0 h)** lezione: Il "Branch and Bound" per problemi di minimo. Le 3 regole di taglio. Esempi ed esercizi. (MASSIMO PAPPALARDO)

22. **Mer 30/10/2024 08:30-10:30 (2:0 h)** lezione: Problemi di copertura. La funzione costo. La matrice di copertura. Tecniche di riduzione. Modello matematico e comandi intlinprog. Valutazioni inferiori e superiori. L'algoritmo di Chvatal. Esempi. Problemi di massima copertura. Il modello matematico. I comandi intlinprog. (MASSIMO PAPPALARDO)

23. **Lun 04/11/2024 10:30-12:30 (2:0 h)** lezione: Esercitazione scritta e commentata sulla prima metà del corso (PL e PLI). (MASSIMO PAPPALARDO)

24. **Mar 05/11/2024 08:30-10:30 (2:0 h)** lezione: Il problema del flusso di costo minimo su reti non capacitate. Le equazioni di bilancio ai nodi. La matrice E di incidenza della rete. Il rango di E è n-1. Reti bilanciate. Basi ed alberi di copertura: relazioni. Flussi di base con visita posticipata per foglia. Caso degenere. Esercizi. (MASSIMO PAPPALARDO)

25. **Mer 06/11/2024 08:30-10:30 (2:0 h)** lezione: Il problema dei potenziali su reti non capacitate. Potenziali di base. Potenziali degeneri. Visita anticipata da una radice. Condizioni di Bellman. Esercizi svolti in aula (MASSIMO PAPPALARDO)

26. **Gio 07/11/2024 08:30-10:30 (2:0 h)** lezione: Il simplesso su reti non capacitate. Il ciclo; la rottura del ciclo. Arco entrante e arco uscente. La correttezza. Il caso degenere. Esercizio svolto in aula. Il problema dell'assegnamento di costo minimo come problema di flusso su una rete (MASSIMO PAPPALARDO)

27. **Lun 11/11/2024 10:30-12:30 (2:0 h)** lezione: Il problema dell'albero orientato dei cammini di costo minimo visto come problema di flusso su reti. il modello matematico. Specifiche del simplesso sul problema dei cammini minimi. Esercizi. Il problema del trasporto di costo minimo rivisto come problema di flusso su reti. (MASSIMO PAPPALARDO)

28. **Mar 12/11/2024 08:30-10:30 (2:0 h)** lezione: Il problema del flusso di costo minimo su reti capacitate. Il modello matematico. Teorema di caratterizzazione delle basi tramite le tripartizioni. Flusso di base. Esercizi. (MASSIMO PAPPALARDO)

29. **Mer 13/11/2024 08:30-10:30 (2:0 h)** lezione: Il problema dei potenziali su reti capacitate. Il modello matematico. Le soluzioni di base. Calcolo esplicito tramite visita dell'albero T. Ammissibilità e condizioni di Bellman. Il caso degenere. Esercizi svolti in aula. (MASSIMO PAPPALARDO)

30. **Gio 14/11/2024 08:30-10:30 (2:0 h)** lezione: L'algoritmo del simplesso per problemi di flusso di costo minimo su reti capacitate. L'arco che viola Bellman. I due casi (da L o da U). Il ciclo ed il verso. Il caso degenere. Esercizi svolti in aula (MASSIMO PAPPALARDO)

31. **Lun 18/11/2024 10:30-12:30 (2:0 h)** lezione: Il problema del flusso massimo. Il modello matematico come problema di PL e i comandi linprog. Il modello matematico come problema di flusso di costo minimo: l'arco fittizio ed il problema di circolazione. Il simplesso su reti adattato al problema del flusso massimo. (MASSIMO PAPPALARDO)

32. **Mar 19/11/2024 08:30-10:30 (2:0 h)** lezione: Il problema del taglio di capacità minima. Il teorema max-flow-min-cut. Esercizi svolti in aula. L'algoritmo di Ford-Fulkerson. I cammini aumentanti. La procedura di Edmonds-Karp. La correttezza di Ford-Fulkerson e la finitezza. Esempi. Il caso dell'utilizzo dell'arco fittizio. (MASSIMO PAPPALARDO)

33. **Mer 20/11/2024 08:30-10:30 (2:0 h)** lezione: L'algoritmo di Dijkstra per l'albero dei cammini minimi. Relazioni con il simplesso su reti. Correttezza e finitezza. Esercizi svolti in aula. Il problema di massima cardinalità. (MASSIMO PAPPALARDO)

34. **Gio 21/11/2024 08:30-10:30 (2:0 h)** lezione: Esercitazione di ricapitolazione generale di tutti gli argomenti svolti. (MASSIMO PAPPALARDO)

35. **Lun 25/11/2024 10:30-12:30 (2:0 h)** lezione: Rivisitazione dei prerequisiti di Analisi II necessari alla Programmazione Non Lineare. Gradiente, hessiana, condizioni necessarie o sufficienti per minimi e/o massimi locali. Equazioni parametriche di restrizioni. (MASSIMO PAPPALARDO)

36. **Mar 26/11/2024 08:30-10:30 (2:0 h)** lezione: Forme quadratiche, loro hessiane e "quadprog". Funzioni convesse e loro punti stazionari. Funzioni coercive ed esistenza del minimo globale. Minimi di restrizioni. Esempi numerici (MASSIMO PAPPALARDO)

37. **Mer 27/11/2024 08:30-10:30 (2:0 h)** lezione: I domini di $\mathbb{R}^n$. Rappresentazione come curve di livelli di funzioni C^1. Domini chiusi e limitati. Domini convessi. Condizione sufficiente per la convessità del dominio. Domini regolari: la condizione di Slater e la condizione di Mangasarian. Esempi. (MASSIMO PAPPALARDO)

38. **Gio 28/11/2024 08:30-10:30 (2:0 h)** lezione: Il teorema di Lagrange-Karush-Kuhn-Tucker (LKKT) per problemi di minimo o massimo vincolati. Esempi ed esercizi. (MASSIMO PAPPALARDO)

39. **Lun 02/12/2024 10:30-12:30 (2:0 h)** lezione: Problemi convessi e problemi concavi. Condizioni sufficienti del primo ordine per minimi globali in problemi convessi e massimi globali in problemi concavi. Massimi di convesse su poliedri. Minimi di concave su poliedri. Esempi ed esercizi. (MASSIMO PAPPALARDO)

40. **Mar 03/12/2024 08:30-10:30 (2:0 h)** lezione: La ricerca locale per discriminare minimi locali da selle e massimi locali da selle. Il concetto di restrizione. Restrizioni a semirette per la ricerca di direzioni di salita o di discesa. Esercizi svolti in aula. (MASSIMO PAPPALARDO)

41. **Mer 04/12/2024 08:30-10:30 (2:0 h)** lezione: Successioni minimizzanti ricorsive. Direzioni di discesa. Una condizione sufficiente. Il metodo del gradiente. Il passo "ideale". Il passo costante. Il metodo di Newton. Teoremi di convergenza dei metodi. Esempi ed esercizi. (MASSIMO PAPPALARDO)

42. **Gio 05/12/2024 08:30-10:30 (2:0 h)** lezione: Il passo secondo le regole di Armijo-Goldstein-Wolfe. Esercizi svolti in aula. Metodi per PNL vincolata su poliedri limitati. L'algoritmo di Frank-Wolfe. (MASSIMO PAPPALARDO)

43. **Lun 09/12/2024 08:30-10:30 (2:0 h)** lezione: Esercizi svolti sull'algoritmo di Frank-Wolfe. La funzione "quadprog". L'algoritmo del gradiente proiettato su poliedri limitati. La matrice della proiezione ortogonale. (MASSIMO PAPPALARDO)

44. **Mer 11/12/2024 08:30-10:30 (2:0 h)** lezione: Il caso della proiezione nulla del gradiente. Eliminazione di un vincolo attivo. Calcolo esplicito dei moltiplicatori LKKT. Esercizio svolto in aula sul metodo del gradiente proiettato. (MASSIMO PAPPALARDO)

45. **Gio 12/12/2024 08:30-10:30 (2:0 h)** lezione: Esercitazione scritta svolta in aula: prova di autovalutazione finale. (MASSIMO PAPPALARDO)

Introduzione

Citazione Prof. Stea: "*dopo che seguite il corso (di **ricerca operativa**), anche andà dar macellaio diventa un problema di ricerca operativa. [...] La cosa positiva di essere un ingegnere informatico è che vi chiameranno tutti per organizzare le partite di 'carcetto*".

Prerequisiti

Per poter sostenere l'esame di **ricerca operativa**, con docente titolare **Massimo Pappalardo** è necessario aver già **sostenuto** e **verbalizzato** gli esami di **Algebra lineare** e **Analisi Matematica II**. Questi esami sono fondamentali in quanto forniscono delle competenze necessarie alla corretta comprensione del corso; Più precisamente

- **Algebra lineare**:
  - *Operazioni tra matrici.*
  - *Matrici inverse.*
  - *Sistemi lineari.*
  - *Autovalori di una matrice.*
- **Analisi Matematica II**:
  - *Derivate parziali*, *derivate direzionali*, *gradienti* e *matrici Hessiane.*
  - *Insieme del piano definiti da curve di livello.*

Problemi della ricerca operativa

I problemi della ricerca operativa sono dei **problemi decisionali** di *ottimizzazione di un obiettivo in presenza di risorse limitate.* La loro caratteristica principale è che, generalmente, si costruisce un modello matematico che descrive il problema, per poi risolverlo.

Un **modello matematico di ottimizzazione** è un problema di **massimo** o **minimo** di una funzione $f : \mathbb{R}^n \to \mathbb{R}$ ristretta ad una regione Ω dello spazio, definita da vincoli espressi tramite **intersezioni di insieme di livello di funzioni a valori reali**

$$\Omega = \{x \in \mathbb{R}^n : g(x) \leq 0,\ h(x) = 0\} \tag{1}$$

dove $g : \mathbb{R}^n \to \mathbb{R}^m$ e $h : \mathbb{R}^n \to \mathbb{R}^p$. Dati due vettori $\vec{a}, \vec{b} \in \mathbb{R}^n$, la notazione $a \leq b$ implica che

$$a_i \leq b_i \ \ \forall i = 1, \ldots, n \tag{2}$$

Un'altra proprietà interessante riguardo ai problemi visti fino ad ora è la **proprietà di equivalenza** che esiste tra i problemi di **massimo** e quelli di **minimo**; Nello specifico, vale che

$$\max f(x) = -\min -f(x) \tag{3}$$

quindi, i punti di massimo di f equivalgono con i punti di minimo di $-f$.

La funzione f è chiamata **funzione obiettivo**, l'insieme Ω **regione ammissibile**, ciascuna disequazione $g_i(x) \leq 0,\ i = 1, \ldots, n$ e $h_j(x) = 0,\ j = 1, \ldots, n$ che descrivono Ω, sono dette **vincoli**.

Punti di massimo e minimo Un punto x_0 è detto **punto di minimo** (o **massimo**) **globale** se e solo se

$$f(x_0) \leq f(x)\ \ (f(x_0) \geq f(x))\ \ \forall x \in \Omega \tag{4}$$

Un punto x_1 è detto **punto di minimo** (o **massimo**) **relativo** di f su Ω se e solo se esiste una sfera $B(x_1, r)$ di centro x_1 e raggio r tale per cui

$$f(x_1) \leq f(x)\ \ (f(x_1) \geq f(x))\ \ \forall x \in \Omega \cap B(x_1, r) \tag{5}$$

Evoluzione di un problema di ricerca operativa

Un problema di **ricerca operativa** si evolve attraverso vari stadi. I principali sono:

- Raccolta dei dati del problema **utili ai fini della soluzione**.
- **Costruzione del modello matematico** che, rappresenti un **giusto equilibrio** tra problema reale e trattabilità matematica.
- Studio delle **proprietà teoriche** del **modello matematico** e **ricera della soluzione ottima**.
- Costruzione e implementazione di un algoritmo.
- Risoluzione numerica.
- **Retroazione** sul problema di partenza.

Gli algoritmi sono gli strumenti matematici per risolvere un problema e che, se la soluzione è condotta tramite il calcolo automatico, divengono di fondamentale importanza. Un algoritmo può essere definito come segue

Definition 0.1 (Algoritmo)**.** Un algoritmo è una sequenza di istruzioni chiare che un esecutore, in tempo limitato, compie per ottenere una soluzione di un problema.

Di fondamentale importanza è **dimostrare la correttezza di un algoritmo**, dimostrando quindi che la soluzione ottenuta è anche quella cercata.

Note dell'autore

la dispensa contiene quanto detto dal professor Pappalardo durante le lezioni, con aggiunta di alcune integrazioni dal libro di testo consigliato. Consiglio anche di seguire attivamente il corso, poiché, durante le lezioni, il professore propone molti quesiti che potranno essere soggetto di esame orale.

Consigli per l'esame

Il consiglio che vi posso dare per passare l'esame di ricerca operativa è evitare di pensare che, come direbbe il **Prof. Stea**, questo esame sia *attaccabile a forza bruta*. Infatti, nonostante gli esercizi siano, per natura, molto algoritmici, l'esame orale richiede una comprensione della materia che va ben oltre il semplice *svolgere bovinamente esercizi*. Un consiglio che posso dare è quello di svolgere gli esercizi in *modalità debugging*, cercando di capire il perché di ogni singolo passaggio di ogni singolo algoritmo.

Inoltre, il professore è estremamente disponibile e tiene molto al fatto che gli studenti capiscano la materia, quindi, se qualcosa non vi torna, non abbiate paura a chiedere. Ricordo che la conoscenza e la comprensione della materia sono imprescindibili per passare l'esame orale, specialmente se si intende anche farlo con un buon voto. All'esame orale sarà fondamentale conoscere i teoremi, ma sarà altrettanto fondamentale conoscere la loro applicazione e il loro utilizzo: una conoscenza limitata alla sola ripetizione a memoria non è sufficiente. Capita molto spesso che le domande dell'orale vertano su esempi reali; ad esempio, al mio orale, il professore mi chiese di spiegare come fare per spedire, dato un grafo, 1000 terabyte di informazione dal nodo 1 al nodo 8.

Parte I

Programmazione lineare

Capitolo 1

Problemi della *programmazione lineare*

*Prima di introdurre i problemi della **programmazione lineare**, occorre fare una piccola nota a margine. Tutti i tre problemi che andiamo ad enunciare, sono in qualche modo, riconducibili a problemi di produzione.*

1 Produzione

I problemi di produzione sono tutti quei problemi in cui si cerca di **massimizzare** o **minimizzare** dei valori legati a un processo produttivo. Ad esempio, supponiamo che si debbano produrre n oggetti, ciascuno dei quali composto da m **diverse materie prime**, diverse da loro, date da una matrice di composizione A.

In tale matrice, l'elemento a_{ij} rappresenta la quantità del *i-esimo* materiale necessaria a produrre il *j-esimo* prodotto.

Supponiamo inoltre di conoscere il guadagno c_j dato dalla vendita di un prodotto j e la disponibilità b_i del materiale i. Si voglia determinare il piano di produzione ottimo, o in altri termini, quello che massimizza il guadagno del prodotto. Possiamo introdurre anche una nuova quantità, $x_j,\ j = 1, \ldots, n$ stante a indicare la quantità prodotta del prodotto j.

Il nostro obiettivo è massimizzare il guadagno ottenuto, andando a variare la quantità di prodotti j che vengono prodotti; In particolare, il nostro obiettivo è trovare un massimo della funzione

$$\max \sum_{j=1}^{n} c_j x_j = c_1 x_1 + c_2 x_2 + \cdots + c_n x_n \tag{1}$$

ora che abbiamo definito la nostra funzione obiettivo, è importante definire la **regione ammissibile** Ω, quindi l'insieme delle x che soddisfano i **vincoli**; Che nel nostro caso sono:

- La **quantità di materiale** utilizzato nella produzione dei prodotti deve essere **minore** o **uguale** a quella **disponibile**. Questo vincolo può essere trasposto in linguaggio matematico come

  $$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \tag{2}$$

  Ovviamente questa condizione deve valere per ogni materiale.

- La quantità di prodotti effettivamente prodotti, deve essere maggiore o uguale di 0.

  $$x_j \geq 0 \tag{3}$$

Una volta definiti i vincoli possiamo trovare il sistema che descrive il nostro problema

$$\begin{cases} \max \sum_{j=1}^{n} c_j x_j \\ \sum_{j=1}^{n} a_{ij} x_j \leq b_i \ \ \forall i = 1, \ldots, m \\ x_j \geq 0 \ \ \forall j = 1, \ldots, n \end{cases} \tag{4}$$

Per comprendere meglio, prendiamo un esempio pratico di formalizzazione di un problema.

> *Un'azienda deve produrre due tipi di tessuto. Supponiamo che per produrre ogni quintale del primo tessuto ci sia bisogno di 28 kg di lana e 7 kg di cotone, mentre per produrre il secondo tipo ci sia bisogno di 7 kg di lana e 14 kg di cotone. Supponiamo inoltre che per produrre i tessuti ci sia bisogno di 3 ore di lavoro di un operaio per ogni quintale di prodotto. Supponiamo di avere a disposizione ogni settimana 168 kg di lana disponibili, 84 kg di cotone disponibili e 42 ore di lavoro disponibile. Siano 20 e 10 euro i guadagni per chilo, rispettivamente, per il tessuto 1 e per il tessuto 2. Indichiamo con x_1 e x_2 rispettivamente i chilogrammi prodotti del primo e del secondo tessuto* (Esempio preso dal libro del docente)

Quindi, la nostra funzione da massimizzare sarà equivalente a quella vista in precedenza

$$\max \sum_{j=1}^{2} c_j x_j = 20x_1 + 10x_2 \tag{5}$$

i vincoli saranno invece

- La quantità di lana utilizzata settimanalmente deve essere inferiore o uguale a quella disponibile
$$28x_1 + 7x_2 \leq 168 \tag{6}$$
- La quantità di cotone utilizzata settimanalmente deve essere inferiore o uguale a quella disponibile
$$7x_1 + 14x_2 \leq 168 \tag{7}$$
- La quantità di ore utilizzata per produrre i tessuti deve essere inferiore o uguale a quella disponibile settimanalmente
$$3x_1 + 3x_2 \leq 42 \tag{8}$$
- Ovviamente, le quantità di tessuti prodotti devono essere maggiori o uguali a 0
$$x_1, x_2 \geq 0 \tag{9}$$

Adesso, avendo definito tutte le caratteristiche del nostro problema, possiamo andare a scrivere il sistema associato

$$\begin{cases} \max 20x_1 + 10x_2 \\ 28x_1 + 7x_2 \leq 168 \\ 7x_1 + 14x_2 \geq 84 \\ 3x_1 + 3x_2 \geq 42 \\ x_1, x_2 \geq 0 \end{cases} \tag{10}$$

Se però abbiamo un prodotto che è **indivisibile**, come ad esempio, un vestito, dobbiamo anche assicurarci che $x_j \in \mathbb{Z}$ (**vincolo di interezza**).

2 Dieta

Si supponga di avere a disposizione n cibi contenenti m principi attivi nutrizionali essenziali. Supponiamo di conoscere la quantità a_{ij} dell'*i-esimo* principio attivo nutrizionale contenuto nel *j-esimo* cibo ed il costo unitario c_j dell'alimento *j-esimo*. Si vuole determinare una dieta giornaliera che minimizzi il costo degli alimenti, soddisfando il fabbisogno giornaliero di ogni principio attivo, definito come b_i. Come nel caso precedente partiamo andando a cercare la **funzione obiettivo**

$$\min \sum_{j=1}^{n} c_j x_j = c_1 x_1 + c_2 x_2 + \cdots + c_j x_j \tag{11}$$

nella formulazione del problema ci siamo posti un vincolo

- Il **fabbisogno giornaliero** di ogni principio attivo **deve essere soddisfatto**

$$\sum_{j=0}^{n} a_{ij} x_j \geq b_i \ \ \forall i = 1, \ldots, m \tag{12}$$

una volta definito anche il vincolo si procede andando a formulare il problema effettivo

$$\begin{cases} \min \sum_{j=1}^{n} c_j x_j \\ \sum_{j=0}^{n} a_{ij} x_j \geq b_i \ \ \forall i = 1, \ldots, m \\ x_j \geq 0 \ \ \forall j = 1, \ldots, n \end{cases} \tag{13}$$

Per comprendere meglio, prendiamo un esempio pratico di formalizzazione di un problema

> *Si hanno a disposizione quattro prodotti grezzi (orzo, avena, polpa di barbabietola, farina di arachidi) con i quali si vuole produrre un mangime per animali. Il cibo deve contenere almeno 20 unità di proteine e 5 unità di grassi*

I costi unitari dei prodotti e le unità di proteine e di grassi contenute in ogni unità di prodotto sono:

	proteine	**grassi**	**costo**
orzo	12	2	24
avena	12	6	30
barbabietola	40	12	40
arachidi	60	2	50

Indichiamo con x_1, x_2, x_3 e x_4 rispettivamente il numero di unità di orzo, avena, barbabietola e farina di arachidi impegnate nel mangime. Possiamo procedere per passi, iniziando a definire la **funzione obiettivo**

$$\min \sum_{j=1}^{4} c_j x_j = c_1 x_1 + c_2 x_2 + c_3 x_3 + c_4 x_4 = 24x_1 + 30x_2 + 40x_3 + 50x_4 \tag{14}$$

Adesso possiamo lavorare i vincoli. Il problema ci richiede che

- La quantità di grassi assunta sia maggiore o uguale a quella necessaria giornalmente

$$2x_1 + 6x_2 + 12x_3 + 2x_4 \geq 6 \tag{15}$$

- La quantità di proteine assunta sia maggiore o uguale a quella necessaria giornalmente

$$12x_1 + 12x_2 + 40x_3 + 60x_4 \geq 20 \tag{16}$$

Il problema può essere quindi così formulato

$$\begin{cases} \min\ 24x_1 + 30x_2 + 40x_3 + 50x_4 \\ 2x_1 + 6x_2 + 12x_3 + 2x_4 \geq 6 \\ 12x_1 + 12x_2 + 40x_3 + 60x_4 \geq 20 \\ x_1, x_2, x_3, x_4 \geq 0 \end{cases} \tag{17}$$

3 Miscelazione

Supponiamo di avere a disposizione alcuni prodotti che devono essere miscelati al fine di ottenerne altri. Nella produzione dei nuovi materiali si suppone che la miscela debba contenere almeno una certa quantità percentuale di ciascuno dei prodotti base. I costi di miscelazione si suppongono proporzionali, ed è normalmente fissata una quantità di partenza disponibile di ognuno dei prodotti base.

Si procede prendendo un problema pratico

> *Una compagnia raccoglie 4 diversi tipi di materiali di scarto per miscelarli e formare 3 prodotti vendibili A, B e C.*

Una prima tabella di dati è la seguente

prodotto	**miscelazione**	**costo** (euro/kg)	**profitto** (euro/kg)
A	materiale 1: $\leq 30\%$ materiale 2: $\geq 40\%$ materiale 3: $\leq 50\%$ materiale 4: $= 20\%$	3	8.50
B	materiale 1: $\leq 50\%$ materiale 2: $\geq 10\%$ materiale 4: $= 10\%$	2.50	7
C	materiale 1: $\leq 70\%$	2	5.50

Supponiamo che il materiale abbia una certa disponibilità ed un costo di trattamento dati da:

DA INSERIRE TABELLA

Supponiamo che siano presenti altri vincoli: **almeno la metà di ogni materiale deve essere utilizzata e sono a disposizione al massimo 3000 euro per il trattamento dei materiali**. Introduciamo una variabile x_{ij} come la quantità (in Kg) del materiale j utilizzata per produrre il prodotto i.

Possiamo ora cercare la funzione obiettivo, che nel nostro caso è la funzione che ci permette di massimizzare il guadagno sulla produzione e vendita del prodotto.

$$\max\ 5.5\left(\sum_{j=1}^{4} x_{Aj}\right) + 4.5\left(\sum_{j=1}^{4} x_{Bj}\right) + 3.5\left(\sum_{j=1}^{4} x_{Cj}\right) \tag{18}$$

I coefficienti iniziali sono ottenuti dalla differenza tra **costo** e **profitto** del prodotto. Adesso possiamo andare a definire tutti i vincoli. Per comodità li divideremo il 4 classi:

- **Vincoli di miscelazione**: specificano che ogni componente presente nella miscela deve essere in proporzione corretta rispetto al totale

$$\begin{cases} x_{A1} \leq 0.3(x_{A1} + x_{A2} + x_{A3} + x_{A4}) \\ x_{A2} \geq 0.4(x_{A1} + x_{A2} + x_{A3} + x_{A4}) \\ x_{A3} \leq 0.5(x_{A1} + x_{A2} + x_{A3} + x_{A4}) \\ x_{A4} = 0.2(x_{A1} + x_{A2} + x_{A3} + x_{A4}) \\ x_{B1} \leq 0.5(x_{B1} + x_{B2} + x_{B3} + x_{B4}) \\ x_{A2} \geq 0.1(x_{B1} + x_{B2} + x_{B3} + x_{B4}) \\ x_{B4} = 0.1(x_{B1} + x_{B2} + x_{B3} + x_{B4}) \\ x_{C1} = 0.7(x_{C1} + x_{C2} + x_{C3} + x_{C4}) \end{cases} \tag{19}$$

- **Vincoli di disponibilità**: specificano che non si può utilizzare più materiale di quello disponibile

$$\begin{cases} x_{A1} + x_{B1} + x_{C1} \leq 3000 \\ x_{A2} + x_{B2} + x_{C2} \leq 2000 \\ x_{A3} + x_{B3} + x_{C3} \leq 4000 \\ x_{A4} + x_{B4} + x_{C4} \leq 4000 \end{cases} \tag{20}$$

- **Vincoli sui materiali trattati**: questo vincolo specifica che la metà di ogni materiale deve essere utilizzata

$$\begin{cases} x_{A1} + x_{B1} + x_{C1} \geq 1500 \\ x_{A2} + x_{B2} + x_{C2} \geq 1000 \\ x_{A3} + x_{B3} + x_{C3} \geq 2000 \\ x_{A4} + x_{B4} + x_{C4} \geq 500 \end{cases} \tag{21}$$

- **Vincoli sui costi di trattamento**: specifica che il costo per il trattamento dei materiali non superi il tetto massimo specificato

$$3(x_{A1}+x_{B1}+x_{C1})+6(x_{A2}+x_{B2}+x_{C2})+4(x_{A3}+x_{B3}+x_{C3})+5(x_{A4}+x_{B4}+x_{C4}) \leq 30000 \tag{22}$$

Capitolo 2

Teoria della PL

1 Geometria della PL

I problemi lineari trovano molto spesso delle rappresentazioni geometriche su piani, è importante quindi avere chiare alcune definizioni fondamentali

Definition 1.1. (Insieme convesso) Un sottoinsieme $K \in \mathbb{R}^n$ si definisce come ***insieme convesso*** se e solo se, dati due punti $x_1, x_2 \in K$ vale che

$$\lambda x_1 + (1-\lambda)x_2 \in K \quad \forall \lambda \in [0,1] \tag{1}$$

in termini più semplici: se K contiene i due punti x_1, x_2 allora contiene anche tutto il segmento che li congiunge.

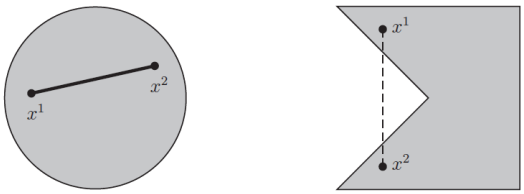


Figura 2.1: A sinistra un insieme convesso, a destra un insieme non convesso

Definition 1.2. (Combinazione convessa) Un punto $x \in \mathbb{R}^n$ si definisce ***combinazione convessa*** di $x_1, x_2, \ldots, x_m \in R^n$ se esistono dei **coefficienti** $\lambda_1, \lambda_2, \ldots, \lambda_m$ tali che

$$x = \sum_{j=1}^{m} \lambda_j x_j \quad \lambda_j \in [0,1], \quad \sum_{j=1}^{m} \lambda_j = 1 \tag{2}$$

La combinazione convessa si definisce come **propria** se $0 < \lambda_j < 1 \ \ \forall j$. La combinazione convessa potrebbe essere interpretata come un punto appartenente alla regione convessa definita dai punti $x_1, x_2, \ldots, x_m \in \mathbb{R}^n$ ottenuto come **somma pesata** dei punti moltiplicati per i rispettivi coefficienti λ.

Definition 1.3. (Involucro convesso) L'**involucro convesso** di un insieme K, denotato come conv(K), è **l'insieme di tutte le possibili combinazioni convesse** di elementi appartenenti a K.

Un'osservazione interessante è quella che si potrebbe fare nel seguente caso. Si prenda il sotto-insieme convesso $K \in \mathbb{R}^n$, esso conterà al suo interno k punti. Se ora si immaginasse di andare a

calcolare conv(K) ci si renderebbe facilmente conto di un fatto: *se per ogni combinazione convessa, mettiamo tutti i λ a 0 tranne il λ_i del punto i-esimo* e calcolassimo l'involucro convesso con questo sistema per ogni punto, si potrebbe dimostrare che

$$\text{conv}(K) = K \tag{3}$$

Da questa deduzione possiamo trarre un informazione importante. Un punto x_i dell'insieme K è ottenibile in due modi differenti

- Come combinazione di stesso con coefficiente di combinazione convessa $\lambda_i = 1$ e con $\lambda_j = 0 \; \forall j \neq i$.
- Come combinazione di tutti gli altri punti dell'insieme convesso K.

Possiamo anche dimostrare che conv(K) è il **più piccolo insieme convesso che contiene K** e quindi un insieme convesso coincide con il suo involucro convesso

$$K \subseteq \text{conv}(K) \tag{4}$$

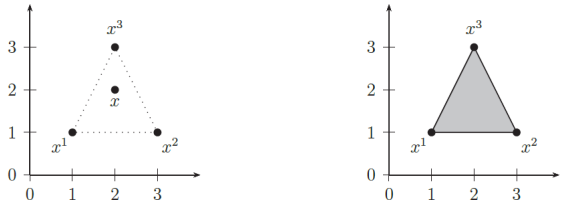
Esempio Prendiamo un insieme Q di tre punti

$$Q = \{(1,1),(3,1),(2,3)\} \tag{5}$$

dalla teoria sulle combinazioni convesse sappiamo che

$$x = \frac{1}{4}x_1 + \frac{1}{4}x_2 + \frac{1}{2}x_3 \tag{6}$$

è una combinazione convessa del nostro insieme Q. Se andassimo a calcolare l'involucro convesso, potremmo verificare facilmente la proprietà (3).



Definition 1.4. (Cono) Un sottoinsieme $K \in \mathbb{R}^n$ si definisce **cono** se e solo se per ogni punto $x \in K$ e per ogni $\lambda \geq 0$ si ha che $\lambda x \in K$. In altre parole, se K contiene un punto diverso dall'origine, allora contiene anche tutta la semiretta uscente dall'origine e passante per x.

Inoltre, un cono può essere **convesso** o **non convesso**.

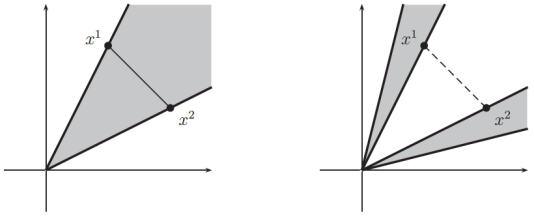


Figura 2.2: A destra un cono convesso, a sinistra uno non convesso

Quindi, un **cono convesso** non è altro che un cono che rispetta la proprietà dell'insieme convesso: *presi due punti a caso, la semiretta che li unisce fa comunque parte del cono.*

Definition 1.5. (Combinazione conica) Un punto $x \in \mathbb{R}^n$ si definisce **combinazione conica** di $x_1, x_2, \ldots, x_m \in \mathbb{R}^n$ se esistono dei coefficienti $\lambda_1, \lambda_2, \ldots, \lambda_m$ tali per cui

$$x = \sum_{i=1}^{m} \lambda_i x_i, \quad \lambda_i \geq 0 \; \forall i = 1, \ldots, m \tag{7}$$

La combinazione conica si dice propria se $\lambda_i \geq 0$.

Definition 1.6. (Involucro conico) L'involucro conico di un insieme K, denotato da cono(K), è l'insieme di tutte le possibili combinazioni coniche di elementi di K. Analogamente all'involucro convesso, possiamo dimostrare che cono(K) è il più piccolo cono convesso che contiene K e che un cono è **convesso** se e solo se coincide con il suo involucro conico.

Esempio Dato l'insieme di punti $Q = \{(2,3),(3,1)\}$, sappiamo che il punto $x = (3, 16/3)$ è una loro combinazione conica, in quanto

$$x = \frac{1}{2}x_1 + \frac{2}{3}x_2 \tag{8}$$

mentre il loro involucro conico è rappresentato dalla figura

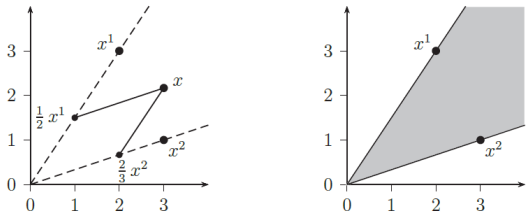


Figura 2.3: a sinistra: x è una combinazione conica di x_1 e x_2; a destra: il cono in grigio è l'involucro conico di x_1 e x_2.

1.1 Poliedri

Un **semispazio chiuso** in $\mathbb{R}^n$ può essere descritto algebricamente come l'insieme delle soluzioni di una disequazione lineare in n variabili.

Possiamo definire come **poliedro** l'intersezione di un numero finito di **semispazi chiusi** e di conseguenza come l'**insieme delle soluzioni di un sistema di m disequazioni a n variabili**

$$P = \{x \in \mathbb{R}^n : \ Ax \leq b\}, \ \ A \in M^{m \times n} \text{ e } x, b \in \mathbb{R}^n \tag{9}$$

Un **poliedro** è un insieme convesso, in quanto i **semispazi chiusi** sono a loro volta insiemi convessi e l'intersezione di n insiemi convessi restituisce comunque un insieme convesso. Inoltre, i poliedri sono divisibili in due classi: **Poliedri *limitati*** e **Poliedri *illimitati***

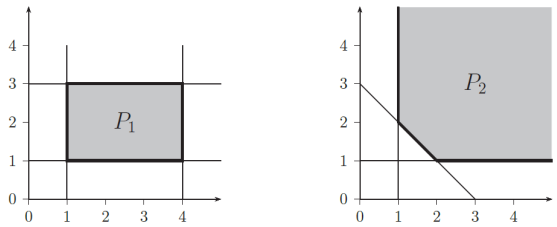


Figura 2.4: P1 è un poliedro limitato, P2 è un poliedro non limitato.

Tutti i poliedri, illimitati o meno, sono **chiusi**, ma affinché un poliedro possa essere detto limitato, bisogna che esista una palla di raggio R, centrata in x che contiene tutto l'insieme. Possiamo anche definire un lemma molto importante a riguardo dei **coni poliedrici**

Lemma 1.1. Se P è un *cono poliedrico* allora esiste una matrice Q tale che

$$P = \{x \in \mathbb{R}^n : Qx \leq 0\} \tag{10}$$

Teorema di rappresentazione dei poliedri

Procediamo adesso a definire un importante teorema che fornisce una caratterizzazione dei poliedri. Premettiamo alcuni lemmi

Lemma 1.2. L'involucro conico di un insieme finito di punti $K = \{e_1, \ldots, e_p\}$ è un ***cono poliedrico***. Di conseguenza esiste una matrice Q tale che

$$\text{cono}(K) = \{x \in \mathbb{R}^n : Qx \leq 0\} \tag{11}$$

Quindi $\text{cono}(e_1, \ldots, e_p)$ è un cono poliedrico e possiamo quindi scriverlo nella forma

$$\{x \in \mathbb{R}^n : Qx \leq 0\} \tag{12}$$

Lemma 1.3. Un cono poliedrico $P = \{x \in \mathbb{R}^n : Ax \leq 0\}$ è *l'involucro conico* di un insieme finito dei suoi punti.

Grazie a questi due lemmi possiamo giungere a definire alcuni tra i risultati più importanti a riguardo dei poliedri.

Theorem 1.4. (Teorema di Weyl) Dato un poliedro P esistono un sottoinsieme finito $V = \{v_1, \ldots, v_s\} \in P$ ed un insieme finito $E = \{e_1, \ldots, e_p\} \subset \mathbb{R}^n$, tali che

$$P = \text{conv}(V) + \text{cono}(E) \tag{13}$$

Intuitivamente il teorema può essere analizzato nei due membri che lo compongono:

- $\text{cono}(E)$: Il cono è come prendere delle direzioni, rappresentate dai vettori E, e spararle all'infinito. Un insieme conico contiene tutte le combinazioni coniche dei vettori in E, cioè permette di creare direzioni che si estendono all'infinito.

- conv(V): L'insieme convesso è la parte limitata del poliedro. L'involucro convesso è l'insieme convesso più piccolo che contiene tutti i vettori di V, potremmo visualizzare l'involucro convesso come un elastico che viene messo intorno agli elementi dell'insieme V.

L'utilità di questo teorema risiede in due suoi aspetti, il primo aspetto è la possibilità di poter rappresentare in modo estremamente semplice, un qualsiasi poliedro. Il secondo aspetto verrà enunciato in seguito.

Domanda Orale

Dato un poliedro scritto nella forma $P = \text{conv}(V) + \text{cono}(E)$ *quali sono le dimensioni della matrice* A *(relativa alla rappresentazione del poliedro come sistema* $Ax \leq b$*)?* Si potrebbe essere indotti a pensare che la risposta sia $m \times n$, ma questa sarebbe un'affermazione errata. Non possiamo determinare con certezza la dimensione della matrice A, poiché non siamo in grado di dedurre a priori il numero esatto di vincoli necessari per rappresentare il poliedro.

Per dimostrare l'utilità del teorema appena enunciato proviamo a svolgere alcune domande che capitano sovente all'orale.

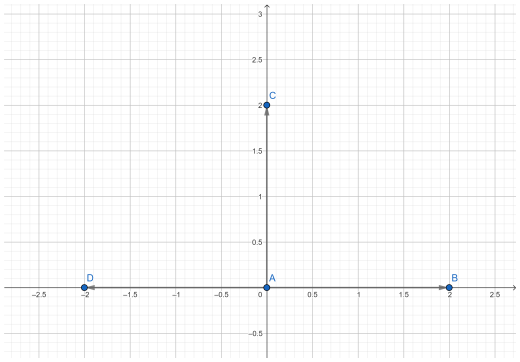


Figura 2.5: Poliedro con $|V| = 1$ e $|E| = 3$

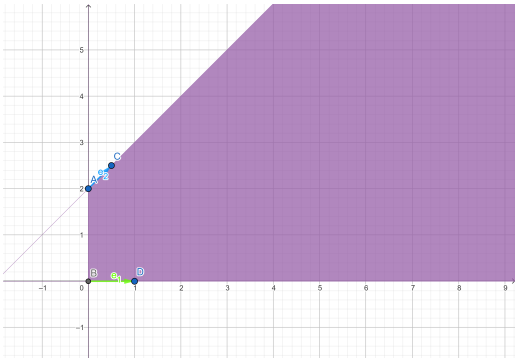


Figura 2.6: Poliedro con $|V| = 2$ e $|E| = 2$

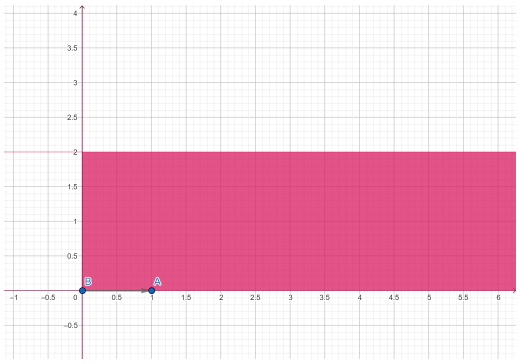


Figura 2.7: Poliedro con $|V| = 2$ e $|E| = 1$

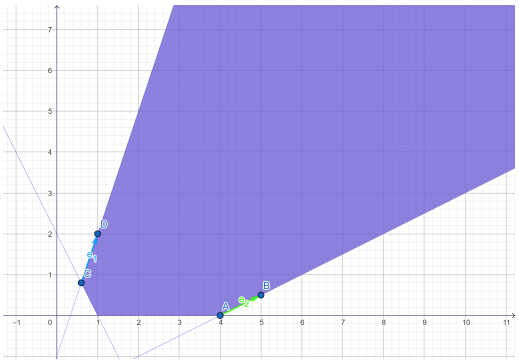


Figura 2.8: Poliedro con $|V| = 3$ e $|E| = 2$

2 Condizioni di ottimalità

Un problema PL della forma

$$\begin{cases} \max c^T x \\ x \in P = \{x \in \mathbb{R}^n : Ax \leq b\} \end{cases} \tag{14}$$

viene chiamato **problema primale in forma standard**. Il valore ottimo del problema $(\mathcal{P})$ verrà indicato con $v(\mathcal{P})$.

Nell'introduzione abbiamo anche visto come ogni problema PL si può trasformare, in modo equivalente, in uno in forma primale standard cambiando

$$\min c^T x \text{ in } -\max -c^t x \tag{15}$$

e trasformando un vincolo di uguaglianza in due vincoli di disuguaglianza e un vincolo di $\geq$ in uno di $\leq$ moltiplicato per -1. Ad esempio, il problema

$$\begin{cases} \min 3x_1 - 5x_2 + 6x_3 \\ x_1 - 2x_2 + 4x_3 = 7 \\ x_2 \geq 0 \end{cases} \tag{16}$$

può essere trasformato come segue

$$\begin{cases} -\max -3x_1 + 5x_2 - 6_3 \\ x_1 - 2x_2 + 4x_3 \leq 7 \\ -x_1 + 2x_2 - 4x_3 \leq -7 \\ -x_2 \leq 0 \end{cases} \tag{17}$$

Come mai c^T ? Prendiamo il primo problema che abbiamo descritto nella sezione. Abbiamo che

$$\begin{aligned} c &= \begin{bmatrix} 3 \\ -5 \\ 6 \end{bmatrix} \text{ è il vettore dei coefficienti della funzione obiettivo} \\ x &= \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \text{ è il vettore delle variabili decisionali} \end{aligned} \tag{18}$$

Quindi, la funzione obiettivo non è altro che il prodotto scalare tra c^T e x, quindi

$$c^T x = \begin{bmatrix} 3 & -5 & 6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = 3x_1 - 5x_2 + 6x_3 \tag{19}$$

Tornando alle soluzioni ottime, stabiliamo ora alcune condizioni che garantiscano che un punto ammissibile di un problema PL sia ottimo.

Theorem 2.1. (Caratterizzazione elementare dell'ottimalità)

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

Consideriamo ora un esempio di un problema in due variabili

$$\begin{cases} \max x_1 + x_2 \\ x_1 \geq 2 \\ x_2 \geq 2 \\ -x_1 + x_2 \leq 6 \\ x_1 \leq 4 \end{cases} \tag{20}$$

Questo problema potrebbe essere risolto geometricamente, andando a disegnare il poliedro ammissibile e le cosiddette linee di **isoguadagno** (o di **isocosto** per problemi di minimo), ossia le rette su cui la funzione obiettivo assume valori costanti.

Se scriviamo la nostra funzione obiettivo nella forma $c^T x$, otteniamo il vettore $c = (1,1)^T$. Le rette di isoguadagno formano un fascio di rette di equazione

$$c^T x = x_1 + x_2 = k \tag{21}$$

dove $k \in \mathbb{R}$ rappresenta il valore della funzione obiettivo su quella retta. Le rette di isoguadagno sono tra loro parallele ed ortogonali rispetto al vettore c. Il valore k cresce nel verso indicato da c.

Per risolvere un problema di PL di massimo bisogna trovare tra tutte le rette di isoguadagno che intersecano P, quella che ha il massimo valore di k ed una soluzione ottima è data dall'intersezione di tale linea di isoguadagno con il poliedro ammissibile.

Passiamo ora a definire e dimostrare uno dei teoremi più importanti della programmazione lineare.

Theorem 2.2. (Teorema fondamentale della PL) Supponiamo P sia rappresentato come

$$P = \text{conv}(v_1, \dots, v_m) + \text{cono}(e_1, \dots, e_p) \;\; v_1, \dots, v_k \in P \neq \emptyset \tag{22}$$

Se il problema $(\mathcal{P})$ ha valore ottimo finito, allora esiste un $k \in \{1, \dots, m\}$ tale che v_k è una soluzione ottima di $(\mathcal{P})$. La dicitura *se il problema ha valore ottimo finito* implica una serie di condizioni

- Il poliedro **non è vuoto**.
- $\text{vert}(P) \neq \emptyset$.
- La funzione obiettivo $c^T x < +\infty$.

Dimostrazione Dal teorema di Weyl abbiamo che $P = \text{conv}(V) + \text{cono}(E)$ abbiamo che un qualsiasi punto $x \in P$ può essere espresso nel seguente modo

$$x = \sum_{i=1}^{m} \lambda_i v_i + \sum_{j=1}^{n} \mu_i e_i \tag{23}$$

dove valgono i seguenti vincoli

$$\begin{aligned} &\lambda \in [0,1], \quad \sum_{i=1}^{m} \lambda_i = 1 \\ &\mu \geq 0 \end{aligned} \tag{24}$$

Abbiamo che per ogni $x \in P$ otteniamo

$$c^T x = \sum_{i=1}^{m} \lambda_i c^T v_i + \sum_{j=1}^{n} \mu_i c^T e_i \tag{25}$$

Il problema ammette soluzione finita, di conseguenza, essendo $\mu \geq 0$, è necesario che $c^T e_i$ sia negativo per tutte le i, altrimenti facendo $\mu \to +\infty$ si avrebbe che $x \to +\infty$ e si andrebbe a creare un assurdo.

$$\begin{aligned} c^T x &= \sum_{i=1}^{m} \lambda_i c^T v_i + \sum_{j=1}^{n} \mu_i c^T e_i \\ &\leq \sum_{i=1}^{m} \lambda_i c^T v_i \\ &\leq \sum_{i=1}^{m} \lambda_i \max_{1 \leq i \leq m} c^T v_k \\ &= \left(\max_{1 \leq i \leq m} c^T v_k \right) \sum_{i=1}^{m} \lambda_i \\ &= \max_{1 \leq i \leq m} c^T v_k \end{aligned} \tag{26}$$

e quindi

$$\max_{1\leq i\leq m} c^T v_k \geq c^T x \tag{27}$$

Poiché v^k appartiene al poliedro P si ottiene anche che

$$c^T v_k \leq \max_{1\leq i\leq m} c^T x \tag{28}$$

e ci conseguenza v_k è soluzione ottima del problema $(\mathcal{P})$.

Essendo una funzione lineare, essa tenderà a salire o scendere in modo monotono. Quindi, necessariamente, la soluzione ottima, se esiste e se finita, si dovrà trovare sul "confine" del poliedro, il quale è rappresentato dai vertici di quest'ultimo. Da questo teorema possiamo estrarre un corollario

> Se il problema $(\mathcal{P})$ ha valore ottimo finito e lineal$(P) = \{0\}$, allora esiste **almeno** un **vertice ottimo**.

Intuitivamente, sappiamo che se lineal$(P) = \{0\}$ non esistono direzioni nelle quali il poliedro si espande infinitamente, inoltre, dal teorema sulle condizioni di ottimalità, sappiamo che se $c \neq 0$, allora la soluzione non può trovarsi "all'interno" del poliedro e dunque, essa dovrà necessariamente trovarsi su uno dei vertici.

Possono esistere anche delle situazioni in cui il problema ha soluzione ottima finita, anche il assenza di vertici ottimi.

3 Teoria della dualità

Dato un problema primale in forma standard

$$\begin{cases} \max c^T x \\ x \in P = \{x \in \mathbb{R}^n : Ax \leq b\} \end{cases} \tag{29}$$

possiamo associare il seguente problema di PL

$$\begin{cases} \min y^T b \\ y \in D = \{x \in \mathbb{R}^m : y^T A = c^T, y \geq 0\} \end{cases} \tag{30}$$

il quale prenderà il nome di **problema duale in forma standard**. Il valore ottimo del problema $(\mathcal{D})$ sarà detto $v(\mathcal{D})$.

Un'altra possibile notazione per il **duale** in forma standard è

$$\begin{cases} \min b^T y \\ A^T y = c \\ y \geq 0 \end{cases} \tag{31}$$

Visto che ad ogni problema di programmazione lineare possiamo associare un corrispettivo problema primale in forma standard $(\mathcal{P})$, allora ogni problema PL ha anche il suo corrispettivo problema duale in forma standard $(\mathcal{D})$.

Un problema duale è un problema che ha come scopo principale quello di ottimizzare le risorse necessarie a soddisfare il problema primale.

Osserviamo che ogni problema di PL si può trasformare in un duale standard di tipo $(\mathcal{D})$. Infatti una disuguaglianza

$$A_i x \leq b_i \tag{32}$$

può essere trasformata in una disuguaglianza

$$A_i x + s_i = b_i \tag{33}$$

aggiungendo i vincoli $s_i \geq 0$, dove s_i sono dette **variabili di scarto**. Queste variabili di scarto sono ottenute andando a sottrarre il valore di $A_i x$ a quello di b_i e il fatto che siano maggiori di 0 garantisce che, sia sempre rispettata la condizione originale $A_i x \leq b_i$.

Inoltre, ogni numero reale è differenza di due numeri non negativi, e quindi si può sempre introdurre il vincolo di positività sulle variabili spezzando ogni variabile non vincolata in segno nella differenza di due variabili vincolate in segno

$$x = x_1 - x_2 \quad x_1, x_2 \geq 0 \tag{34}$$

Esempio intuitivo sulla teoria della dualità In una azienda è necessario massimizzare il guadagno della vendita di due tessuti, chiamati rispettivamente tessuto A e tessuto B. Denotiamo come x_a la quantità di tessuto A prodotto e x_b la quantità di tessuto B prodotto. Denotiamo inoltre rispettivamente c_a, c_b come la quantità dei tessuti prodotti di tipo A e tipo B. Assumendo l'esistenza di dei vincoli (non sono importanti per il nostro esempio) problema primale sarebbe quindi

$$\begin{cases} \max c^T x \\ 4x_a + 5x_b \leq 3 \\ 3x_a + 6x_b \leq 10 \\ x_a, x_b \geq 0 \end{cases} \tag{35}$$

Il rispettivo problema duale sarebbe un problema il cui obiettivo non è più massimizzare il guadagno, ma minimizzare il costo delle nostre materie prime.

Possiamo anche definire un set di regole che permettono di costruire il duale di un generico problema di PL senza dover passare per il primale stadard.

1. Se il primale è un problema di **massimo** (**minimo**), il duale è un problema di **minimo** (**massimo**).

2. Ad ogni vincolo primale $A_i x = b_i$ è associata una variabile duale y_i, non vincolata in segno e di costo uguale a b_i (quindi moltiplicata per b_i).

3. Se il primale è un problema di massimo (minimo), allora ad ogni vincolo $A_i x \leq b$ è associata una variabile duale $y_i \geq 0$ ($y_i \leq 0$), mentre ad ogni vincolo primale $A_i x \geq b_i$ è associata una variabile duale $y_i \leq 0$ ($y_i \geq 0$). Il costo di y_i è uguale a quello di b_i.

4. Ad ogni variabile x_i non vincolata in segno è associato un vincolo di uguaglianza del problema duale. Nel vincolo duale, i coefficienti delle variabili sono i coefficienti con cui la variabile x_i compare nei vincoli primali ed il termine noto è il costo di x_i.

5. Se il duale è un problema di minimo (massimo), allora ad ogni variabile primale $x_i \geq 0$ è associato un vincolo di tipo $\geq$ ($\leq$), mentre ad ogni variabile $x_i \leq 0$ è associato un vincolo di tipo $\leq$ ($\geq$). Nel vincolo duale, i coefficienti delle variabili sono i coefficienti con cui la variabile x_i compare nei vincoli primali e il termine noto è il costo di x_i.

Prendiamo un esempio. Considerato il problema primale

$$\begin{cases} \max 3x_1 - 4x_2 + 2x_3 \\ 6x_1 - x_2 \geq 1 \\ 2x_1 - 5x_3 \leq 5 \\ x_1 + 3x_2 + 4x_3 = 10 \\ x_3 \geq 0 \end{cases} \tag{36}$$

definiamo il problema duale corrispettivo. Il primo passo è quello di determinare il tipo di problema duale; Dalle regole che abbiamo definito sopra sappiamo che: *essendo il problema primale, un problema di massimo, allora il duale rispettivo è un problema di minimo.*

Inoltre, dalla definizione di duale che abbiamo visto in precedenza sappiamo che la funzione obiettivo associata è definita come

$$y^T b \tag{37}$$

y^T è il vettore riga delle variabili duali. Per trovare le nostre variabili duali dobbiamo andare a verificare la presenza di un vincolo primale $A_i x = b_i$, che nel nostro caso è dato dal vincolo

$$x_1 + 3x_2 + 4x_3 = 10 \tag{38}$$

dalle regole per la costruzione del duale sappiamo che

> Ad ogni vincolo primale $A_i x = b_i$ è associata una variabile duale y_i, non vincolata in segno e di costo uguale a b_i (quindi moltiplicata per b_i).

nel nostro caso ci sono tre possibili vincoli che dal principio di equivalenza visto in precedenza possono essere portati nella forma specificata dalla proprietà, di conseguenza il vettore colonna y equivale a

$$y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \text{ una riga per ogni vincolo} \tag{39}$$

il vincolo di non negatività non viene considerato in quanto implicito. Ad ognuna di queste variabili possiamo associare un costo, il quale è il corrispettivo termine noto della disuguaglianza, ad esempio

- Al primo vincolo associamo la variabile duale y_1 di costo pari a 1.
- Al secondo vincolo associamo la variabile duale y_2 di costo uguale a 5.
- Al terzo vincolo associamo la variabile duale y_3 di costo uguale a 10.

Quindi, la nostra funzione da minimizzare è

$$y_1 + 5y_2 + 10y_3 = y^T b = \begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix} \begin{bmatrix} 1 \\ 5 \\ 10 \end{bmatrix} \tag{40}$$

dove b è il vettore dei costi. Dopo aver calcolato la funzione obiettivo Posso procedere con la descrizione dei vincoli duali. Dalle regole sulla costruzione del duale sappiamo che

> Ad ogni variabile x_i non vincolata in segno è associato un vincolo di uguaglianza del problema duale. Nel vincolo duale, i coefficienti delle variabili sono i coefficienti con cui la variabile x_i compare nei vincoli primali ed il termine noto è il costo di x_i.

Abbiamo quindi tre variabili primali e di conseguenza tre vincoli

$$\begin{aligned} 6y_1 + 2y_2 + 1y_3 &= 3 \\ -y_1 + 3y_3 &= -4 \\ -5y_2 + 4y_3 &= 2 \end{aligned} \tag{41}$$

Dall'ultima regola deduciamo sappiamo che

> Se il duale è un problema di minimo (massimo), allora ad ogni variabile primale $x_i \geq 0$ è associato un vincolo di tipo $\geq$ ($\leq$), mentre ad ogni variabile $x_i \leq 0$ è associato un vincolo di tipo $\leq$ ($\geq$). Nel vincolo duale, i coefficienti delle variabili sono i coefficienti con cui la variabile x_i compare nei vincoli primali e il termine noto è il costo di x_i.

E deduciamo anche quindi l'ultimo vincolo del nostro problema

$$-5y_2 + 4y_3 \geq 2 \tag{42}$$

Scriviamo il problema duale associato

$$\begin{cases} \min y_1 + 5y_2 + 10y_3 \\ 6y_1 + 2y_2 + 1y_3 = 3 \\ -y_1 + 3y_3 = -4 \\ -5y_2 + 4y_3 = 2 \\ y_1 \leq 0 \\ y_2 \geq 0 \end{cases} \tag{43}$$

Esiste un teorema che pone una relazione tra problema primale e problema duale.

Theorem 3.1. Il duale di $(\mathcal{D})$ è $(\mathcal{P})$

Esistono anche due corollari che può essere utile enunciare

- *Se* $P \neq \emptyset$ *e* $D = \emptyset$, allora $v(\mathcal{P}) = v(\mathcal{D}) = +\infty$.
- *Se* $P = \emptyset$ *e* $D \neq \emptyset$, allora $v(\mathcal{P}) = v(\mathcal{D}) = -\infty$.

Possiamo giustificare questi due corollari nel seguente modo

- Se il duale non ha soluzioni ammissibili, significa che il sistema vincolato descritto dal duale non può limitare la crescita della funzione obiettivo primale. Il duale, di solito, serve a fornire un limite superiore o inferiore al valore del primale. Se il duale non esiste, allora non ci sono limiti, e quindi il valore della funzione obiettivo del primale può crescere all'infinito.
- Se il primale non ha soluzioni ammissibili, è impossibile soddisfare i vincoli del problema primale, il che porta a considerare il valore del problema primale come estremamente basso. Dato che il duale ha soluzioni ammissibili, queste soluzioni impongono limiti sulla funzione obiettivo primale, spingendola verso $-\infty$

Passiamo ora a enunciare due **risultati fondamentali sulla teoria della dualità**

Theorem 3.2. (Dualità debole) Se i poliedri P e D **non sono vuoti**, allora

$$c^T x \leq y^T b \tag{44}$$

Il teorema della dualità **debole** ci dice che il valore della funzione obiettivo del duale, calcolato in un qualsiasi punto della regione ammissibile della funzione duale costituisce un **confine superiore** per l'ottimo del primale. Viceversa, ogni valore della funzione obiettivo del primale calcolata nella regione ammissibile costituisce un **confine inferiore** per l'ottimo del primale.

Assumiamo di avere un'azienda che voglia massimizzare il guadagno dalla vendita di alcuni prodotti. Dal teorema della **dualità debole**, sappiamo che, per qualsiasi soluzione ammissibile dei problemi primale $(\mathcal{P})$ e duale $(\mathcal{D})$, vale la relazione:

$$c^T x \leq y^T b$$

dove x è una soluzione ammissibile del primale e y una soluzione ammissibile del duale.

Ora, supponiamo di avere la soluzione ottima $\overline{x}$ del primale $(\mathcal{P})$ e la soluzione ottima $\overline{y}$ del duale $(\mathcal{D})$. Utilizzando il teorema della dualità debole, possiamo affermare che:

$$c^T \overline{x} \leq \overline{y}^T b$$

Questo risultato è estremamente utile, poiché stabilisce un **limite superiore** al valore ottimale della funzione obiettivo del primale. Ci dice, infatti, che la funzione obiettivo del primale non può crescere oltre il valore ottimo del duale.

Passiamo ora a enunciare un altro importante risultato della **teoria della dualità**

Theorem 3.3. (Dualità forte) Se i poliedri P e D non sono vuoti, allora

$$-\infty < v(\mathcal{P}) = v(\mathcal{D}) < +\infty \tag{45}$$

Grazie a questo teorema abbiamo dimostrato che il problema primale e il problema duale hanno lo stesso valore ottimo quando almeno uno dei due poliedri non è vuoto.

Possiamo riassumere in un'unica tabella le relazioni che esistono tra i valori ottimi del duale e del primale

	$D \neq \emptyset$	$D = \emptyset$
$P \neq \emptyset$	$v(\mathcal{P}) = v(\mathcal{D}) \in \mathbb{R}$	$v(\mathcal{P}) = v(\mathcal{D}) = +\infty$
$P = \emptyset$	$v(\mathcal{P}) = v(\mathcal{D}) = -\infty$	$v(\mathcal{P}) = -\infty$ $v(\mathcal{D}) = +\infty$

Una delle possibili utili applicazioni della teoria della dualità è la seguente

Theorem 3.4. (Scarti complementari) Supponiamo che $\overline{x}$ e $\overline{y}$ siano ammissibili e rispettivamente per i problemi $(\mathcal{P})$ e $(\mathcal{D})$. Allora si ha

$$\overline{x}, \overline{y} \text{ ottime} \iff \overline{y}^T(b - A\overline{x}) = 0 \tag{46}$$

quando $\overline{y}^T(b - A\overline{x}) = 0$ si dice che $\overline{x}$ e $\overline{y}$ sono in scarti complementari.

Possiamo analizzare anche il significato di questo teorema. La condizione $b - A\overline{x} = 0$ si può verificare in un solo caso, quando $A\overline{x} = b$ e di conseguenza, quando b si trova sul bordo del poliedro.

4 Complementi di algebra della PL

Dal **teorema del rango**, studiato in **algebra lineare**, sappiamo che: Se il rango di una matrice è minore del numero di colonne, allora il sistema lineare $Ax = 0$ **ammette infinite soluzioni**. Quindi, se il rango della matrice è minore del numero di colonne, si ha che lineal$(P) \neq \{0\}$, che equivale a dire che P non contiene nessuno vertice.

Definition 4.1. (Base) Siano date due matrici, una matrice A e una matrice b tale che $Ax \leq b$, con $m \geq n$. Definiamo un insieme $B \subseteq \{1, \ldots, m\}$ che prenderà il nome di base, tale che la sotto-matrice A_B, ottenuta da A estraendo le righe A_i con $i \in B$ sia invertibile ($\det(A) \neq 0$). La matrice A_B prenderà il nome di matrice di base. Definiamo inoltre la sottomatrice b_B ottenuta da b estraendo le righe b_i con $i \in B$. Definiamo inoltre come N l'insieme degli indici non appartenenti a B.

Data una base B definiamo due tipologie di soluzione

- **Soluzione di base primale**: definita come la soluzione del sistema

$$A_B\overline{x} = b_B \tag{47}$$

- **Soluzione di base duale**: definita come la soluzione del sistema

$$\overline{y} = \begin{bmatrix} \overline{y}_B \\ \overline{y}_N \end{bmatrix} \quad \overline{y}_B^T A_B = c^T \iff \overline{y}_B = cA_B^{-1}, \ \overline{y}_N = 0 \tag{48}$$

Le soluzioni di base di dividono in due categorie:

- Soluzioni **ammissibili o non ammissibili**
  - **Soluzioni *ammissibili***: se appartengono al poliedro.
  - **Soluzioni *non ammissibili***: se non appartengono al poliedro.
- Soluzioni **degeneri o non degeneri**:
  - **Soluzione di base *degenere***: una soluzione di base degenere è una soluzione di base in cui non solo le variabili fuori base valgono 0, ma anche una (o più) variabili in base. Intuitivamente, possiamo vedere una soluzione di base **degenere** è una soluzione di base in cui ci sono più vincoli soddisfatti che variabili di base necessarie per definirla e di conseguenza, ottenibile attraverso più basi diverse.
  - **Soluzione di base *non degenere***: una soluzione di base **non degenere** è una soluzione di base ottenibile da una e una sola base.

Possiamo avvalerci di una veloce tabella per valutare in quale di queste categorie rientra la nostra soluzione

	$\bar{x}$	$\bar{y}$
ammissibile	per ogni $i \in N$ si ha $A_i\bar{x} \leq b_i$	per ogni $i \in B$ si ha $\bar{y}_i \geq 0$
non ammissibile	esiste $i \in N$ tale che $A_i\bar{x} > b_i$	esiste $i \in B$ tale che $\bar{y}_i < 0$
degenere	esiste $i \in N$ tale che $A_i\bar{x} = b_i$	esiste $i \in B$ tale che $\bar{y}_i = 0$
non degenere	per ogni $i \in N$ si ha $A_i\bar{x} \neq b_i$	per ogni $i \in B$ si ha $\bar{y}_i \neq 0$

Passiamo adesso a enunciare un teorema molto utile per dare una caratterizzazione algebrica dei vertici di un poliedro.

Theorem 4.1. (Caratterizzazione dei vertici) Un punto $x \in P$ del poliedro è un **vertice** se e solo se **è una soluzione di base ammissibile**.

Data una coppia di soluzioni di base $\overline{x}$ e $\overline{y}$ rispettivamente di $(\mathcal{P})$ e di $(\mathcal{D})$, si dice che esse sono **complementari** se e solo se

Lemma 4.2. Due soluzioni di base $\overline{x}$ e $\overline{y}$ complementari sono in **scarti complementari**.

Da questo lemma e dal teorema degli scarti complementari possiamo giungere ad un teorema fondamentale di questo paragrafo

Theorem 4.3. (Condizioni sufficienti di ottimalità per soluzioni di base) Date due soluzioni di base **complementari** $\overline{x}$ e $\overline{y}$ si ha

$$\begin{bmatrix} \overline{x} \text{ è ammissibile per } (\mathcal{P}) \\ \overline{y} \text{ è ammissibile per } (\mathcal{D}) \end{bmatrix} \iff \begin{bmatrix} \overline{x} \text{ è ottima per } (\mathcal{P}) \\ \overline{y} \text{ è ottima per } (\mathcal{D}) \end{bmatrix} \tag{49}$$

Consideriamo il problema duale. Nel paragrafo precedente abbiamo detto che possiamo esprimere il vettore c come combinazioe lineare dei vettori riga della matrice A con coefficienti le componenti base di y. La soluzione duale $\overline{y}$ è ammissibile se e solo se le componenti in base sono maggiori o uguali a 0, ossia se c è una combinazione conica delle righe della matrice A o, in altri termini, se c appartiene al cono definito dalle righe di base. Quando una delle componenti di base è nulla e quindi c può essere scritto come combinazione lineare di un sottoinsieme di queste righe, la soluzione è degenere.

Algoritmo per il calcolo dei vertici

Sia data una matrice A, divisa rispettivamente in una **sottomatrice di base** A_B e una **sottomatrice non di base**; scriviamo A come

$$A = \begin{bmatrix} A_B \\ A_N \end{bmatrix} \tag{50}$$

sotto l'ipotesi che $\det(A_B) \neq 0$. Andiamo successivamente a applicare l'algoritmo per il calcolo dei vertici: risolviamo il sistema della sottomatrice a nella forma

$$A_B x = b_B \tag{51}$$

Assumiamo di avere un numero m di vincoli, sappiamo che per ottenere una soluzione di base devo prendere n vincoli. Definiamo Q come l'insieme delle soluzioni di base, allora il numero di possibili soluzioni di base è dato da

$$|Q| = \binom{m}{n} = \frac{m!}{n! \cdot (m-n)!} \tag{52}$$

l'algoritmo è estremamente complesso da eseguire, in quanto, più aumentano i vincoli, maggiore è la complessità totale dell'algoritmo

Capitolo 3

Algoritmo del simplesso

Nell'ultimo capitolo sulla programmazione lineare andiamo a descrivere l'algoritmo per la risoluzione di un problema di programmazione lineare: **il simplesso**. Andremmo a descrivere due versioni di simplesso: **simplesso primale** e **simplesso duale**.

Il simplesso è un algoritmo nato intorno agli anni 90, studiato e ideato da Danzig

1 Simplesso primale

Descriviamo l'algoritmo del simplesso primale per la risoluzione di un problema di programmazione lineare in **forma primale standard**

$$\begin{cases} \max c^T x \\ Ax \leq b \end{cases} \tag{1}$$

in cui la matrice $A \in M^{m \times n}$, ha rango uguale a n. L'algoritmo parte da una soluzione primale ammissibile (quindi un vertice). Se la soluzione di base complementare (quindi la duale) è ammissibile, allora esse sono rispettivamente ottime per il primale e il duale, altrimenti si cambia base. Il cambio di base è definito in modo che:

- Se la nuova soluzione di base è diversa da quella vecchia, il valore della funzione obiettivo cresce.
- Se la nuova soluzione di base è uguale a quella vecchia, si evita di ciclare sulle stesse basi con opportuni sistemi di anticiclo.

Quindi, ogni qualvolta ci troviamo ad una soluzione di base ammissibile primale, ma che non è ammissibile duale ci muoviamo attraverso i vertici del nostro poliedro, lungo la direzione, tra le variabili **non di base**, che migliora la funzione obiettivo, fintanto che non si trova la soluzione ottima. L'algoritmo può essere sintetizzato nei seguenti passi

1. Trovare una base B che genera una soluzione di base ammissibile.
2. Calcola la soluzione di base primale $\overline{x}$ e la soluzione duale $\overline{y}$. Dopo di ché:
   - Se la soluzione di base duale $\overline{y} \geq 0$ si ferma l'iterazione, in quanto le due soluzioni sono complementari e ammissibili entrambi e di conseguenza, per teorema, anche ottime.
   - Altrimenti si calcola l'indice usciente

     $$h := \min\{i \in B : \overline{y}_i < 0\} \tag{2}$$

     e si pone $W := -A_B^{-1}$ e si indica con W^h la h-esima colonna di W.
3. Si valuta $A_i W^h \leq 0 \forall i \in N$:

- Se la condizione è verificata si termina l'iterazione; $(\mathcal{P})$ ha valore ottimo $+\infty$ e $(\mathcal{D})$ è vuoto. Il motivo è che lungo la semiretta mi posso spostare all'infinito, infatti, per ogni possibiel valore di $\lambda \geq 0$ abbiamo che

$$A\overline{x} + \lambda A_i W^k \in P \ \ \forall \lambda \in [0, +\infty) \tag{3}$$

  quindi il problema non ha soluzione ottima.

- Altrimenti si definisce

$$\theta := \min\left\{ \frac{b_i - A_i\overline{x}}{A_i W^h} : i \in N, A_i W^h > 0 \right\} \tag{4}$$

  dal quale si calcola l'indice entrante

$$k := \min\{i \in N : A_i W^h > 0, \frac{b_i - A_i\overline{x}}{A_i W^h} = \theta\} \tag{5}$$

4. Si aggiona la base

$$B := B\text{differenza}\{k\} \cup \{h\} \tag{6}$$

   per poi tornare al passo 2.

Andiamo adesso a spiegare passo passo le varie fasi dell'algoritmo

Vertice di partenza : supponiamo di avere una **base** B di partenza a cui è associato un vertice $\overline{x} \in P$ definito come

$$\overline{x} = A_B^{-1} b_B \tag{7}$$

Test di ottimalità : Costruiamo la soluzione **complementare** di $\overline{x}$:

$$\overline{y}_B^T = c^T A_B^{-1} \longrightarrow \overline{y} \tag{8}$$

verifico se $y_B \geq 0$ per tutte le componenti. Se la condizione è verificata l'algoritmo termina, è stata trovata la soluzione ottima per i rispettivi problemi.

Quando invece ho un coefficiente y_i della soluzione di base duale associata negativo, vuol dire che stiamo violando un vincolo del duale e che la funzione obiettivo può essere ancora migliorata; in particolare stiamo dicendo che esiste un vincolo che non sta contribuendo alla crescita della funzione obiettivo. Dobbiamo quindi fare un cambio di base

Cambio di base Definiamo la seguente matrice

$$W = -A_B^{-1} \tag{9}$$

prendiamo la colonna h-esima della matrice andiamo a definire la seguente semiretta

$$\overline{x}(\lambda) = \overline{x} + \lambda W^h, \quad \lambda \geq 0 \tag{10}$$

Essa ha origine nel vertice $\overline{x}$. Visto che, da teorema degli scarti complementari, la soluzione del nostro problema **non è ottima**, vogliamo lasciare il vertice in cui ci troviamo e spostarci verso un altro. L'*altro* vertice non può però essere scelto a caso, occorre predisporre un criterio. Nel nostro caso, il criterio è che andiamo a spostarci lungo lo spigolo in cui il valore della funzione obiettivo aumenta. In generale, data la base iniziale A_B vogliamo ottenere una nuova base con $n-1$ indici in comune e un indice diverso. Dobbiamo quindi individuare

- un indice uscente h, che vogliamo sostituire.
- un indice entrante k, quello che andrà a sostituire h

Indice uscente Andiamo a calcolare la variazione della nostra funzione obiettivo lungo la semiretta che abbiamo definito nel cambio di base

$$c\overline{x}(\lambda) = c(\overline{x} + \lambda W^h) = c\overline{x} + \lambda cW^h \tag{11}$$

il primo termine è la funzione obiettivo calcolata nel vertice corrente, il secondo termine determina la variazione della funzione obiettivo lungo la semiretta. Se il secondo termine è positivo, la funzione cresce in valore. Ricordiamo inoltre che

$$cW^i = c(-A_B^{-1})^i = -c(A_B^{-1})^i = -\overline{y}_i \tag{12}$$

Per ottenere la funzione obiettivo è necessario un indice h tale per cui $c \cdot W^h > 0$ $(-y_i > 0)$, quindi, un indice per cui $\overline{y}_i < 0$. L'idea che potrebbe venire in mente è quella di considerare la componente più piccola tra quelle negative, in modo da massimizzare il più possibile

$$\begin{aligned} &\theta = \{\overline{y}_i < 0,\ i \in B\} \\ &h = \min\{i \in B : \overline{y}_i = \theta\} \end{aligned} \tag{13}$$

Però, per via della presenza delle regole di **anticiclo di Bland**, che permettono di evitare ci ciclare all'infinito sul caso degenere, andiamo a scegliere l'indice minore tra quelli che violano le condizioni di ammissibilità del duale.

$$h = \min\{i \in B : \overline{y}_i < 0\} \tag{14}$$

Visto che $\overline{x}$ non è sicuramente l'ottimo, siamo sicuri che esista una $\overline{y}_i < 0$ con $i \in B$, quello che noi facciamo è prendere il primo tra i possibili.

Indice entrante Abbiamo quindi trovato la semiretta lungo la quale la funzione obiettivo cresce e, di conseguenza, lungo la quale noi ci spostiamo. Continuiamo a muoverci su quella semiretta e andiamo a valutare quali punti della nostra semiretta appartengono al poliedro, quindi, quali rispettano la disuguaglianza

$$A_i\overline{x}(\lambda) \leq b \tag{15}$$

Considerando prima i vincoli di base abbiamo che sono tutti rispettati, infatti il prodotto A_iW^h può valere -1, nel caso peggiore e 0, nel resto dei casi; possiamo esprimere questo fatto come

$$A_i\overline{(\lambda)} = A_i\overline{x} + \lambda A_iW^h = \begin{cases} b_i & i \in B\backslash\{h\} \\ b_i - \lambda \leq b_i & i = h \end{cases} \tag{16}$$

Quindi, i vincoli di base sono tutti soddisfatti su tutti i punti della semiretta; dobbiamo verificare che lo siano anche quelli non di base. In generale, la condizione che deve essere sempre rispettata è che

$$A_i\overline{x} + \lambda(A_iW^h) \leq b_i \quad \forall i \in N \tag{17}$$

Dobbiamo quindi capire quali valori di lambda sono ammissibili; lo facciamo grazie ad alcuni passaggi algebrici

$$\begin{aligned} &A_i\overline{x} + \lambda(A_iW^h) \leq b_i \\ &\lambda A_iW^h \leq b_i - A_i\overline{x} \\ &\lambda \leq \frac{b_i - A_i\overline{x}}{A_iW^h} \quad \forall i \in N \end{aligned} \tag{18}$$

In particolare, deve essere rispettata la condizione che

$$0 \leq \lambda \leq \frac{b_i - A_i\overline{x}}{A_iW^h} \quad \forall i \in N \tag{19}$$

Rispetto a prima non abbiamo certezze sui prodotti scalari A_iW^h, potrebbe anche verificarsi che

$$A_iW^h < 0 \quad \forall i \in N \tag{20}$$

In questo caso abbiamo che

$$c^T x(\lambda) \to +\infty \text{ per } \lambda \to \infty \tag{21}$$

Invece, se $\exists i : A_i W^h > 0$ allora la semiretta non è contenuta per intera. Vogliamo quindi un valore di λ tale che

$$\lambda \leq \frac{b_i - A_i \overline{x}}{A_i W^h} \forall i \in N \tag{22}$$

Il rapporto appena definito è la "*lunghezza*" massima che la semiretta può avere prima di uscire dal poliedro; di conseguenza, per fare in modo che tutti i vincoli siano rispettati, prendo il rapporto più piccolo.

$$\theta := \min\left\{ \frac{b_i - A_i \overline{x}}{A_i W^h} : i \in N, A_i W^h > 0 \right\} \tag{23}$$

Abbiamo quindi che $x(\lambda)$ è ammissibile se e solo se

$$0 \leq \lambda \leq \theta \tag{24}$$

Osserviamo che il punto ammissibile $x(\theta)$ è la soluzione di base primale associata alla nuova base $B_1 = B \backslash \{h\} \cup \{k\}$; infatti, abbiamo già dimostrato all'inizio che

$$i \in B \backslash \{h\} \implies A_i x(\theta) = b_i \tag{25}$$

Prendendo θ accordato alla definizione abbiamo anche che

$$i = k \implies A_k x(\theta) = A_k(\overline{x} + \theta A_k W^h) = A_k(\overline{x}) + \frac{b_k - A_k \overline{x}}{A_k W^h} A_k W^h = b_k \tag{26}$$

Quindi, l'indice k è il nostro indice **entrante**. Sempre per le regole di **anticiclo di Bland**, se ho più di un indice che ha lo stesso valore di θ, prendo quello di valore minore

$$k := \min\{ i \in N : A_i W^h > 0, \frac{b_i - A_i \overline{x}}{A_i W^h} = \theta \} \tag{27}$$

In conclusione, esiste un teorema fondamentale a riguardo del simplesso primale

Theorem 1.1. Il simplesso primale risolve $(\mathcal{P})$ in un numero finito di iterazioni.

2 Simplesso duale

Descriviamo ora l'algoritmo del simplesso duale per risolvere un problema duale standard:

$$\begin{cases} \min y^T b \\ y^T A = c^T \\ y \geq 0 \end{cases} \tag{28}$$

L'algoritmo è analogo a quello del simplesso primale, con la differenza che ad ogni passo si mantiene ammissibile la soluzione di base duale e si controlla l'ammissibilità di quella primale. Se la soluzione primale è ammissibile allora abbiamo trovato una coppia di soluzioni primale duale complementare e ammissibile per i rispettivi problemi (e quindi ottima). Altrimenti si cambia base, in questo caso il cambio di base è definito in modo che se la nuova soluzione di base duale è diversa da quella vecchia, il valore della funzione obiettivo diminuisce, mentre se la nuova soluzione duale coincide con la vecchia, si evita di ciclare sfruttando opportune politiche di anticiclo. Come nel caso del simplesso primale

Theorem 2.1. Il simplesso duale risolve $(\mathcal{D})$ in un numero finito di iterazioni.

L'algoritmo del simplesso duale può essere schematizzato nel seguente modo

- Trova una base B che genera una soluzione di base duale ammissibile.

- Calcola la soluzione primale di base $A_B\overline{x} = b_B$.
- Se $b_N - A_N\overline{x} \geq 0$ allora ci si ferma, abbiamo trovato la coppia di soluzioni ottime. Altrimenti si calcola l'indice entrante

$$k := \min\{i \in N : b_i - A_i\overline{x} < 0\} \tag{29}$$

  poniamo in seguito $W = -A_B^{-1}$ e indico con W^i la i-esima colonna di W.
- Se $A_kW^i \geq 0, \quad \forall i \in B$ allora ci si ferma, il problema duale ha soluzione ottima $-\infty$ e il problema primale è vuoto. Altrimenti calcoliamo l'indice uscente

$$\begin{aligned} &\theta := \min\{\frac{\overline{y_i}}{-A_kW^i} : i \in N, A_kW^i < 0\} \\ &h := \min\{i \in B : A_kW^i < 0, \overline{y_i} - A_kW^i : i \in N, A_kW^i = \theta\} \end{aligned} \tag{30}$$

3 Problema ausiliario duale

Consideriamo il problema in forma duale standard

$$\begin{cases} \min y^Tb \\ y^TA = c^T \\ y \geq 0 \end{cases} \tag{31}$$

Per poter trovare una base ammissibile del problema duale possiamo procedere andando a costruire il **problema ausiliario duale**

$$\begin{cases} \min \sum_{i=1}^{n} \epsilon_i \\ y^TA + I\epsilon^T = c^T \\ y \geq 0 \\ \epsilon \geq 0 \end{cases} \tag{32}$$

La matrice identità applicata come nel problema implica che venga aggiunta una ϵ ad ogni riga della matrice

$$(A|I)\begin{bmatrix} x \\ \epsilon \end{bmatrix} \tag{33}$$

La base formata dagli indici relativi alle variabili ausiliarie $\epsilon_1, \ldots, \epsilon_n$ con la matrice identità come matrice base, è una base ammissibile per il problema ausiliario. Infatti la corrispondente soluzione di base è $\overline{y} = 0$, $\epsilon = c \geq 0$. A partire da tale base possiamo applicare il simplesso duale per risolvere il problema ausiliario. La soluzione del problema ausiliario stabilisce una base ammissibile per $(\mathcal{D})$.

Theorem 3.1. (Teorema del duale ausiliario)

- Se il valore ottimo $(\mathcal{D}_{aux}) > 0$ allora $(\mathcal{D})$ non ha soluzioni ammissibili.
- Se il valore ottimo $(\mathcal{D}_{aux}) = 0$ allora esiste una base ammissibile per $(\mathcal{D})$.

Quindi, nella sostanza, se si verifica il caso in cui non si trovano soluzioni ammissibili, possiamo dire che il poliedro duale è vuoto. Dal corollario definito in precedenza, supponendo P il poliedro primale e D il poliedro duale, sappiamo che

$$P \neq \emptyset \text{ e } D = \emptyset \implies v(\mathcal{P}) = +\infty \tag{34}$$

Il problema che si pone adesso è quello del vertice di partenza; infatti non possiamo semplicemente calcolare il duale del duale, infatti seguendo questo sistema potremmo andare avanti all'infinito. Quello che però possiamo fare è porre tutte le x a 0 e ottenere l'identità

$$I\epsilon = b \tag{35}$$

che può essere espresso come

$$\begin{bmatrix} \epsilon_1 \\ \vdots \\ \epsilon_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix} \tag{36}$$

quindi, la soluzione di base ammissibile di partenza, per il **problema ausiliario duale**, è quella in cui tutte le $x_i = 0$ per ogni $x_j = 1, \ldots, n$ e dove $\epsilon_i = b_i$ per ogni $i = 1, \ldots, m$

$$\overline{x} = (x_1 = 0, \ldots, x_n = 0, b_1, \ldots, b_m) \tag{37}$$

4 Il caso degenere

Prendiamo un problema di PL e scriviamone il duale

$$\begin{cases} \max x_1 + 2x_2 \\ x_1 \leq 1 \\ x_2 \leq 1 \\ x_1 + x_2 \leq 1 \\ -x_1 \leq 0 \\ -x_2 \leq 0 \end{cases} \qquad \begin{cases} \min y_1 + y_2 + y_3 \\ y_1 + y_3 - y_4 = 1 \\ y_2 + y_3 - y_5 = 2 \\ y \geq 0 \ \ \forall i = 1, \ldots, m \end{cases} \tag{38}$$

quindi, abbiamo un vertice degenere, in quanto può essere ottenuto con 3 differenti basi. Partiamo con la base $B = \{1, 3\}$ e applichiamo il simplesso

$$\overline{x} = (1, 1) \quad \overline{y} = (-1, 0, 2, 0, 0) \tag{39}$$

La base non è quindi ammissibile; calcolando sia indice uscente che indice entrante la nuova base diventa $B = \{2, 3\}$. Andiamo a verificare nuovamente se la base è ottima

$$\overline{x} = (1, 1) \quad \overline{y} = (0, 1, 1, 0, 0) \tag{40}$$

Questa è la soluzione ottima, in quanto le due soluzioni sono ambo ammissibili e, per teorema, ottime. Notiamo però una cosa, la soluzione ammissibile del problema primale è uguale in entrambi i casi; quindi, nonostante il cambio di base, il vertice è rimasto uguale; questa occorrenza si verifica nel caso degenere.

Il motivo per cui si verifica questa problematica è dovuto dal fatto che il simplesso controlla le basi e utilizza la soluzione duale come condizione di ottimalità; per l'algoritmo la prima base del caso degenere non è ottima nonostante essa produca una soluzione ottima del primale in quanto produce una soluzione duale non ammissibile. Quindi il simplesso può anche iterare rimanendo nello stesso vertice e ribattezzando quello che soddisfa la condizione come **base ottima**.

Regole anticiclo di Bland

Le regole di **anticiclo di *Bland*** sono un inseme di regole, usate nell'algoritmo del simplesso, per fare in modo che esso possa ciclare in modo corretto evitando di entrare in dei loop. Esistono due teoremi fondamentali

Theorem 4.1. L'algoritmo del simplesso primale con le regole di anticiclo di Bland non cicla.

Theorem 4.2. L'algoritmo del simplesso duale con le regole di anticiclo di Bland non cicla.

Queste regole possono essere riassunte nel modo seguente.

- Regole di **anticiclo** per il simplesso primale:
  - **Indice uscente**: Si sceglie il minimo degli indici negativi della soluzione del duale.
  - **Indice entrante**: A parità dei rapporti scelgo il minimo degli indici.

- Regole di **anticiclo** per il simplesso duale:
  - **Indice entrante**: Si sceglie il primo vincolo violato dalla soluzione del primale associato.
  - **Indice uscente**: si sceglie il minimo degli indici a parità di rapporti.

Parte II

Programmazione lineare intera

Capitolo 4

Teoria della PLI

L'idea alla base della risoluzione di un problema di **programmazione lineare intera** è quella di definire un **gap**, quindi un intervallo all'interno del quale sarà presente la soluzione ottima. Questo intervallo è composto da una

- **Valutazione inferiore**: $v_I(\mathcal{P})$.
- **Valutazione superiore**: $v_S(\mathcal{P})$.
- **Valore ottimo**: $v(\mathcal{P})$

Questi vincoli devono rispettare la proprietà

$$v_I(\mathcal{P}) \leq v(\mathcal{P}) \leq v_S(\mathcal{P}) \tag{1}$$

Passiamo brevemente a descrivere come possiamo calcolare queste valutazioni. Il primo passo è quello di identificare il tipo di problema, in particolare

Lemma 0.1. Dato un problema di **massimo** (**minimo**), la valutazione superiore v_S è definita come l'**arrotondamento per difetto** del valore della soluzione del rilassato continuo (una **soluzione ammissibile** del rilassato continuo); la valutazione inferiore è definita come una soluzione ammissibile (**arrotondamento per difetto** del valore della soluzione del rilassato continuo) del rilassato continuo.

Una possibile esempio di **valutazione inferiore** per un problema di **massimo** (o **valutazione superiore** per un problema di **minimo**) potrebbe essere quella di prendere la soluzione ottima del rilassato continuo, arrotondarla per difetto e calcolarne il valore.

1 Relazioni tra PL e PLI

Un problema della **PLI** consiste nel trovare il minimo di una funzione lineare su una regione definita da vincoli lineari e da **vincoli di interezza sulle variabili** e supponendo che i dati del problema siano numeri interi. Possiamo quindi supporre che tale problema possa essere strutturato come

$$\begin{cases} \max c^T x \\ Ax \leq b \\ x \in \mathbb{Z}^n \end{cases} \tag{2}$$

dove A è una matrice $m \times n$ a componenti intere, $b \in \mathbb{Z}^m$ e $c \in \mathbb{Z}^n$. I problemi in cui le variabili sono **binarie** sono chiamati **problemi di *combinatorica***. Nella PL il poliedro definito dai vincoli del problema coincideva anche con la regione ammissibile; nella PLI invece esiste una relazione diversa tra poliedro del problema e regione ammissibile

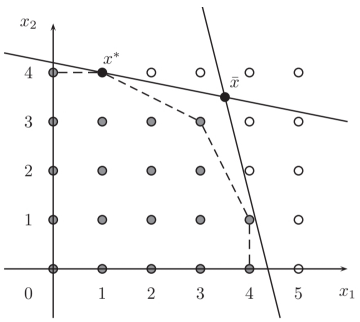
$$P = \{x \in \mathbb{R}^n : Ax \leq b\} \implies \Omega = \{P \cap \mathbb{Z}^n\} \tag{3}$$

Quindi, considerando la regione ammissibile Ω come quella appena definita, il problema

$$\begin{cases} \max c^T b \\ Ax \leq b \end{cases} \tag{4}$$

viene definito **rilassamento continuo** (*del problema principale*). Una conclusione importante a cui possiamo arrivare è che se $\overline{x}$ è soluzione ottima del rilassato continuo, allora né la soluzione ottenuta arrotondando $\overline{x}$, né la soluzione ammissibile più vicina sono, in generale, soluzioni ottime del problema di PLI.

Visualizziamo graficamente il motivo



Notiamo facilmente che la soluzione ottima del problema di PLI è data da $(1, 4)$, mentre la soluzione ottima del **rilassamento continuo** è $(7/2, 7/2)$. Né arrotondando e né tanto meno andando a cercare la soluzione più vicina ammissibile per il problema di PLI (rispetto alla soluzione del rilassato continuo) si riesce ad ottenere la soluzione ottima del problema di PLI. Quindi, per risolvere un problema di PLI non è sufficiente trovare la soluzione ottima del rilassamento continuo. Analizziamo allora quale ruolo la soluzione del rilassato continuo potrebbe avere sulla soluzione ottima del problema PLI.

Si osservi però che se la soluzione $v(\mathcal{P})$ del problema di PL è anche ammissibile per il problema di PLI, allora abbiamo già trovato la soluzione ottima

Nella grafico appena descritto che il poliedro delimitato dagli assi e dai vincoli lineari tratteggiati rappresenta l'involucro convesso delle soluzioni ammissibili

$$\text{conv}(\Omega) = \left\{ x \in \mathbb{R}^n : \exists x_1, \ldots, x_m \in \Omega, x = \sum_{i=1}^{m} \lambda_i x_i, \lambda_1, \ldots, \lambda_n \mid \sum_{i=1}^{m} \lambda_i = 1 \right\} \tag{5}$$

Poiché si ha che $\Omega \subseteq \text{conv}(\Omega) \subseteq P$, vale anche la seguente catena di disuguaglianze

$$\max_{x \in P} c^T x \geq \max_{x \in \text{conv}(\Omega)} c^T x \geq \max_{x \in \Omega} c^T x \tag{6}$$

Passiamo ora a definire un importante lemma sulla PLI

Lemma 1.1. Esiste un insieme finito di punti $\{q^l\}_{l \in L} \in \Omega$ ed un insieme finito di *direzioni di recessione* $\{r^j\}_{j \in J} \in P$ tali che

$$\Omega = \left\{ x \in \mathbb{R}^n_+ : x = \sum_{l \in L} \alpha_l q_l + \sum_{j \in J} \beta_j r_j, \sum_{l \in L} \alpha_l = 1, \alpha \in \mathbb{Z}^{|L|}, \beta \in \mathbb{Z}^{|J|} \right\} \tag{7}$$

Questo teorema è utilissimo, infatti, dato un insieme finito di punti della regione ammissibile, che potrebbe pure essere un singolo punto, e date le direzioni in cui si estende all'infinito il poliedro, io posso ricostruire facilmente e per intero la regione ammissibile del nostro problema di PLI.

Il secondo passo è quello di dimostrare che $\text{conv}(\Omega)$ è un poliedro a **coefficienti razionali**, quindi esiste un insieme finito di vettori $q_1, \ldots, q_{|L|}$ e $r_1, \ldots, r_{|J|}$ a componenti razionali, tali che

$$\text{conv}(\Omega) = \text{conv}(\{q^l\}_{l \in L}) + \text{cono}(\{r^j\}_{j \in J}) \tag{8}$$

Attraverso un piccolo esempio andiamo a dimostrare che se le matrici A e b non sono a coefficienti razioali, allora conv(Ω) non può essere un poliedro. Consideriamo il poliedro

$$P = \{x \in \mathbb{R}^2 : 0 \leq x_2 \leq \sqrt{2}x_1\} \tag{9}$$

Esiste un teorema di equivalenza tra PL e PLI, che è opportuno enunciare. Si considerino i due problemi

$$v = \max_{x \in S} c^T x \qquad v^* = \max_{x \in \text{conv}(S)} c^T x \tag{10}$$

Il teorema di equivalenza ci assicura che

Theorem 1.2. (Teorema di equivalenza della PLI) I valori di v e v^* coincidono. Inoltre, se v^* è finito, allora esiste $x^* \in S$ tale per cui

$$c^T x^* = v = v^* \tag{11}$$

Questo teorema sembra, almeno all'apparenza, un teorema che ci da un risultato fortissimo. A patto di essere in grado di costruire conv(s) siamo in grado di ricondurre un qualsiasi problema di PLI a un problema di PL, ma, come si suol dire, *non è tutto oro quel che luccica*. Il problema che nasce dall'utilizzare questo teorema è proprio la costruzione di conv(s), quest'operazione ha infatti costo esponenziale.

Esiste però una classe di problemi di PLI la cui soluzione ottima coincide con la soluzione ottima del rilassamento continuo. Questa categoria di problemi è quella in cui la matrice dei vincoli è **totalmente unimodulare**.

Definition 1.1. (Matrice unimodulare) Data una matrice A, diciamo che **A è totalmente unimodulare** quando il determinante di ogni sottomatrice quadrata è 1, -1 o 0.

Supponendo quindi che il nostro problema abbia una matrice totalmente unimodulare, possiamo dire che

Theorem 1.3. Se A e b sono a componenti **intere** e A è una **matrice unimodulare**, allora le soluzioni di base del poliedro

$$P = \{x \in \mathbb{R}^n : Ax \leq b\} \tag{12}$$

sono a componenti intere.

Rientra in questa categoria di problemi il problema dell'assegnamento di costo minimo, o anche, supponendo che la domanda e la capacità siano intere per ogni nodo, anche il problema del trasporto di costo minimo.

Capitolo 5

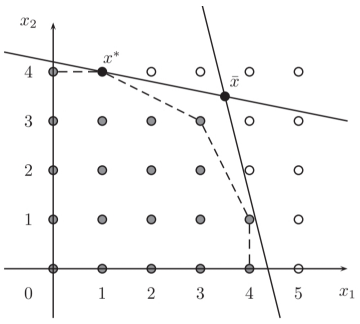
Piani di taglio

Il primo metodo di *riduzione del gap* che andiamo a studiare è il metodo dei **piani di taglio**. In generale, ogni metodo di *riduzione del gap* dei problemi di PLI, parte da un assunto fondamentale: dato un problema $(\mathcal{P})$, la sua soluzione ottima è compresa tra la **valutazione inferiore** e la **valutazione superiore** del problema stesso

$$V_I(\mathcal{P}) \leq V(\mathcal{P}) \leq V_S(\mathcal{P}) \tag{1}$$

Nello specifico, l'idea alla base del metodo dei piani di taglio è quella di prendere delle porzioni della regione ammissibile e tagliarle introducendo nuovi vincoli. In particolare, l'idea è quella di calcolare iterativamente la soluzione del rilassamento continuo del problema $\overline{x}$, se la soluzione è tutta a componenti intere abbiamo trovato l'ottimo del nostro problema; altrimenti si crea un piano di taglio che tagli la porzione di regione dove è contenuta $\overline{x}$. I piani di taglio permettono quindi di tagliare porzioni della regione ammissibile, escludendo le soluzioni **non ammissibili** trovate tramite il rilassamento continuo, senza andare a inficiare con la ricerca dell'ottimo del problema.

Assumiamo di avere una regione ammissibile così strutturata



Supponiamo di aver appena calcolato la soluzione ottima $\overline{x}$ del nostro rilassato continuo, andando a osservarla ci si accorge che ha almeno una parte frazionaria e, quindi, che non può essere l'ottima del nostro problema. Procediamo, quindi, a calcolare il piano di taglio; supponendo che il poliedro sia definito nella forma

$$Ax \leq b \tag{2}$$

si aggiunge un vincolo (**piano di taglio**) $\omega^T x \leq \omega_0$, facendo diventare il nostro poliedro ammissibile

$$\begin{cases} Ax \leq b \\ \omega^T x \leq \omega_0 \end{cases} \tag{3}$$

si procede poi ricalcolando la soluzione del rilassamento continuo. Però, non tutti i piani di taglio sono validi, infatti, affinché questo vincolo sia ritenuto valido deve valere una condizione

Lemma 0.1. La disuguaglianza $\omega^T x \leq \omega_0$ è detta disuguaglianza valida (DV) per l'insieme Ω se

$$\omega^T x \leq \omega_0 \quad \forall x \in \Omega \tag{4}$$

Affinché però questo vincolo sia definibile **piano di taglio** occorre che sia valida anche un'altra condizione

Lemma 0.2. Sia $\overline{x}$ l'ottimo del rilassamento continuo. Una disuguaglianza valida $\omega^T x \leq \omega_0$ per Ω tale che $\omega^T \overline{x} > \omega_0$ si dice piano di taglio.

In conclusione, come abbiamo già detto all'inizio del paragrafo

L'iterazione si ferma quando la nuova soluzione ottima del rilassato continuo, calcolata introducendo i vincoli di taglio, ha componenti intere.

1 Piani di taglio di Gomory

Un insieme classico di piani di taglio, che affronteremo durante il corso, è quello dei **piani di taglio di Gomory**. Supponiamo che il nostro problema di PLI sia nella forma

$$\begin{cases} \max c^T x \\ Ax = b \\ x \geq 0 \\ x \in \mathbb{Z}^n \end{cases} \tag{5}$$

Supponiamo inoltre che B sia una base ottima del rilassamento continuo del nostro problema (che possiamo calcolare tranquillamente con Matlab). Poniamo

$$A = \begin{bmatrix} A_B & A_N \end{bmatrix} \quad x = \begin{bmatrix} x_B \\ x_N \end{bmatrix} \tag{6}$$

a questo punto poniamo

$$\overline{b} = \overline{x}_B \quad A = A_B^{-1} A_N \tag{7}$$

e applichiamo il seguente risultato

Theorem 1.1. Se $\overline{b}_r \notin \mathbb{Z}$, allora la disuguaglianza

$$\sum_{j \in N} \{\{\overline{a}_{rj}\} x_j \geq \{\overline{b}_r\}\} \tag{8}$$

è un piano di taglio per il problema di PLI.

Richiamiamo un concetto fondamentale prima di enunciare l'esempio

Definition 1.1. Parte frazionaria La **mantissa**, o *parte frazionaria* di un numero è definita come la differenza tra il numero e la sua parte intera

$$\{\overline{x}\} = \overline{x} - [\overline{x}] \tag{9}$$

Esempio sui piani di taglio

Si consideri il seguente problema di PLI

$$\begin{cases} \max 5x_1 + 14x_2 \\ 18x_1 + 8x_2 \leq 55 \\ 14x_1 + 18x_2 \leq 61 \\ x_i \in \mathbb{Z}_+ \end{cases} \tag{10}$$

Calcolare i possibili piani di taglio di Gomory

Risoluzione

Il primo passo è portare il problema nella forma (6); lo facciamo introducendo le variabili di scarto x_3 e x_4:

$$\begin{cases} \max x_1 + 3x_3 \\ x_1 + 5x_2 + x_3 = 21 \\ 8x_1 + 2x_2 + x_4 = 35 \\ x_i \geq 0 \\ x_i \in \mathbb{Z}^4 \end{cases} \tag{11}$$

in questo caso abbiamo

$$A = \begin{bmatrix} 1 & 5 & 1 & 0 \\ 8 & 2 & 0 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 21 \\ 33 \end{bmatrix}, \quad c^T = \begin{bmatrix} 1 & 3 & 0 & 0 \end{bmatrix} \tag{12}$$

Andando a risolvere il rilassamento continuo otteniamo che la base ottima è $B = \{1, 2\}$, quindi

$$A_B = \begin{bmatrix} 1 & 5 \\ 8 & 2 \end{bmatrix}, \quad A_B^{-1} = \begin{bmatrix} -\frac{1}{19} & \frac{5}{38} \\ \frac{4}{19} & -\frac{1}{38} \end{bmatrix} \tag{13}$$

ed, inoltre

$$\overline{x}_B = \begin{bmatrix} \frac{7}{2} \\ \frac{7}{2} \end{bmatrix} \quad x_N = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \overline{A} = \begin{bmatrix} -\frac{1}{19} & \frac{5}{38} \\ \frac{4}{19} & -\frac{1}{38} \end{bmatrix} \tag{14}$$

il vettore $b = \overline{x}^T$ ha entrambe le componenti non intere, quindi ci sono due possibili tagli di Gomory

- per $r = 1$ il taglio è

$$\left\{ -\frac{1}{19} \right\} x_3 + \left\{ \frac{5}{38} \right\} x_4 \geq \left\{ \frac{7}{2} \right\} \tag{15}$$

  dal quale ricavo

$$36x_3 + 5x_4 \geq 19 \tag{16}$$

  che nelle variabili x_1, x_2 diventa

$$2x_1 + 5x_2 \leq 24 \tag{17}$$

- per $r = 2$ il taglio è

$$\left\{ \frac{4}{19} \right\} x_3 + \left\{ -\frac{1}{38} \right\} x_4 \geq \left\{ \frac{7}{2} \right\} \tag{18}$$

  che ricavando le variabili x_1, x_2 equivale a

$$8x_1 + 3x_2 \leq 38 \tag{19}$$

Capitolo 6

Metodo del Branch And Bound

Il metodo del *branch and bound* si basa su un principio fondamentale: quando abbiamo una stima inferiore, possiamo esplorare l'albero delle soluzioni possibili alla ricerca di una soluzione ammissibile (ossia con variabili intere) migliore. Se durante questa esplorazione troviamo una soluzione che supera la precedente, essa diventa la nostra nuova soluzione ammissibile. Una volta completata l'analisi dell'intero albero, la soluzione ammissibile identificata sarà considerata anche ottima.

Supponiamo che il problema di PLI da risolvere sia

$$\begin{cases} \max c^T x \\ Ax \leq b \\ x \in \mathbb{Z}^n \end{cases} \tag{1}$$

e che abbia una regione ammissibile limitata. Essendo **finito** il numero delle soluzioni ammissibili, potrebbe essere utile calcolare il valore della funzione obiettivo in tali punti e andare a trovare l'ottimo confrontando i le soluzioni e trovando quella di valore massimo.

Per esplorare tutte le soluzioni ammissibili è costruire il cosidetto **albero di enumerazione totale**. Esso ha per radice il problema $(\mathcal{P})$; facendo una partizione della regione ammissibile Ω di $(\mathbb{P})$ in due o più sottoinsiemi $\Omega_1, \ldots, \Omega_p$ si ottengono dei figli del nodo $(\mathcal{P})$, che corrispondono a sotto-problemi di quest'ultimo aventi regioni ammissibili $\Omega_1, \ldots, \Omega_p$; il resto dell'albero viene generato in egual modo, procedendo fintanto che non si arriva a sotto-problemi la cui regione ammissibile è vuota o con un singolo elemento.

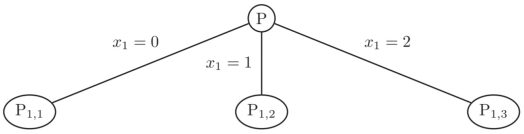
Prendiamo come esempio il problema

$$\begin{cases} \max 5x_1 + 6x_2 \\ 3x_1 + 4x_2 \leq 7 \\ x \geq 0 \\ x \in \mathbb{Z}^2 \end{cases} \tag{2}$$

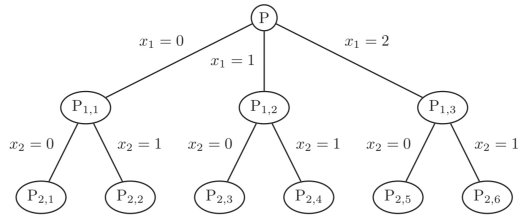
I vincoli del problema impongono che la variabile x_1 possa assumere solamente valori $\overline{x} \in \{0, 1, 2\}$. Partizionando la regione ammissibile abbiamo che

$$\Omega = (\Omega \cap \{x_1 = 0\}) \cup (\Omega \cap \{x_1 = 1\}) \cup (\Omega \cap \{x_1 = 2\}) \tag{3}$$

che sull'albero di enumerazione totale, potremmo far corrispondere a 3 figli.



Analogamente alla prima, x_2 può assumere solo valori $\overline{x}_2 \in \{0, 1\}$; quindi, ciascuno dei figli della radice, avrà a sua volta altri due figli. L'albero di enumerazione finale risulta quindi essere



Le foglie $P_{2,1}, \ldots, P_{2,5}$ sono delle **soluzioni ammissibili**, mentre invece la foglia $P_{2,6}$ corrisponde al vettore $\overline{x} = (2, 1)$ che **non è ammissibile**. I valori della funzione obiettivo nei rispettivi punti sono $0, 6, 5, 11, 10$, pertanto la soluzione ammissibile è il corrispondenza di $P_{2,4}$.

Questa strategia però ha un grosso problema, **il numero delle soluzioni ammissibili cresce esponenzialmente rispetto al numero delle variabili del problema**; questo fatto rende impossibile l'applicazione di questo algoritmo. Una strategia che permette di esplorare l'albero di enumerazione totale senza dover necessariamente enumerare tutte le foglie è il **Branch and Bound**.

Indichiamo con $v(\mathcal{P})$ il valore ottimo del problema e indichiamo con $v_I(\mathcal{P})$ e $v_S(\mathcal{P})$, rispettivamente, una valutazione inferiore e una superiore del valore ottimo del problema P

$$v_I(\mathcal{P}) \leq v(\mathcal{P}) \leq v_S(\mathcal{P}) \tag{4}$$

Indichiamo infine con $P_{i,j}$ il j-esimo problema a livello i dell'albero di enumerazione. Le tecniche più comuni di **bound** che forniscono una valutazione superiore dell'ottimo di un generico sotto-problema $P_{i,j}$ sono

- Eliminazione di uno o più vincoli.
- Somma di vincoli.
- Rilassamento continuo: cambiare $x \in \mathbb{Z}^n$ con $x \in \mathbb{R}^n$.

Alcune tecniche per la valutazione inferiore di $(\mathcal{P})$:

- euristiche per il calcolo di una soluzione ammissibile.
- risoluzione di un problema ottenuto aggiungendo uno o più vincoli a $(\mathcal{P})$.

D'altra parte, per le tecniche di **Branch** saranno descritte le regole più comuni

Theorem 0.1. Sia x una soluzione ammissibile di $(\mathcal{P})$ tale per cui $V_I(\mathcal{P}) = c^T x$

- Se $V_S(\mathcal{P}_{i,j}) \leq V_I(\mathcal{P})$ oppure la regione ammissibile è vuota, allora si può effetturare una visita implicita del nodo $P_{i,j}$, cioè nel sottoalbero di radice $P_{i,j}$, non esiste una soluzione ammissibile di valore superiore a x.
- Se $V_S(\mathcal{P}_{i,j}) > V_I(\mathcal{P})$ e l'ottimo $\overline{x}$ del rilassamento continuo di $P_{i,j}$ è ammissibile per $\mathcal{P}$, allora $\overline{x}$ è una soluzione migliore di x, quindi si aggiorna $\overline{x}$ e si effettua una visita implicita di $P_{i,j}$

Quello che facciamo con il Branch and bound è muovermi tra i vertici del poliedro introducendo dei vincoli; se il problema P è di **massimo** (**minimo**) e introducendo il vincolo troviamo una valutazione **superiore** (**inferiore**) del sotto-problema $P_{i,j}$ migliore della valutazione **inferiore** (**superiore**)

Capitolo 7

Il problema dello *Zaino*

In generale, il problema dello zaino binario può essere formulato nel seguente modo: dato un contenitore di volume b e n oggetti aventi ciascuno un valore c_i ed un volume a_i, dobbiamo scegliere quali oggetti inserire nel contenitore in modo da massimizzare il valore totale degli oggetti caricati. Se ad ogni oggetti associamo una variabile binaria x_i tale che

$$x_i = \begin{cases} 1 \text{ se l'oggetto i viene caricato} \\ 0 \text{ altrimenti} \end{cases} \tag{1}$$

allora tale problema può essere formulato nel modo seguente

$$\begin{cases} \max \sum_{i=1}^{n} v_i x_i \\ \sum_{i=1}^{n} a_i x_i \leq b \\ x_i \in \{0,1\} \end{cases} \tag{2}$$

Questa formulazione però non considera una possibilità, quella che uno stesso oggetto possa essere inserito in più unità. In questo caso il modello matematico corrispondente non avrà più il vincolo $x_i \in \{0,1\}$, che verrà invece sostituito dal vincolo $x \in \mathbb{Z}_+$.

$$\begin{cases} \max \sum_{i=1}^{n} v_i x_i \\ \sum_{i=1}^{n} a_i x_i \leq b \\ x \in \mathbb{Z}_+ \end{cases} \tag{3}$$

Per trovare una possibile soluzione ottima andiamo a definire una nuova grandezza, il ***rendimento***, definita come

Definition 0.1. Si definisce come ***rendimento*** il rapporto tra il **valore** dell'oggetto e il suo **ingombro**

$$r_i = \frac{v_i}{a_i} \tag{4}$$

Definiamo r_k come il massimo tra i rendimenti

$$r_k = \max_{i=1}^{n} \; r_i \tag{5}$$

Andiamo ora ad analizzare distintamente i due casi

1 Problema a variabili intere

Sia dato il problema di caricamento

$$\begin{cases} \max \sum_{i=1}^{n} v_i x_i \\ \sum_{i=1}^{n} a_i x_i \leq b \\ x_i \in \mathbb{Z}_+ \quad \forall i = 1, \ldots, n \end{cases} \tag{6}$$

Il primo passo è quello di trovare una valutazione superiore, essendo il nostro un problema di massimo, dalle definizioni date all'inizio della sezione, dobbiamo trovare una soluzione ottima del rilassato continuo. Il problema da risolvere diventa quindi

$$\begin{cases} \max \sum_{i=1}^{n} v_i x_i \\ \sum_{i=1}^{n} a_i x_i \leq b \\ x_i \geq 0 \quad \forall i = 1, \ldots, n \end{cases} \tag{7}$$

Andiamo a definire il rispettivo problema duale

$$\begin{cases} \min y^T b \\ y \geq \dfrac{v_1}{a_1} \\ \vdots \\ y \geq \dfrac{v_n}{a_n} \end{cases} \tag{8}$$

Andando a risolvere il problema duale, supponendo che $r_k = v_1/a_1$, si trova che la soluzione è

$$\overline{x} = \left(\frac{b}{a_1}, 0, \ldots, 0 \right) \tag{9}$$

il che equivale a dire che lo zaino viene saturato solo con l'oggetto di rendimento massimo; abbiamo quindi la nostra valutazione superiore

$$v_S = \left\lfloor \frac{b}{a_1} \right\rfloor \tag{10}$$

Il simbolo utilizzato indica **l'intero inferiore**, quindi il valore ottenuto arrotondando per difetto il numero.

Il secondo passo è quello di trovare una valutazione inferiore del problema. Per farlo applichiamo un algoritmo greedy: procediamo in ordine di **rendimento decrescente** e inseriamo gli oggetti nello zaino nella massima quantità possibile, controllando che venga rispettato il vincolo di capacità. L'algoritmo può essere sintetizzato come

- Per ogni oggetto da inserire, indicizzato con l'indice **k**, andiamo a controllare se lo spazio occupato è maggiore dello spazio disponibile ($a_k \geq b$)
  - Se la condizione non è verificata, andiamo a calcolare la quantità massima che possiamo inserire:

    $$x_k = \left\lfloor \frac{b}{a_k} \right\rfloor \tag{11}$$

    e andiamo ad aggiornare lo spazio disponibile

    $$b = b - a_k x_k \tag{12}$$

– Se la condizione è verificata, si pone la quantità da inserire uguale a 0.

$$x_k = 0 \tag{13}$$

- Si continua l'iterazione per gli altri elementi.

La valutazione inferiore dell'algoritmo è definita come la somma dei prodotti tra il **guadagno** e il **quantitativo** degli oggetti inseriti; definiamo l'insieme S come l'insieme degli oggetti nello zaino

$$v_I = \sum_{i \in S} v_i x_i \tag{14}$$

Si noti bene che questo algoritmo fornisce una valutazione inferiore, quindi una soluzione ammissibile, ma non necessariamente una **soluzione ottima**. Adesso che abbiamo le nostre valutazioni possiamo passare all'applicazione del *Branch-and-Bound*.

L'applicazione del *Branch-and-Bound* in questo caso non permette un partizionamento **binario** di $x_i = 0$ o $x_i = 1$, in quanto il vincolo che stiamo utilizzando non è più $x_i \in \{0, 1\}$, ma $x_i \in \mathbb{Z}^n$. Eistono due diverse alternative per l'applicazione dell'algoritmo

- Utilizzare una partizione *ennaria* che tenga conto di tutto l'insieme dei valori che la variabile può assumere, andando a creare un ramo diverso per ogni possibile valore. Questo sistema presenta però un piccolo problema; la variabile deve essere **discreta**, quindi, deve esse noto l'*insieme dei valori che può assumere.*

- L'altro metodo consiste nell'utilizzare una partizione **binaria**. In particolare, andiamo a calcolare una soluzione ottima del rilassamento continuo $\overline{x}$ e si considera l'elemento $\overline{x}_i \in \overline{x}$ che assume valore frazionario. L'obiettivo della partizione è impedire che la variabile x_i assuma nuovamente il valore frazionario $\overline{x}_i$ nei due sottoproblemi figli. A tale fine andiamo a imporre la condizione $\lfloor x_i \rfloor$ su un ramo e la condizione $\lceil x_i \rceil$.

Proviamo a fare un esempio con un problema reale. Sia dato il seguente problema

$$\begin{cases} \max 4x_1 + 20x_2 + 27x_3 + 26x_4 \\ 4x_1 + 19x_2 + 16x_3 + 14x_4 \leq 32 \\ x_i \in \mathbb{Z}_+ \quad \forall i = 1, \ldots, 4 \end{cases} \tag{15}$$

Dal vincolo sulle capacità dello zaino possiamo ricavare le seguenti disuguaglianze

$$\begin{aligned} 4x_1 &\leq 32 \\ 19x_2 &\leq 32 \\ 16x_3 &\leq 32 \\ 14x_4 &\leq 32 \end{aligned} \tag{16}$$

dai quali, andando a fare gli opportuni arrotondamenti all'intero inferiore, otteniamo che

$$x_1 \leq 8 \quad x_2 \leq 1 \quad x_3 \leq 2 \quad x_4 \leq 2 \tag{17}$$

Andiamo anche a calcolare le due valutazioni:

- la valutazione inferiore ottenuta attraverso l'algoritmo **greedy** è $v_I = 56$, corrispondente alla soluzione $(1, 0, 0, 2)$.

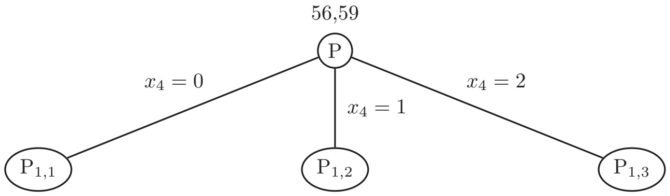
- La valutazione superiore è ottenuta attraverso la soluzione ottima del rilassamento continuo; quindi $v_S = 59$, corrispondente alla soluzione $(0, 0, 0, 16/7)$.

Inoltre, i rendimenti sono tutti in ordine crescente

$$\frac{4}{4} < \frac{20}{19} < \frac{27}{16} < \frac{26}{14} \tag{18}$$

Passiamo ora all'applicazione del metodo del *Branch-and-Bound* sfruttando quello che abbiamo detto fino ad adesso

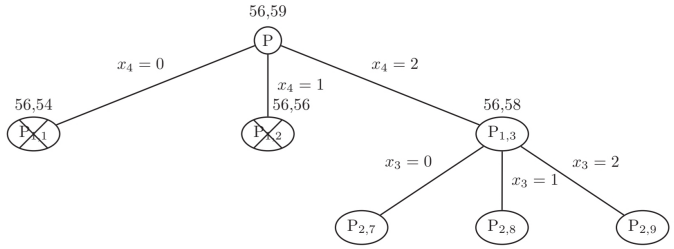
Visto che l'ultimo elemento ha componente frazionaria, possiamo iniziare a esplorare l'albero partendo da esso



Se andiamo ad aggiungere il vincolo al nostro problema e calcoliamo la soluzione del rilassamento continuo abbiamo che

- La soluzione ottima di $P_{1,1}$ è $(0, 0, 2, 0)$, quindi $v_S(P_{1,1}) = 54 \leq V_I(P)$ e di conseguenza possiamo chiudere il nodo 1.
- La soluzione ottima di $P_{1,2}$ è $(0, 0, 9/8, 1)$, quindi $v_S(P_{1,1}) = 56 = V_I(P)$ e di conseguenza possiamo chiudere il nodo 2.
- soluzione ottima di $P_{1,3}$ è $(0, 0, 1/4, 2)$, quindi $v_S(P_{1,1}) = 58 > V_I(P)$ e di conseguenza il nodo 3 rimane aperto.

Quindi, dal nodo $P_{1,3}$ andiamo a instanziare la variabile x_3



Aggiungiamo l'ulteriore vincolo e andiamo a calcolare nuovamente la soluzione del rilassamento continuo

- L'ottimo del rilassamento di $P_{2,7}$ è $(0, 4/19, 0, 2)$, per cui $v_S(P_{2,7}) = 56 = v_I$ e chiudiamo il nodo $P_{2,7}$
- Per gli altri due vincoli troviamo che non esistono soluzioni ammissibili, quindi possiamo chiudere anche i due rami.

Deduciamo quindi che la soluzione ottima è $(1, 0, 0, 2)$ con valore 56.

2 Problema a variabili binarie

Sia dato il seguente problema di caricamento

$$\begin{cases} \max \sum_{i=1}^{n} v_i x_i \\ \sum_{i=1}^{n} a_i x_i \leq b \\ x_i \in \{0, 1\} \end{cases} \tag{19}$$

Il primo passo è quello di trovare una valutazione superiore, essendo il nostro un problema di massimo, dalle definizioni date all'inizio della sezione, dobbiamo trovare una soluzione ottima del

rilassato continuo. Il problema da risolvere diventa quindi

$$\begin{cases} \max \sum_{i=1}^{n} v_i x_i \\ \sum_{i=1}^{n} a_i x_i \leq b \\ x_i \geq 0 \quad \forall i = 1, \ldots, n \end{cases} \tag{20}$$

Andiamo a definire il rispettivo problema duale

$$\begin{cases} \min y^T b \\ y \geq \dfrac{v_1}{a_1} \\ \vdots \\ y \geq \dfrac{v_n}{a_n} \end{cases} \tag{21}$$

Andando a risolvere il problema duale, supponendo che $r_k = v_1/a_1$, si trova che la soluzione è

$$\overline{x} = \left(\frac{b}{a_1}, 0, \ldots, 0 \right) \tag{22}$$

il che equivale a dire che lo zaino viene saturato solo con l'oggetto di rendimento massimo; abbiamo quindi la nostra valutazione superiore

$$v_S = \left\lfloor \frac{b}{a_1} \right\rfloor \tag{23}$$

Il simbolo utilizzato indica **l'intero inferiore**, quindi il valore ottenuto arrotondando per difetto il numero.

Il secondo passo è quello di trovare una valutazione inferiore del problema. Per farlo applichiamo un algoritmo greedy: procediamo in ordine di **rendimento decrescente** e inseriamo gli oggetti nello zaino progressivamente, controllando che venga rispettato il vincolo di capacità. L'algoritmo può essere sintetizzato come

- Per ogni oggetto da inserire, indicizzato con l'indice **k**, andiamo a controllare se lo spazio occupato è maggiore dello spazio disponibile ($a_k \geq b$)
  - Se la condizione non è verificata, inseriamo l'oggetto all'interno dello zaino $x_k = 1$.
  - Se la condizione è verificata non inseriamo l'oggetto nello zaino $x_k = 0$.
- Si continua l'iterazione per gli altri elementi.

La valutazione inferiore dell'algoritmo è definita come la somma dei dei valori degli oggetti inseriti; definiamo l'insieme S come l'insieme degli oggetti nello zaino

$$v_I = \sum_{i \in S} v_i x_i \tag{24}$$

Si noti bene che questo algoritmo fornisce una valutazione inferiore, quindi una soluzione ammissibile, ma non necessariamente una **soluzione ottima**. Adesso che abbiamo le nostre valutazioni possiamo passare all'applicazione del *Branch-and-Bound*.

Esempio di applicazione del branch and bound sullo zaino binario

Per capire meglio l'applicazione dell'algoritmo prendiamo in esame il seguente problema

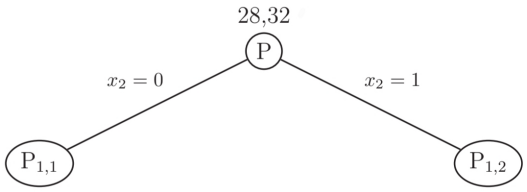
$$\begin{cases} \max 10x_1 + 13x_2 + 18x_3 + 24x_4 \\ 2x_1 + 3x_2 + 4x_3 + 6x_4 \leq 7 \\ x_i \in \{0,1\} \ \ \forall i = 1,\dots,4 \end{cases} \tag{25}$$

Per trovare le rispettive valutazioni disponiamo le variabili in ordine decrescente di rendimento Applicando il nostro algoritmo greedy otteniamo la soluzione ammissibile $(1,0,1,0)$ e quindi

Variabili	1	3	2	4
Rendimenti	5	4.5	4.3	4

$v_I(P) = 28$.

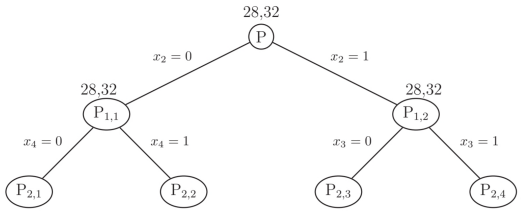
La soluzione del rilassamento continuio è invece $(1, 1/3, 1, 0)$, quindi $v_S(P) = 32$. Vista la componente frazionaria della variabile x_2 iniziamo a lavorare partendo da essa Dopo di che andiamo



a calcolare le soluzioni ottime dei rispettivi rilassati continui con l'aggiunta del vincolo

- La soluzione ottima del rilassamento continuo di $P_{1,1}$ è $(1,0,1,1/6)$, quindi $v_S(P_{1,1}) = 32 \geq 28$, pertanto il nodo $P_{1,1}$ rimane aperto.
- La soluzione ottima del rilassamento di $P_{1,2}$ è $(1,1,1/2,0)$, quindi $v_S(P_{1,2}) = 32 > 28$, pertanto il nodo $P_{1,2}$ rimane aperto.

Dal nodo $P_{1,1}$ istanziamo la variabile x_4, mentre dal nodo $P_{1,2}$ istanziamo la variabile x_3. Andiamo



a calcolare le soluzioni del rilassamento continuo introducendo i nuovi vincoli

- La soluzione ottima del rilassamento $P_{2,1}$ è $(1,0,1,0)$, quindi $v_S(P_{1,2}) = 28 \leq v_I(P)$, pertanto il ramo $P_{2,1}$ deve essere chiuso.
- La soluzione ottima del rilassamento di $P_{2,2}$ è $(1/2,0,0,1)$, quindi $v_S(P_{2,2}) = 29 > v_I(P)$, pertanto il ramo può rimanere aperto.
- La soluzione ottima del rilassamento di $P_{2,3}$ è $(1,1,0,1/3)$, quindi $v_S(P_{2,3}) = 29 > v_I(P)$, pertanto il ramo può rimanere aperto.
- La soluzione del rilassamento di $P_{2,4} = 31$, oltre a essere una soluzione ammissibile del rilassato continuo, è anche una soluzione ammissibile del problema di PLI; inoltre, è anche maggiore rispetto alla soluzione ammissibile usata fino a questo momento. Di conseguenza aggiorniamo la soluzione e $v_I(P) = 31$ e chiudiamo il nodo $P_{2,4}$.

  Dopo di che controlliamo se i nodi aperti $P_{2.2}$ e $P_{2,3}$ sono ancora validi rispetto alla nuova valutazione.

– $v_S(P_{2.2}) \leq v_I(P)$.
– $v_S(P_{2.3}) \leq v_I(P)$.

Alla luce di questa nuova valutazione, entrambi i nodi vengono chiusi.

La soluzione ottima è quindi $(0, 1, 1, 0)$ con valore ottimo 31.

3 Problema dello zaino multidimensionale

Il modello di zaino che abbiamo analizzato finora prevede soltanto il rispetto del vincolo di capacità, ossia che la somma dei pesi degli oggetti presenti nello zaino non superi la capacità dello zaino stesso:

$$\sum_{i=1}^{m} p_i x_i \leq C \tag{26}$$

Possiamo, tuttavia, aggiungere anche altri vincoli. Supponiamo ora che lo zaino abbia anche un volume definito Q e che gli oggetti occupino un volume $q_i \geq 0$. Come possiamo modificare il modello dello zaino (prendendo come esempio quello binario) per introdurre questo vincolo?

$$\begin{cases} \max \sum_{i=1}^{m} v_i x_i \\ \sum_{i=1}^{m} p_i x_i \leq C \\ x_i \in \{0, 1\} \end{cases} \xrightarrow{\text{Aggiunta del vincolo}} \begin{cases} \max \sum_{i=1}^{m} v_i x_i \\ \sum_{i=1}^{m} p_i x_i \leq C \\ \sum_{i=1}^{m} q_i x_i \leq Q \\ x_i \in \{0, 1\} \end{cases} \tag{27}$$

Le considerazioni fatte sulla variazione del modello comportano anche modifiche nel calcolo delle valutazioni. In particolare, è necessario soffermarsi sull'utilizzo dei rendimenti. Non è più possibile utilizzare il rendimento come rapporto tra il **valore del bene** e il **peso del bene**; è quindi necessario definire un nuovo tipo di rendimento. Una possibilità è quella di definirlo come:

$$r_i = \frac{v_i}{q_i + p_i} \tag{28}$$

Questa variante presenta, tuttavia, un caso limite problematico: quando si hanno prodotti con volumi molto piccoli e pesi molto grandi, o viceversa. L'opzione migliore consiste nel calcolare una *media pesata* tra il volume e il peso del bene, i coefficienti devono essere scelti dall'utenti valutando quali prodotti abbiamo da inserire. Definiamo d_i come il divisore del rendimento:

$$d_i = \frac{s_1 q_i + s_2 p_i}{s_1 + s_2} \tag{29}$$

dove s_1 e s_2 rappresentano i pesi, in termini di media pesata, assegnati rispettivamente al volume e al peso. Possiamo quindi descrivere la nuova formula dei rendimenti come:

$$r_i = \frac{v_i}{d_i} = \frac{(s_1 + s_2) v_i}{s_1 q_i + s_2 p_i} \tag{30}$$

Capitolo 8

Il problema del *Commesso Viaggiatore*

Consideriamo un grafo orientato completo

Definition 0.1. (**Grafo orientato**) Un grafo **orientato** è una **coppia** (N, A), dove **N è un insieme di nodi** e $A \subseteq N \times N$ è un **insieme di archi**, cioè, coppie ordinate di nodi in N. Ad ogni nodo è associata un etichetta, definita come il suo **costo**.

Un ciclo orientato che passa per tutti i nodi del grafo una e una sola volta è detto **ciclo hamiltoniano** e il suo costo è definito come la somma dei costi degli archi da cui è formato.

Il problema di trovare il **ciclo hamiltoniano** di costo minimo viene chiamato problema del **commesso viaggiatore(***Traveling Salesman Problem, TPS***)**, perché un esempio tipico di tale situazione si può trovare nel problema che ha un commesso viaggiatore che deve visitare un numero fissato di città una ed una sola volta.

Come nel caso del problema dello zaino distinguiamo due casi differenti del problema del *commesso viaggiatore*. Se per ogni arco $(i, j) \in A$ si ha che $c_{ij} = c_{ji}$, allora il problema è detto **simmetrico**, altrimenti viene detto **asimmetrico**.

1 Problema *asimmetrico*

Rappresentiamo un **ciclo hamiltoniano** $\mathcal{C}$ mediante le variabili binarie x_{ij}, dove

$$x_{ij} = \begin{cases} 1 \text{ se } (i,j) \in \mathcal{C} \\ 0 \text{ altrimenti} \end{cases} \tag{1}$$

Una volta data la variabile andiamo a costruire un modello di problema; partiamo dalla funzione obiettivo. Presumendo che il nostro obiettivo sia quello di minimizzare la distanza percorsa, la nostra funzione obiettivo sarà

$$\min \sum_{(i,j)\in A} c_{ij} x_{ij} \tag{2}$$

quindi dobbiamo minimizzare la somma dei costi del **ciclo hamiltoniano** andando a variare gli archi che vengono attraversati. Il nostro primo vincolo è che tutti i nodi siano attraversati almeno una volta

$$\begin{aligned} \sum_{i\in N, i\neq j} x_{ij} = 1 \ \ \forall j \in N \\ \sum_{j\in N, i\neq j} x_{ij} = 1 \ \ \forall i \in N \end{aligned} \tag{3}$$

il primo vincolo stabilisce che il **ciclo hamiltoniano** abbia un solo arco entrante per ogni nodo $j \in N$, mentre il secondo stabilisce che il **ciclo hamiltoniano** abbia un solo nodo uscente per ogni nodo $i \in N$. Infine, l'ultimo vincolo da aggiungere è quello definito **vincolo di connessione**, il

quale impone che il **ciclo hamiltoniano** abbia almeno un arco uscente da ogni sottoinsieme non vuoto si S nodi, in modo da evitare la formazione di cicli orientati che non passano per tutti i nodi.

$$\sum_{i \in S} \sum_{j \notin S} x_{ij} \geq 1 \ \ \forall S \subset N, S \neq \emptyset \tag{4}$$

Quindi, il modello per il problema del *commesso viaggiatore* è

$$\begin{cases} \min \sum_{(i,j) \in A} c_{ij} x_{ij} \\ \sum_{i \in N, i \neq j} x_{ij} = 1 \ \ \forall j \in N \\ \sum_{j \in N, i \neq j} x_{ij} = 1 \ \ \forall i \in N \\ \sum_{i \in S} \sum_{j \notin S} x_{ij} \geq 1 \ \ \forall S \subset N, S \neq \emptyset \\ x_{ij} \in \{0,1\} \ \ \forall (i,j) \in A \end{cases} \tag{5}$$

Per applicare efficacemente l'algoritmo del *Branch-and-Bound* è fondamentale disporre di valutazioni inferiori buone. Un approccio comune consiste nel rimuovere il vincolo di interezza, risolvendo così il corrispondente **rilassamento continuo**. Tuttavia, il vincolo di connessione risulta particolarmente complesso da modellare. Per ovviare a tale difficoltà, si opta per la sua rimozione, riducendo il problema a un problema di assegnamento di costo minimo, più semplice da trattare computazionalmente.

$$\begin{cases} \min \sum_{(i,j) \in A} c_{ij} x_{ij} \\ \sum_{i \in N, i \neq j} x_{ij} = 1 \ \ \forall j \in N \\ \sum_{j \in N, i \neq j} x_{ij} = 1 \ \ \forall i \in N \\ x_{ij} \in \{0,1\} \ \ \forall (i,j) \in A \end{cases} \tag{6}$$

Dal punto di vista statistico, è possibile dimostrare che il problema di assegnamento fornisce valutazioni inferiori di alta qualità per il *TSP asimmetrico*. Tuttavia, la soluzione ottima dell'assegnamento di costo minimo può **violare il vincolo di connessione**, determinando la presenza di **sotto-cicli disgiunti**. Per ottenere una stima superiore, a partire da questa valutazione inferiore, si ricorre all'*algoritmo delle toppe*, progettato per correggere tali violazioni, pemrettendo di arrivare ad una valutazione superiore.

Algoritmo delle toppe

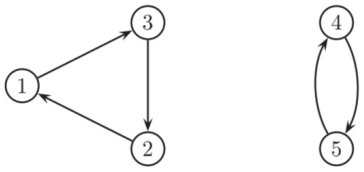
1. Sia $C = \{C_1, \ldots, C_p\}$ l'insieme dei cicli orientati corrispondenti alla soluzione ottima del (7).
2. Per ogni coppia di cicli $C_h, C_k \in C$, valuta l'incremento di costo γ_{hk} corrispondente alla fusione tra i due cicli nel modo più conveniente possibile.
3. Effettua la fusione tra C_h e C_k ai quali corrisponde il minimo valore di γ_{hk}. Aggiorna C e pone $p = p - 1$.
4. Se $p = 1$ allora abbiamo finito, altrimenti torna al passo 2.

Esempio di applicazione del branch and bound sul TSP Asimmetrico

Proviamo a fare un esempio, risolviamo con l'algoritmo del *Branch-and-Bound* il TPS asimmetrico su un grafo orientato descritto dalla seguente matrice di adiacenza

	1	2	3	4	5
1	0	33	13	25	33
2	33	0	46	58	76
3	39	33	0	12	30
4	35	29	12	0	23
5	60	54	30	23	0

Per prima cosa troviamo la valutazione inferiore del valore ottimo andando a risolvere il rilassamento (7). La soluzione ottima è quella data dai due cicli disgiunti descritti in figura



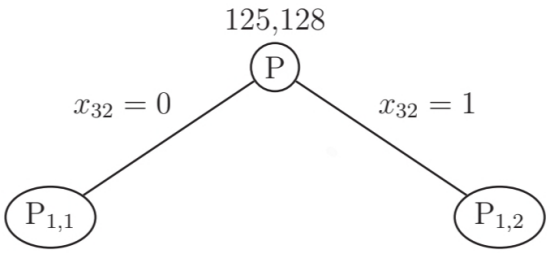
Il valore di tale soluzione è quindi $v_I = 125$.

Per ottenere una valutazione superiore applichiamo l'algoritmo delle toppe. Poiché la soluzione ottima del problema di assegnamento contiene solo due cicli disgiunti, dobbiamo decidere quale fusione sia più conveninente. Le possibili fusioni sono

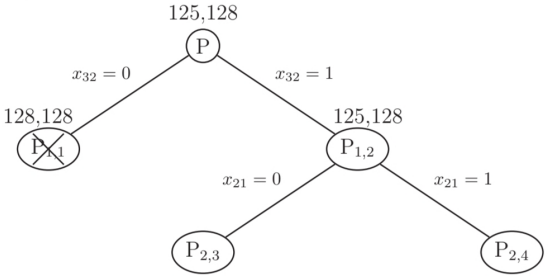
- Sostituire gli archi $(1,3)$ e $(4,5)$ con gli archi $(1,5)$ e $(4,3)$, ottenendo il ciclo hamiltoniano 1-5-4-3-2 di costo 134.
- Sostituire gli archi $(1,3)$ e $(5,4)$ con gli archi $(1,4)$ e $(5,3)$, ottenendo il ciclo hamiltoniano 1-4-5-3-2 di costo 144.
- Sostituire gli archi $(2,1)$ e $(4,5)$ con gli archi $(2,5)$ e $(4,1)$, ottenendo il ciclo hamiltoniano 1-3-2-5-4 di costo 180.
- Sostituire gli archi $(2,1)$ e $(5,4)$ con gli archi $(2,4)$ e $(5,1)$, ottenendo il ciclo hamiltoniano 1-3-2-4-5 di costo 187
- Sostituire gli archi $(3,2)$ e $(4,5)$ con gli archi $(3,5)$ e $(4,2)$, ottenendo il ciclo hamiltoniano 1-3-5-4-2 di costo 128.
- Sostituire gli archi $(3,2)$ e $(5,4)$ con gli archi $(3,4)$ e $(5,2)$, ottenendo il ciclo hamiltoniano 1-3-4-5-2 di costo 135.

Ricaviamo che la fusione più conveniente è quella con cui si ottiene il ciclo 1-3-5-4-2, quindi $v_S = 128$.

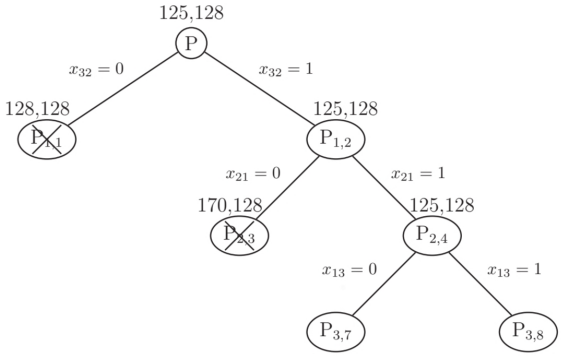
Possiamo iniziare ora l'esplorazione dell'albero di enumerazione totale



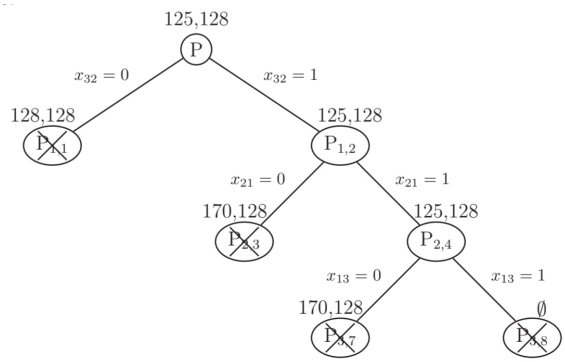
Per ottenere una valutazione inferiore del valore ottimo di $P_{1,1}$ è sufficiente risolvere il problema (7) con il vincolo aggiuntivo $x_{32} = 0$. Il valore ottimo di tale problema è 128, ossia, $v_I(P_{1,1}) = v_S(P)$, quindi chiudiamo il nodo $P_{1,1}$. Osserviamo anche che la soluzione ottima del rilassamento di (P) contiene anche l'arco $(3, 2)$, quindi essa risolve anche il rilassamento di $P_{1,2}$. Otteniamo così $v_I(P_{1,2}) = 125 \leq V_S(P)$ e quindi rimane aperto il nodo $P_{1,2}$. Fissiamo ora x_{21}



Risolvendo il rilassamento di $P_{2,3}$ otteniamo $v_I(P_{2,3}) = 170 > 128 = v_S(P)$, quindi chiudiamo il nodo $P_{2,3}$. Poiché la soluzione ottima del rilassato continui di (P) contiene sia l'arco $(2, 3)$ che l'arco $(2, 1)$, essa è ottima anche per il rilassamento di $P_{2,4}$ e quindi $v_I(P_{2,4}) = 125 \leq 128 = v_S(P)$, quindi non possiamo chiudere il nodo $P_{2,4}$. Fissiamo ora la variabile x_{13}



Risolvendo il rilassamento di $P_{3,7}$ otteniamo $v_I(P_{3,7}) = 170 > 128 = v_S(P)$, di conseguenza chiudiamo il nodo $P_{3,7}$. Anche il nodo $P_{3,8}$ può essere chiuso in quanto non contiene nessuna soluzione ammissibile.



Concludiamo quindi che la soluzione ottima del problema di TSP è il **ciclo hamiltoniano** 1-3-5-4-2 di costo 128

2 Problema *simmetrico*

Il TSP simmetrico potrebbe essere trattato come un caso particolare del TSP **asimmetrico**, tuttavia è preferibile formularlo in maniera più efficace. Essendo la matrice dei costi simmetrica (quindi $c_{ij} = c_{ji}$ possiamo *supporre che gli archi non siano orientati* e che l'arco che collega due nodi i e j sia indicato con (i, j). Per individuare un **ciclo hamiltoniano** $\mathcal{C}$ usiamo come prima variabile x_{ij}

$$x_{ij} = \begin{cases} 1 \text{ se } (i,j) \in \mathcal{C} \\ 0 \text{ altrimenti} \end{cases} \tag{7}$$

Non essendo gli archi orientati, il numero di **cicli hamiltoniani** si dimezza, così come anche il numero di variabili del nostro sistema.

Il modello matematico per il TSP simmetrico può essere formulato partendo dalla funzione obiettivo. Come nel caso asimmetrico, il nostro obiettivo è quello di trovare il **ciclo hamiltoniano** di costo minimo

$$\min \sum_{(i,j)\in A} c_{ij} x_{ij} \tag{8}$$

Il primo vincolo stabilisce che il grado di ogni nodo sia 2, quindi, che ci siano due archi incidenti su ogni nodo

$$\sum_{(i,k)\in A} x_{ik} + \sum_{(p,i)\in A} x_{pi} = 2 \quad \forall i \in N \tag{9}$$

Il secondo vincolo che andiamo a definire viene detto *vincolo di connessione*, ossia impone che, comunque si scelga un sottoinsieme S di nodi, esista almeno un arco entrante in S o un arco uscente da S.

$$\sum_{(i,j)\in A, i\in S, j\notin S} x_{ij} + \sum_{(i,j)\in A, i\notin S, j\in S} x_{ij} \geq 1 \tag{10}$$

Descriviamo ora un metodo euristico per trovare un **ciclo hamiltoniano**, l'*algoritmo del nodo più vicino*. Attraverso l'algoritmo siamo in grado di una valutazione superiore (quindi una soluzione ammissibile) del problema di TSP simmetrico. L'idea dell'algoritmo è quella di partire da un nodo s_1 collegandolo al nodo più vicino s_2 usando come metrica per le distanze i costi c_{ij}. Successivamente, sempre con la stesa metrica, si collega il nodo s_2 al più vicino s_3 escludendo quelli già raggiunti; l'algoritmo termina quando abbiamo raggiunto tutti i nodi.

Algoritmo del nodo più vicino

Si parte scegliendo un nodo $i \in N$, poniamo $s_i := i$, $N := N - \{i\}$, $k := 1$. Successivamente, fintanto che l'insieme $N \neq \emptyset$:

- Troviamo un nodo $j \in N$ con la distanza minima dal nodo s_k
- Poniamo $s_{k+1} = j$, $N := N - \{j\}$ e $k = k + 1$.

Per trovare una valutazione inferiore del problema di TSP simmetrico procediamo a definire un rilassamento, che fornisce buone valutazioni inferiori

$$\begin{cases} \min \sum_{(i,j)\in A} c_{ij} x_{ij} \\ \sum_{(i,k)\in A} x_{ik} + \sum_{(i,p)\in A} x_{ip} = 2 \quad \forall i \in N \\ \sum_{(i,j)\in A, i\in S, j\notin S} x_{ij} + \sum_{(i,j)\in A, i\notin S, j\in S} x_{ij} \geq 1 \\ x_{ij} \in \{0,1\} \end{cases} \tag{11}$$

Per risolvere questo problema senza utilizzare il rilassamento possiamo utilizzare un algoritmo greedy: l'**algoritmo di Kruskal**. Possiamo giustificare l'utilizzo dell'algoritmo notando che **il rilassamento corrisponde al problema dell'**r-albero **di costo minimo**.

Definition 2.1. (**r-albero**) Un r-albero è un insieme di n archi di cui

- $n-2$ archi formano un albero di copertura sul sottografo formato dai nodi $N-\{r\}$.
- 2 archi sono incidenti sul nodo r.

Per completezza, diamo anche una definizione di **albero di copertura** di un grafo

Definition 2.2. (**Albero di copertura**) Si definisce come albero di copertura di un grafo è un insieme di archi connesso e privo di cicli che contiene tutti i nodi del grafo.

Quindi, per risolvere il problema dell'r-albero di costo minimo è sufficiente prendere due archi di minimo costo incidenti sul nodo r ed un albero di copertura di costo minimo T sul **sottografo** $N\backslash\{r\}$. L'albero di copertura di costo minimo può essere ottenuto facilmente utilizzando un algoritmo **greedy**, chiamato **algoritmo di Kruskal**

Algoritmo di Kruskal

Siano $n = |N|$ e $m = |A|$. Si ordina gli archi $a_1, \ldots, a_m$ di A in ordine crescente di costo. Poniamo inoltre $T := \{a_1\}$ e $k = 2$. Si controlla se a_k non forma un ciclo con gli archi presenti in T:

- Se non forma un ciclo allora $T = T \cup \{a_k\}$ e $k = k + 1$;
- Se forma un ciclo non si fa niente.

Adesso che abbiamo trovato un modo per ottenere una valutazione inferiore (quindi la soluzione del rilassato continuo) e una valutazione superiore (quindi una soluzione ammissibile possiamo procedere applicando il **branch and bound** al problema nello stesso identico modo del problema asimmetrico.

Capitolo 9

Problema del Bin-Packing

In generale, un problema di **bin-packing** può essere formulato nel seguente modo

> Dati n oggetti di peso $p_1, \ldots, p_n$ e m contenitori ognuno di capacità C. Trovare il numero minimo di contenitori in cui inserire tutti gli oggetti.

Introduciamo due variabili, la prima x_{ij} ci dice se l'oggetto j è inserito nel contenitore j, la seconda y_i ci dice se il contenitore i è usato.

$$x_{ij} = \begin{cases} 1 \text{ se oggetto j sta in contenitore i} \\ 0 \text{ altrimenti} \end{cases} \qquad y_i = \begin{cases} 1 \text{ se i è usato} \\ 0 \text{ altrimenti} \end{cases} \tag{1}$$

i vincoli che devono essere rispettati da queste variabili sono

$$\begin{aligned} &\sum_{i=1}^{m} x_{ij} = 1 \ \ \forall j = 1, \ldots, n \\ &\sum_{j=1}^{n} p_j x_{ij} > C y_i \\ &x_{ij} \in \{0,1\} \ \ \forall i, j \\ &y_i \in \{0,1\} \ \ \forall i \end{aligned} \tag{2}$$

L'obiettivo del problema è quello di minimizzare il numero di contenitori utilizzati; la funzione obiettivo può essere scritta come

$$\min \sum_{i=1}^{m} y_i \tag{3}$$

Unendo vincoli e funzione obiettivo riusciamo a ricavare il modello matematico corrispondente

$$\begin{cases} \min \sum_{i=1}^{m} y_i \\ \sum_{i=1}^{m} x_{ij} = 1 \ \ \forall j = 1, \ldots, n \\ \sum_{j=1}^{n} p_j x_{ij} > C y_i \\ x_{ij} \in \{0,1\} \ \ \forall i, j \\ y_i \in \{0,1\} \ \ \forall i \end{cases} \tag{4}$$

Per andare a risolvere il problema sfruttando l'algoritmo del *Branch-and-Bound* devo, prima di tutto, determinare una **stima superiore** e una **stima inferiore** del problema. Determinare una valutazione superiore è un'operazione che sfrutta uno tra questi tre algoritmi *greedy*

- Algoritmo **Next-Fit Decreasing** (*NFD*): Esamina gli oggetti in ordine di peso decrescente e considera il primo contenitore come **contenitore corrente**. Se possibile, assegna un oggetto al contenitore corrente; altrimenti lo assegna ad un nuovo contenitore, che diventa quello corrente.

j	1	2	3	4	5	6	7	8	9	
p_j	70	60	50	33	33	33	11	7	3	$C = 100$

Contenitori	Oggetti	Capacità residua
1	1	30
2	2	40
3	3 4	50 17
4	5 6 7 8 9	67 34 23 16 13

- Algoritmo **First-Fit Decreaging** (*FFD*): Esamina gli oggetti in ordine di peso decrescente e assegna ogni oggetto al primo contenitore **usato** che può contenerlo. Se nessuno di essi può contenerlo, assengna l'oggetto ad un nuovo contenitore.

j	1	2	3	4	5	6	7	8	9	
p_j	70	60	50	33	33	33	11	7	3	$C = 100$

Contenitori	Oggetti	Capacità residua
1	1 7 8 9	30 19 12 9
2	2 4	40 7
3	3 5	50 17
4	6	67

- Algoritmo **Best-Fit Decreasing** (*BFD*): Esamina gli oggetti in ordine di peso decrescente. Tra tutti i contenitori usati sceglie quello con la capacità residua massima. se nessuno di essi può contenerlo, assegna l'oggetto a un nuovo contenitore.

j	1	2	3	4	5	6	7	8	9	
p_j	70	60	50	33	33	33	11	7	3	$C = 100$

Contenitori	Oggetti	Capacità residua
1	1	30
2	2 4 8	40 7 0
3	3 5 7 9	50 17 6 3
4	6	67

D'altra parte per ottenere una valutazione inferiore dobbiamo procedere andando a calcolare una soluzione del **rilassato continuo**. Il modello matematico del rilassato è

$$\begin{cases} \min \sum_{i=1}^{m} y_i \\ \sum_{i=1}^{m} x_{ij} = 1 \ \ \forall j = 1, \ldots, n \\ \sum_{j=1}^{n} p_j x_{ij} > C y_i \\ 0 \leq x_{ij} \leq 1 \ \ \forall i, j \\ 0 \leq y_i \leq 1 \ \ \forall i \end{cases} \tag{5}$$

Esiste anche un caso particolare di cui tenere conto, ...; il questo caso la valutazione inferiore diventa

$$v_I = \frac{\sum_{i=1}^{n} p_i}{C} \tag{6}$$

Capitolo 10

Problema del *facility-location*

L'ultimo problema che affronteremo a riguardo della PLI è il problema della **facility-location**. Questa tipologia di problemi riguarda il posizionamento **ottimale** di serivizi. Applicazioni di questa classe di problema si trovano in diversi ambiti della realtà, come, ad esempio: *il problema del posizionamento ottimale di una stazione dei vigili del fuoco in una cittadina.* I dati che ci vengono dati sono

Problema di *facility-location*

- **Insieme di nodi domanda** (I).
- **Insieme dei siti candidati ad ospitare il seriizio** (J).
- costo per aprire il servizio nel sito (c_j).
- distanza tra il sito candidato j e il nodo di domanda i (d_{ij}).
- distanza di copertura D; diciamo che i è coperto da j se e solo se $d_{ij} \leq D$.

Definiamo a_{ij} come una variabile tale per cui

$$a_{ij} = \begin{cases} 1 & \text{se } d_{ij} \leq D \\ 0 & \text{altrimenti} \end{cases}$$

Quindi, a_{ij} vale 1 se aprendo il servizio nel sito j riusciamo a arrivare al nodo di domanda i. Definiamo anche la variabile x_j che vale 1 se e solo se apriamo il servizio nella zona j.

Prima di analizzare il metodo per calcolare le valutazioni superiori e inferiori andiamo a enunciare alcune semplici regole di semplificazioni del problema

- Presa una riga A_i della matrice, tale per cui

$$a_{ij} = 1 \quad \forall j = 1, \ldots, m \tag{1}$$

  la posso eliminare. Il motivo è presto detto, se il nodo i viene servito indipendentemente da dove metto il servizio, non è necessario considerarlo ai fini del calcolo della soluzione.

- Presa una riga A_i della matrice, tale per cui

$$a_{ij} = 0 \quad \forall j = 1, \ldots, m \tag{2}$$

  la posso eliminare. Anche qui il motivo è presto detto, se il nodo i è inservibile da ogni sito candidato il poliedro è vuoto; però possiamo rimuoverlo e trovare la soluzione ottima considerando solo quei nodi che possono essere serviti da almeno un sito.

- Presa una riga A_i della matrice, tale per cui

$$\sum_{j=1}^{m} a_{ij} = 1 \tag{3}$$

la posso eliminare e pongo la x_j tale per cui $a_{ij} = 1$, uguale a 1. Anche qui il motivo è semplice, se il nodo i è servibile da un solo j

- L'ultima regola è più complessa da descrivere. Supponiamo di avere due indici, rispettivamente, k e r; supponiamo di avere la *k-esima* colonna di A e la *r-esima* colonna di A, chiamate, rispettivamente A^k e A^r. Secondo la regola di dominanza, possiamo eliminare la colonna **che contiene il minor numero di 1 se e solo se** $c_k \leq c_r$

Una volta ridotto il nostro problema possiamo trovare le rispettive valutazioni. Essendo un problema di minimo abbiamo la valutazione inferiore ottenibile come arrotondamento per difetto del rilassamento continuo del problema

$$\begin{cases} \min c^T x \\ Ax \geq 1 \\ 0 \leq x \leq 1 \end{cases} \tag{4}$$

La valutazione superiore è ottenibile attraverso un algoritmo greedy che genera una soluzione ammissibile del problema

Algoeritmo di Cheval

Definisco una grandezza, chiamata **costo unitario** e definita come il rapporto tra il costo per aprire un servizio in un determinato sito e il numero di aree servite

$$C_j^V = \frac{c_j}{\sum_{i=1} a_{ij}} \tag{5}$$

Successivamente procedo a aprire il servizio con C^V minimo e cancello la colonna j e tutte le righe servite aprendo il servizio in j.

Continuo con l'algoritmo iterando fino a finire le righe.

Una piccola osservazione è quella sul procedimento per ottenere la matrice A e il vettore b del nostro problema. Prendiamo un esempio; Una ASL deve decidere dove posizionare alcune ambulanze in una città. Esistono 5 possibili siti in cui posizionare le ambulanze (massimo una per sito) e 10 zone da coprire. I tempi (in minuti) per raggiungere le zone sono date dalla seguente tabella

	siti				
zone	1	2	3	4	5
1	2	7	13	11	12
2	5	13	9	12	11
3	7	15	2	16	4
4	19	7	6	18	2
5	11	2	12	8	19
6	12	18	9	10	11
7	13	17	6	18	7
8	18	12	12	13	6
9	11	6	7	8	9
10	8	14	5	6	13

L'ASL deve posizionare le ambulanze in modo che ogni zona possa essere raggiunta da un'ambulanza entro 10 minuti. Infine, i costi per posizionare le varie ambulanze sono invece

	siti				
costo	6	9	10	8	7

Il primo passo è quello di calcolarci la matrice di copertura A utilizzando le regole viste in precedenza per la variabile a_{ij} Non ci sono righe con tutti 1 o con tutti 0, ma notiamo che vi è la

	siti				
zone	1	2	3	4	5
1	0	0	1	1	1
2	0	1	0	1	1
3	0	1	0	1	0
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	0	1	0
8	1	1	1	1	0
9	1	0	0	0	0
10	0	1	0	0	1

riga corrispondente al nodo 9 che presenta un solo 1; affinché questa zona sia coperta è necessario che $x_1 = 1$.

Una variante di questo problema è il **problema di massima copertura** in presenza di un **budget limitato** e in presenza di un numero prefissato di posti dove aprire il servizio. In questo modello non abbiamo più il vettore dei costi, ma il vettore h_i associato alla domanda del nodo *i-esimo*. I dati de problema sono quindi

Problema di massima copertura

- L'insieme dei nodi di domanda I.
- Il vettore h, dove l'elemento *i-esimo* è definito come la domanda relativa al nodo *i-esimo*.
- L'insieme J dei siti candidati ad aprire il servizio.
- Il numero p prefissato di posti in cui aprire il servizio.
- La distanza d_{ij} che separa il nodo di domanda *i-esimo* dal nodo di offerta *j-esimo*.
- La distanza di copertura D.

L'idea alla base di questo problema è quella di massimizzare la *domanda soddisfatta*, cerchiamo di aprire il servizio in modo da coprire le zone con la domanda maggiore. Definiamo una nuova variabile z_i, la quale ci permette di identificare se il nodo *i-esimo* è coperto dal servizio

$$z_i = \begin{cases} 1 & \text{se il nodo } i\text{-esimo è coperto} \\ 0 & \text{altrimenti} \end{cases} \tag{6}$$

La funzione obiettivo che dobbiamo massimizzare è

$$\max \sum_{i \in I} h_i z_i \tag{7}$$

Inoltre, il vincolo che abbiamo in più rispetto al problema di *facility-location* è che vi sia un numero prefissato di siti in cui aprire il servizio

$$\sum_{j \in J} x_j = p \tag{8}$$

L'ultimo vincolo che dobbiamo definire è un vincolo che ci garantisce che, se il nodo *i-esimo* è coperto (quindi $z_i = 1$) allora esista effettivamente un sito di copertura *j-esimo* che è in grado di

coprire quel nodo *i-esimo*

$$\sum_{j \in J} a_{ij} x_j \geq z_i \quad \forall i \in I \tag{9}$$

La condizione potrebbe risultare ostica da comprendere, proviamo a comprenderla in un modo quanto più intuitivo possibile, se abbiamo $z_i = 1$, quindi il sito è aperto, allora dovrà necessariamente esistere un sito che copre questo servizio, ergo, esisterà almeno un $x_j = 1$. Per verificarlo vado nella matrice e prendo la *i-esima* riga e la scorro controllando gli elementi a_{ij} relativi a quella zona e moltiplico ogni singolo elemento per il rispettivo x_j. Se, come avevamo detto, $h_i = 1$, allora **almeno** uno di questi prodotti deve fare 1 e quindi abbiamo verificato la condizione, perché vuol dire che quella zona è coperta da almeno uno dei nodi che offrono il servizio. Il modello della copertura massima è quindi

$$\begin{cases} \max \sum_{i \in I} h_i z_i \\ \sum_{j \in J} a_{ij} x_j \geq z_i \\ \sum_{j \in J} x_j = p \\ x_j \in \{0, 1\} \\ z_i \in \{0, 1\} \\ a_{ij} \in \{0, 1\} \end{cases} \tag{10}$$

Questo problema presenta un'ulteriore variante, che però non verrà affrontata durante questo corso; in questa variantesi suppone che non sia conosciuta a priori la distanza (in minuti) di copertura.

Parte III

Programmazione lineare su reti

Capitolo 11

I problemi della *programmazione lineare su reti*

Una classe interessante di problemi di PL trova una sua modellizzazione tramite il concetto di grafo

Definition 0.1. (**Grafo**) Si definisce come grafo una coppia (V, A), dove V identifica un insieme $V = \{v_1, v_2, \ldots, v_n\}$ detti **vertici** (o *nodi*), mentre $A \subseteq V \times V$ è un insieme di **spigoli** (o *archi*), nello specifico, coppie ordinati di vertici appartenenti a V

Durante la trattazione dei problemi di PLR sfrutteremo altre due definizioni:

- Un nodo $v \in V$ si definisce un **pozzo** se e solo se tutti gli archi incidenti su v sono **entranti**.
- Un nodo $v \in V$ si definisce un **sorgente** se e solo se tutti gli archi incidenti su v sono **uscenti**.

In questo paragrafo ci limiteremo solo a dare una breve trattazione dei problemi; nei paragrafi successivi invece, andremo a dare una spiegazione più dettagliata di alcuni di questi.

1 Problema del flusso di costo minimo

Supponiamo che su ogni arco (i, j) siano definiti un costo c_{ij} per unità di flusso, una capacità inferiore l_{ij} ed una superiore u_{ij} del flusso e, supponiamo, che sui nodi siano assegnati dei bilanci; nello specifico

- Se il **bilancio è positivo** il nodo è un **pozzo**.
- Se il **bilancio è negativo** il nodo è una **sorgente**.

Introduciamo una variabile x_{ij} che identifica il valore del flusso lungo uno specifico arco $(i, j) \in A$. Il problema che ci poniamo è quello di determinare il **flusso di costo minimo**. Quindi, il nostro problema è un **problema di minimo** e la funzione obiettivo è definita come

$$\min \sum_{(i,j)\in A} c_{ij} x_{ij} \tag{1}$$

quindi, la somma finita del prodotto tra il flusso che passa su un nodo e il suo costo (che, ricordiamoci, è la grandezza che vogliamo **minimizzare**). Ua volta determinata la funzione obiettivo, passiamo a determinare i vincoli. Il primo vincolo che vogliamo definire è detto **vincolo di capacità**; inizialmente abbiamo detto che per ogni arco sono definite due grandezze: una capacità inferiore l_{ij} ed una superiore u_{ij} del flusso. Occorre quindi definire il modello matematico del vincolo

$$l_{ij} \leq x_{ij} \leq u_{ij} \tag{2}$$

Infine, l'ultimo vincolo che vogliamo definire è quello **di bilancio**, vogliamo assicurarci che la somma dei flussi entranti e uscenti su ogni nodo sia uguale al bilancio che abbiamo definito all'inizio su ogni nodo

$$\sum_{(p,i)\in A} x_{pi} - \sum_{(i,q)\in A} x_{iq} = b_i \ \ \forall i \in N \tag{3}$$

Possiamo adesso scrivere il modello matematico generale del problema

$$\begin{cases} \min \sum_{(i,j)\in A} c_{ij}x_{ij} \\ \sum_{(p,i)\in A} x_{pi} - \sum_{(i,q)\in A} x_{iq} = b_i \ \ \forall i \in N \\ l_{ij} \leq x_{ij} \leq u_{ij} \end{cases} \tag{4}$$

Possiamo anche dare a questo problema una notazione compatta. Definiamo la matrice $E \in M^{|N|\times|A|}$ come **matrice di incidenza del grafo**, tale per cui, preso un elemento $p \in N$ e un elemento $(i,j) \in A$ abbiamo che

$$E_{p,(i,j)} = \begin{cases} +1 \text{ se } p = i \\ -1 \text{ se } p = j \\ 0 \text{ altrimenti} \end{cases} \tag{5}$$

Il modello matematico compatto diventa

$$\begin{cases} \min c^T x \\ Ex = b \\ 0 \leq x \leq u \end{cases} \tag{6}$$

Inoltre, durante la nostra trattazione faremo sempre due supposizioni; la prima è che $l_{ij} = 0$ per ogni arco (i,j) e che la somma di tutti i bilanci dei nodi sia nulla.

Capitolo 12

Flusso di costo minimo

Andiamo adesso ad approfondire il problema del flusso di costo minimo. Per completezza, ripetiamo la definizione del modello matematico del problema:

Supponiamo che su ogni arco (i, j) siano definiti un costo c_{ij} per unità di flusso, una capacità inferiore l_{ij} ed una superiore u_{ij} del flusso e, supponiamo, che sui nodi siano assegnati dei bilanci; nello specifico

- Se il **bilancio è positivo** il nodo è un **pozzo**.
- Se il **bilancio è negativo** il nodo è una **sorgente**.

Introduciamo una variabile x_{ij} che identifica il valore del flusso lungo uno specifico arco $(i, j) \in A$. Il problema che ci poniamo è quello di determinare il **flusso di costo minimo**. Il problema può essere così formulato

$$\begin{cases} \min c^T x \\ Ex = b \\ 0 \leq x \leq u \end{cases} \tag{1}$$

Dove x è il **vettore dei flussi**. La matrice $E \in M^{|N| \times |A|}$ è nota come **matrice di incidenza del grafo**; tale per cui, preso un elemento $p \in N$ e un elemento $(i, j) \in A$ abbiamo che

$$E_{p,(i,j)} = \begin{cases} -1 \text{ se } p = i \\ +1 \text{ se } p = j \\ 0 \text{ altrimenti} \end{cases} \tag{2}$$

Quindi, la nostra matrice di incidenza è una matrice che ha per colonne gli archi del grafo e per righe i nodi; inoltre, ogni coefficiente vale 1 se p è l'origine dell'arco, vale invece -1 se p è la destinazione dell'arco. Se non esiste un arco tra due nodi, il coefficiente viene posto a 0.

Il poliedro $P = \{x \in \mathbb{R}^n : Ex = b, 0 \leq x \leq u\}$ è definito **poliedro dei flussi**.

Passiamo adesso ad analizzare la **matrice dei flussi** descrivendone le proprietà; prima di procede però analizziamo due definizioni che torneranno molto utili nel proseguire della trattazione

Definition 0.1. (**Grafo Connesso**) Un grafo si dice connesso se per ogni coppia di nodi del grafo esiste un cammino (*non necessariamente orientano*) che li connette.

Definition 0.2. (Albero di copertura) Dato un grafo $G = (N, A)$, si definisce come albero di copertura un sottinsieme di archi $T \subset A$ tale che il sottografo (N, T) sia connesso e non contenga cicli. Una foglia dell'albero di copertura è un nodo sul quale incide un solo arco T.

Assunte queste due definizioni, posso dire che

- Un **grafo** è **connesso** se e solo se **contiene un albero di copertura**.
- In un **grafo connesso** con n nodi, ogni albero di copertura contiene esattamente $n-1$ archi ed almeno 2 foglie.

D'ora in poi supporremo sempre che il grafo (N, A), con n nodi e m archi sia connesso.

Elenchiamo ora le principali proprietà della matrice di incidenza

Lemma 0.1. Il rango di E è minore o uguale a $(n-1)$ ed il sistema $Ex = b$ è sovradeterminato

Ricordiamo che un sistema è sovradeterminato quando esistono più equazioni che incognite nel sistema. Focalizziamoci un attimo sul valore del rango della matrice, il motivo per cui esso equivale a $(n-1)$ è presto detto.

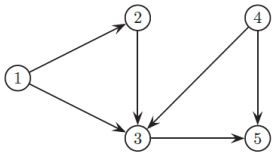
in un grafo, la somma dei flussi in entrata in ogni nodo deve essere uguale alla somma dei flussi in uscita, quindi c'è una dipendenza lineare tra le righe della matrice di incidenza. Questo implica che almeno una delle righe è ridondante e quindi, il rango uguale a $(n-1)$

Passiamo ora enunciare il secondo lemma

Lemma 0.2. Se T è un albero di copertura, allora la sottomatrice quadrata E_T di dimensione $(n-1) \times (n-1)$ ottenuta scegliendo le colonne relative agli archi di T, è invertibile. Inoltre

$$\det(E_T) = \pm 1 \tag{3}$$

Per dimostrarlo dobbiamo scrivere in forma **triangolare inferiore** la matrice di incidenza del sottografo E_T. L'unica possibilità che abbiamo per farlo è effettuando una **visita per foglie** di T. Partiamo scegliendo una qualsiasi foglia z di T e consideriamo l'unico arco a di T che incide sul nodo z. Costruiamo la matrice E_T mettendo il nodo z come prima riga e l'arco a come prima colonna e cancelliamo dal grafo (N, T) il nodo z e l'arco a. Procediamo nello stesso modo andando a inserire la nuova foglia e il nuovo arco sulla, rispettivamente, seconda riga e seconda colonna della matrice. Ripetiamo fintanto che non si ottiene la matrice E in forma triangolare inferiore. Analizziamo un esempio, consideriamo il seguente grafo connesso



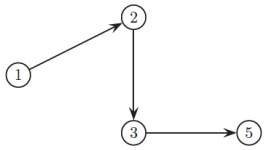
Togliamo la prima riga dalla matrice di incidenza ottenendo

$$E = \begin{array}{c|cccccc} & (1,2) & (1,3) & (2,3) & (3,5) & (4,3) & (4,5) \\ \hline 2 & 1 & 0 & -1 & 0 & 0 & 0 \\ 3 & 0 & 1 & 1 & -1 & 1 & 0 \\ 4 & 0 & 0 & 0 & 0 & -1 & -1 \\ 5 & 0 & 0 & 0 & 1 & 0 & 1 \end{array} \tag{4}$$

L'insieme di archi $T = \{(1,2), (2,3), (3,5), (4,5)\}$ forma un albero di copertura. Dimostriamo ora che la sottomatrice è invertibile scrivendola in forma triangolare inferiore. Prendiamo una foglia diversa dal nodo 1; l'unica scelta possibile ricade sul nodo 4, scegliamo l'unico arco che incide sul nodo 4, quindi l'arco $(4,5)$. Costruiamo E_T mettendo il nodo 4 come prima riga e l'arco $(4,5)$ come prima colonna (ponendo il resto a 0)

$$E = \begin{array}{c|cccc} & (4,5) & & & \\ \hline 4 & -1 & 0 & 0 & 0 \end{array} \tag{5}$$

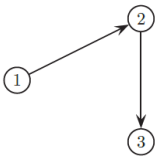
Cancellando dal grafo (N, T) il nodo e l'arco appena inseriti otteniamo



Applichiamo a questo grafo la regola appena vista: scegliamo una foglia diversa dal nodo 1, cioè il nodo 5 e l'unico arco incidente sul nodo 5: $(3, 5)$. Riscriviamo nuovamente la matrice inserendo il nodo nella seconda riga e l'arco nella seconda colonna

$$E = \begin{array}{c|cccc} & (4,5) & (3,5) & & \\ \hline 4 & -1 & 0 & 0 & 0 \\ 5 & 1 & 1 & 0 & 0 \end{array} \tag{6}$$

Eliminando dal grafo (N, T) l'arco e il nodo appena inseriti nella matrice, otteniamo il grafo



Scegliamo la foglia 3 e l'arco $(2, 3)$. La matrice diventa quindi

$$E = \begin{array}{c|cccc} & (4,5) & (3,5) & (2,3) & \\ \hline 4 & -1 & 0 & 0 & 0 \\ 5 & 1 & 1 & 0 & 0 \\ 3 & 0 & -1 & 1 & 0 \end{array} \tag{7}$$

Rimuovendo anche questa foglia e questo arco rimaniamo con la sola scelta della foglia 2 e dell'arco $(1, 2)$

$$E = \begin{array}{c|cccc} & (4,5) & (3,5) & (2,3) & (1,2) \\ \hline 4 & -1 & 0 & 0 & 0 \\ 5 & 1 & 1 & 0 & 0 \\ 3 & 0 & -1 & 1 & 0 \\ 2 & 0 & 0 & -1 & 1 \end{array} \tag{8}$$

Con questa scrittura della matrice E_T possiamo dimostrare che il determinante è pari a ± 1 e che quindi la matrice è invertibile. Notiamo che vale anche l'inverso, se E_T è una matrice invertibile di rango $(n - 1)$ allora T è un albero di copertura.

1 Flusso di costo minimo non capacitato

Trattiamo una variante del problema del flusso, chiamata **problema del flusso di costo minimo non capacitato**. In questa variante assumiamo che ogni arco sia illimitato superiormente, ovvero $u_{ij} = +\infty$. In questo caso il problema può essere semplicemente formulato come

$$\begin{cases} \min c^T x \\ Ex = b \\ x \geq 0 \end{cases} \tag{9}$$

Se infatti provassimo a moltiplicare la nostra matrice di incidenza per il vettore colonna x otterremo il bilancio dei flussi su ogni nodo. La forma del problema è riconducibile alla forma **duale standard**. Le basi del nostro problema coincidono con gli alberi di copertura del grafo

$$\text{B è una base} \iff \text{B è un albero di copertura} \tag{10}$$

Quindi per trovare una base è sufficiente fare una partizione degli archi in due sotto-insiemi (T, L), dove T è un albero di copertura corrispondente. Il relativo flusso di base è quindi

$$\overline{x}_T = E_T^{-1} b \quad \overline{x}_L = 0 \tag{11}$$

Il flusso è ammissibile se e solo se $X_T \geq 0$ ed è degenere se e se solo se esiste un arco $(i, j) \in T$ tale che $\overline{x}_{ij} = 0$.

Theorem 1.1. (Teorema dell'interezza) Se il vettore b è componenti intere, allora anche i flussi di base sono a componenti intere.

Il problema è che la matrice E_T^{-1} potrebbe avere delle dimensioni tali da essere molto complessa da invertire. Nella realtà esiste un altro metodo che ci permette di calcolare delle soluzioni ammissibili (vedere l'appendice).

Il problema che poniamo adesso è capire se presa una base ammissibile, allora essa è anche ottima. Per farlo dobbiamo introdurre un nuovo problema: il problema duale. La struttura di questo problema è così definita

$$\begin{cases} \max \pi^T b \\ \pi^T E \leq c^T \end{cases} \tag{12}$$

ed è chiamato **problema dei potenziali ai nodi della rete**. Ricordiamo un fatto

Il numero di variabili del duale è uguale al numero di vincoli del primale; il numero di variabili del primale è uguale al numero di vincoli del duale

Visto che abbiamo rimosso una delle righe nel nostro problema primale, il problema duale sarà sotto-determinato e di conseguenza, poniamo $\pi_1 = 0$.

Dalle regole definite durante lo studio della programmazione lineare sappiamo che, il potenziale relativo alla base T è la soluzione del sistema

$$\pi^T E_T = c^T \tag{13}$$

Andando ad operare un semplice passaggio algebrico, otteniamo che il potenziale di base è equivalente a

$$\overline{\pi}^T = c_T^T E_T^{-1} \tag{14}$$

che, sempre dalle regole della programmazione lineare, sappiamo essere ammissibile se e solo se tutti i vincoli **non di base** sono rispettati

$$\overline{\pi}^T E_L \leq c_L \tag{15}$$

Definiamo una nuova grandezza, il **costo ridotto** dell'arco (i, j) relativo al potenziale $\overline{\pi}$, definita come

$$C_{(i,j)}^{\overline{\pi}} = c_{ij} + \overline{\pi}_i - \overline{\pi}_j \tag{16}$$

Per gli archi che si trovano in base questa grandezza è sempre nulla, lo si può osservare facilmente dal fatto che se abbiamo una base ammissibile, allora tutti i vincoli di non di base sono risolti con l'uguale; d'altra parte però, non possiamo dire nulla di certo per tutti quegli archi che non si trovano in base; definiamo, quindi, un teorema di ottimalità per il problema dei potenziali

Theorem 1.2. (Teorema di Bellman) Supponiamo che una partizione (T, L) degli archi generi un flusso di base $\overline{x}$ ammissibile. Se il potenziale di base $\overline{\pi}$ è tale che

$$C_{(i,j)}^{\overline{\pi}} \geq 0 \tag{17}$$

allora $\overline{x}$ è ottimo.

Inoltre, se esiste almeno un arco **non in base**, tale per cui, il costo unitario è 0, allora la soluzione è **degenere**. Prima di procedere con un esempio pratico indichiamo una veloce tabella di riferimento per le soluzioni dei problemi di PLR

	flusso di base $\bar{x}_T = E_T^{-1}b \quad \bar{x}_L = 0$	**potenziale di base** $\bar{\pi}^\top = c_T^\top E_T^{-1}$
ammissibile	$\forall (i,j) \in T$ si ha $\bar{x}_{ij} \geq 0$	$\forall (i,j) \in L$ si ha $c_{ij}^{\bar{\pi}} \geq 0$
non ammissibile	$\exists (i,j) \in T$ tale che $\bar{x}_{ij} < 0$	$\exists (i,j) \in L$ tale che $c_{ij}^{\bar{\pi}} < 0$
degenere	$\exists (i,j) \in T$ tale che $\bar{x}_{ij} = 0$	$\exists (i,j) \in L$ tale che $c_{ij}^{\bar{\pi}} = 0$
non degenere	$\forall (i,j) \in T$ si ha $\bar{x}_{ij} \neq 0$	$\forall (i,j) \in L$ si ha $c_{ij}^{\bar{\pi}} \neq 0$

Tabella 12.1: Tabella di riferimento per le soluzioni del problemi di PLR

Un modo estremamente veloce per calcolare i potenziali di base è quello di disegnare l'albero di copertura associato alla nostra base T, dopo di ché, partendo dal nodo a potenziale 0, andare a calcolare tutti i potenziali sfruttando la relazione

$$\pi_j - \pi_i = c_{ij} \tag{18}$$

Procediamo con un esempio pratico

Nel paragrafo della programmazione lineare abbiamo enunciato il **teorema degli scarti complementari**; possiamo definirne una variante analoga anche per il problema dei flussi di costo minimo

Theorem 1.3. (Scarti complementari) Un **flusso ammissibile** $\overline{x}$ è **ottimo** se e solo se

$$\begin{aligned} C_{ij}^{\pi} &\geq 0 && \text{se } x_{ij} = 0 \\ C_{ij}^{\pi} &= 0 && \text{se } x_{ij} > 0 \end{aligned}$$

Analogamente, possiamo dire che un **potenziale di base ammissibile** $\overline{pi}$ è **ottimo** se e solo se

$$\begin{aligned} Ex &= b \\ x_{ij} &\geq 0 && C_{ij}^{\pi} = 0 \\ x_{ij} &= 0 && C_{ij}^{\pi} > 0 \end{aligned}$$

Algoritmo del simplesso per flussi L'algoritmo del simplesso per i flussi, nel caso **non capacitato**, è un algoritmo di simplesso duale **semplificato**. Infatti non dobbiamo calcolarci nessuna matrice e nessun prodotto scalare; sono sufficienti gli algoritmi che abbiamo visto fino ad adesso. Enunciamo i passi

Simplesso per flussi

1. Trovare una partizione degli archi (T, L) con T **albero di copertura**, che generi un **flusso ammissibile**.
2. Calcolare il **flusso di base** $\overline{x}$:

$$\overline{X}_T = E_T^{-1}b \quad \overline{x}_L = 0 \tag{19}$$

abbiamo detto nell'introduzione che non è necessario calcolarsi i prodotti tra matrici, infatti, possiamo tranquillamente calcolare il flusso di base con l'algoritmo visto in precedenza. Successivamente calcoliamo il potenziale di base

$$\overline{\pi}_T = c^T E_T^{-1} \tag{20}$$

Anche in questo caso possiamo utilizzare l'algoritmo che abbiamo descritto in precedenza. Definiamo infine i **costi ridotti**, calcolati su ogni arco **non di base**

$$c = c_{ij} + \overline{\pi}_i - \overline{\pi}_j \tag{21}$$

3. Se $C_{ij}^{\overline{\pi}} \geq 0 \ \forall (i,j) \in L$ ci fermiamo, la base è ottima. Altrimenti calcoliamo l'arco entrante, definito come

$$(p,q) = \min\{(i,j) \in L : C_{ij}^{\overline{\pi}} < 0\} \tag{22}$$

dove il minimo è fatto rispetto all'ordine lessicografico sugli archi. Ricordiamo che l'arco entrante forma un ciclo $\mathcal{C}$ con gli archi dell'albero di copertura T. Fissiamo un *verso* concorde a quello di (p,q) e dividiamo $\mathcal{C}$ in due sotto-insiemi

- $\mathcal{C}^-$: L'insieme di tutti gli archi $(i,j) \in \mathcal{C}$ tale per cui il verso di (i,j) è discorde a quello di (p,q).
- $\mathcal{C}^+$: L'insieme di tutti gli archi $(i,j) \in \mathcal{C}$ tale per cui il verso di (i,j) è concorde a quello di (p,q).

4. Se $\mathcal{C}^- = \emptyset$ ci fermiamo; il flusso di costo minimo ha valore $-\infty$. Altrimenti calcoliamo l'indice entrante; il primo passo è calcolare l'indice θ

$$\theta = \min\{\overline{x}_{ij} : (i,j) \in \mathcal{C}^-\} \tag{23}$$

successivamente calcolo l'**indice entrante**, definito come

$$(r,s) = \min\{(i,j) \in \mathcal{C}^- : \overline{x}_{ij} = \theta\} \tag{24}$$

5. Si aggiorna la partizione degli archi introducendo l'arco entrante e rimuovendo l'arco uscente.

Diamo ora una spiegazione formale dell'algoritmo. L'arco entrante (p,q) forma un ciclo con gli archi dell'albero di T; il costo di tale ciclo è uguale al costo ridotto dell'arco (p,q) che è **negativo**. Quindi spedire il flusso lungo tale ciclo fa **diminuire la funzione obiettivo**.

$$c^T x(\theta) = c^T \overline{x} + \theta C_{(i,j)}^{\pi} \tag{25}$$

Per calcolare un nuovo flusso di base basta spedire θ unità di flusso lungo il ciclo trovato nel cambio di base; fissato un verso concorde a quello dell'arco entrante dividiamo l'insieme $\rfloor$ in due sottoinsiemi

- $\mathcal{C}^-$: L'insieme di tutti gli archi $(i,j) \in \mathcal{C}$ tale per cui il verso di (i,j) è discorde a quello di (p,q).
- $\mathcal{C}^+$: L'insieme di tutti gli archi $(i,j) \in \mathcal{C}$ tale per cui il verso di (i,j) è concorde a quello di (p,q).

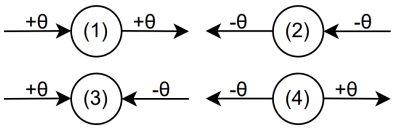
Successivamente procediamo spedendo le θ unità di flusso seguendo questa regola

$$x(\theta) \begin{cases} \overline{x}_{ij} + \theta & \text{se } (i,j) \in \mathcal{C}^+ \\ \overline{x}_{ij} - \theta & \text{se } (i,j) \in \mathcal{C}^- \\ \overline{x}_{ij} & \text{altrimenti} \end{cases} \tag{26}$$

Dobbiamo adesso capire per quali θ il flusso $x(\theta)$ è ammissibile; dobbiamo quindi vedere se **i bilanci sono rispettati**.

- Per i nodi esterni al ciclo non cambia nulla, di conseguenza le equazioni di bilancio sono rispettate.

- Nel caso dei nodi del ciclo dobbiamo analizzare 4 casi possibili



  Nel primo caso abbiamo θ unità di flusso che entrano e θ unità di flusso che escono, di conseguenza i bilanci sono rispettati. Nel secondo caso arrivano θ unità di flusso in meno da una parte, ma al contempo vengono richieste θ unità di flusso in meno dall'altra parte; di conseguenza l'equazione di bilancio è rispettata. Nel terzo e nel quarto caso arrivano θ unità di flusso in meno da una parte e θ unità di flusso in più dall'altra; di conseguenza l'equazione di bilancio è rispettata.

Inoltre, θ è ammissibile se e solo se

$$\theta \leq \min_{(i,j)\in\mathcal{C}^-}\{\overline{x}_{ij}\} \tag{27}$$

Quindo, l'arco uscente è quello tale per cui

$$\theta = \min_{(i,j)\in\mathcal{C}^-}\{\overline{x}_{ij}\} \tag{28}$$

Se $\theta = 0$ ci troviamo nel caso degenere, cioè, la partizione viene cambiata ma il flusso rimane uguale. Infine, possiamo dire che se non ci sono archi discordi, allora non ci sono limite sul flusso che può essere spedito lungo il ciclo, e quindi il valore della funzione obiettivo tende a $-\infty$.

2 Il problema del flusso minimo capacitato

All'inizio della nostra trattazione sui problemi di flusso minimo abbiamo fatto una supposizione importante; abbiamo supposto che il capacità superiore degli archi fosse $+\infty$. Questa condizione non è però sempre vera, può infatti esistere il caso in cui $u_{ij} < +\infty$ e quindi, dobbiamo anche rispettare il vincolo $x_{ij} \leq u_{ij} \ \ \forall (i,j) \in A$; possiamo riformulare il problema come

$$\begin{cases} \min c^T x \\ Ex = b \\ 0 \leq x \leq u \end{cases} \tag{29}$$

Il problema del flusso minimo non capacitato è un problema in forma duale standard; proviamo a portare anche il nostro problema capacitato nello stesso formato. Ricordiamo che possiamo, introducendo uno **slug**, trasformare i vincoli di maggioranza o di minoranza in vincoli di uguaglianza; possiamo quindi dire che

$$x \leq u \longleftrightarrow x + w = u \ \ w \geq 0 \tag{30}$$

Attraverso questa semplice regola possiamo procedere a riscrivere il nostro problema in una forma concorde a quella del duale standard

$$\begin{cases} \min c^T x \\ Ex = b \\ x + w = u \\ w \geq 0 \\ x \geq 0 \end{cases} \tag{31}$$

La forma matriciale la possiamo scrivere andando a modificare leggermente il nostro problema; scriviamo

$$\begin{cases} \min c^T x \\ Ex = b \\ Ix + Iw = u \\ w \geq 0 \\ x \geq 0 \end{cases} \tag{32}$$

La matrice del sistema può essere quindi riscritta sotto forma di matrice a blocchi

$$\left(\begin{array}{c|c} E & 0 \\ \hline I & I \end{array}\right)$$

Quindi, il problema può essere scritto in forma matriciale come

$$\begin{cases} \min c^T x \\ \left(\begin{array}{c|c} E & 0 \\ \hline I & I \end{array}\right) \begin{bmatrix} x \\ w \end{bmatrix} = \begin{bmatrix} b \\ u \end{bmatrix} \\ w \geq 0 \\ x \geq 0 \end{cases} \tag{33}$$

La matrice appena descritta ha dimensione pari a $(n+m)\times(2m)$; inoltre è una matrice **triangolare inferiore a blocchi** e di conseguenza ha determinante pari a ± 1 e rango pari a $(n+m-1)$. Definiamo adesso un importante teorema a riguardo della caratterizzazione delle basi

Theorem 2.1. (Caratterizzazione delle basi) Consideriamo una tripartizione (T, L, U) dell'insieme degli archi, ossia studiamo gli archi dividendoli in tre insiemi disgiunti T, L e U, dove T è un **albero di copertura**. Indichiamo con T', L', U' le righe di M corrispondenti alla partizione delle variabili di scarto w. All'ora l'insieme

$$B = T \cup U \cup T' \cup L' \tag{34}$$

è una base ed il determinate della corrispondente matrice è ± 1.

Per calcolare la soluzione di base relativa a B dobbiamo procedere considerando che, in quanto L e U' non sono in base, allora $\overline{x}_L = 0$ è $\overline{w}_{U'} = 0$. Possiamo quindi procedere come nell'algoritmo visto in precedenza, con una visita posticipata per foglie; con l'unica differenza che dobbiamo **necessariamente**, considerare come **saturo** l'arco $(i, j) \in U$.

Una volta determinata una soluzione di base dobbiamo verificare anche se è ammissibile; ci costruiamo il corrispettivo problema primale. Il primo passo è quello di trasporre la matrice a blocchi

$$\left(\begin{array}{c|c} E & 0 \\ \hline I & I \end{array}\right)^T = \left(\begin{array}{c|c} E^T & I \\ \hline 0 & I \end{array}\right) \tag{35}$$

Una volta trasposta la nostra matrice possiamo definire il nostro problema primale associato

$$\begin{cases} \max \quad (b, u)^T \begin{pmatrix} \pi \\ \mu \end{pmatrix} \\ \left(\begin{array}{c|c} E^T & I \\ \hline 0 & I \end{array}\right) \begin{pmatrix} \pi \\ \mu \end{pmatrix} \leq \begin{pmatrix} c \\ 0 \end{pmatrix} \end{cases} \tag{36}$$

che può, ulteriormente, essere scritto nella forma

$$\begin{cases} \max \quad b^T \pi + u^T \mu \\ E^T \pi + \mu \leq c \\ \mu \leq 0 \end{cases} \tag{37}$$

Possiamo calcolarci la soluzione di base del problema dei potenziali associata alla base $B = T \cup U \cup T' \cup L'$. Ricordiamo che, nel problema primale, una soluzione di base è ammissibile quando risolve i vincoli di base con l'uguale e i vincoli di non di base con il minore o uguale; quindi, le nostre soluzioni di base sono

$$\begin{aligned} & E_T^T \pi + \mu_T = c_T^T \\ & E_U^T \pi + \mu_U = c_U^T \\ & \mu_T = 0 \\ & \mu_L = 0 \end{aligned} \tag{38}$$

andando a sostituire la terza equazioni nella prima, ricaviamo che

$$E_T^T \pi = C_T \tag{39}$$

Inoltre, ponendo $\pi_1 = 0$ otteniamo il potenziale di base $\pi = E_T^{-1} c$; questo stesso potenziale può essere ottenuto sfruttando l'algoritmo già visto in precedenza. Inoltre, dalla seconda equazione otteniamo $\mu_U = C_U - E^T \pi$. Questo risultato è utilissimo, infatti, sfruttando un algoritmo semplicissimo, come quello del calcolo dei potenziali e facendo una semplice differenza, siamo in grado di trovare la soluzione di base del problema primale. Quindi la nostra soluzione di base è

$$\begin{cases} \overline{\pi} = E_T^{-1} c \\ \overline{\mu}_U = C_U - E_U^T \overline{\pi} \\ \mu_T = 0 \\ \mu_L = 0 \end{cases} \tag{40}$$

Inoltre, il potenziale di base è ammissibile se e solo se

$$\begin{cases} E_L^T \overline{\pi} + \overline{\mu}_L = E_L^T \overline{\pi} \leq C_L \\ \overline{\mu}_U = c_U - E_U^T \overline{\pi} \end{cases} \tag{41}$$

Riconducendoci alle condizioni di ottimalità di Bellman, possiamo anche analogamente definire delle condizioni di ammissibilità che coinvolgano i costi ridotti

Theorem 2.2. Sia (T, L, U) una tripartizione degli archi di un grafo (N, A) tale che:

- T sia un **albero di copertura**.
- Generi un flusso di base $\overline{x}$ ammissibile.

Supponiamo che $\overline{\pi}$ sia il **potenziale di base** associato alla tripartizione degli archi. Se il potenziale di base $\overline{\pi}$ è tale che

$$\begin{cases} C_{ij}^{\overline{\pi}} \geq 0 & \forall (i,j) \in L \\ C_{ij}^{\overline{\pi}} \leq 0 & \forall (i,j) \in U \end{cases} \tag{42}$$

Allora il flusso di base $\overline{x}$ è **ottimo**.

3 Algoritmo del simplesso per flussi

Nel paragrafo sul flusso minimo abbiamo studiato un'implementazione dell'algoritmo del simplesso; quello che però non abbiamo considerato è che quell'implementazione è riferita solamente al simplesso per flussi non capacitati. Nel caso di flussi capacitati dobbiamo estendere l'algoritmo studiato fino ad'ora

Simplesso per flussi

1. Trovare una partizione degli archi (T, L, U) che generi un flusso di base $\overline{x}$ **ammissibile**.
2. Calcoliamo il flusso di base sfruttando una visita posticipata per foglie o calcolando i seguenti valori
$$\overline{x}_T = E_T^{-1}(b - E_U u_U), \quad \overline{x}_L = 0, \quad \overline{x}_U = u_U \tag{43}$$
Calcoliamo anche il potenziale base sfruttando una visita anticipata per foglie o, in alternativa, con il prodotto
$$\overline{\pi} = E_T^{-1} c \tag{44}$$
Calcoliamo infine i costi ridotti $C_{ij}^{\overline{\pi}} = c_{ij} + \overline{\pi}_i - \overline{\pi}_j$ per ogni arco $(i,j) \in L \cup U$.
3. Se le condizioni di Bellman sono rispettate allora ci possiamo fermare; abbiamo raggiunto la soluzione ottima. Altrimenti si calcola l'arco entrante, definito come
$$(p,q) = \min\left(\{(i,j) \in L : C_{ij}^{\overline{\pi}} < 0\} \cup \{(i,j) \in U : C_{ij}^{\overline{\pi}} > 0\} \right) \tag{45}$$

Una volta trovato l'arco entrante si viene a formare un ciclo $\mathcal{C}$. Si fissa quindi un verso, concorde al verso di (p,q) se $(p,q) \in L$, altrimenti, se $(p,q) \in U$, discorde al verso di (p,q) e si divide l'insieme degli archi di $\mathcal{C}$ in due partizioni

- $\mathcal{C}^+$: l'insieme degli archi **concordi** al verso fissato.
- $\mathcal{C}^-$: l'insieme degli archi **discordi** al verso di fissato.

4. Si calcola

$$\begin{aligned} \theta^+ &= \min\{u_{ij} - \overline{x}_{ij} : (i,j) \in \mathcal{C}^-\} \\ \theta^- &= \min\{\overline{x}_{ij} : (i,j) \in \mathcal{C}^-\} \\ \theta &= \min\{\theta^+, \theta^-\} \end{aligned} \tag{46}$$

Se $\theta = +\infty$ ci fermiamo, il flusso di costo ottimo ha valore $-\infty$. Altrimenti calcoliamo l'arco uscente

$$(r,s) = \min\Big(\{(i,j) \in \mathcal{C}^+ : u_{ij} - \overline{x}_{ij} = \theta\} \cup \{(i,j) \in \mathcal{C}^- : \overline{x}_{ij} = \theta\}\Big) \tag{47}$$

5. Si aggiorna la tripartizione seguendo queste regole

- Se $(p,q) \in L$ allora
  - Se $\theta = \theta^-$ abbiamo che un arco di $\mathcal{C}^-$ si è svuotato; questo arco è l'arco uscente (r,s) e viene posto in L.
  - Se $\theta = \theta^+$ abbiamo che un arco di $\mathcal{C}^+$ si è saturato; questo arco è l'arco uscente (r,s) e viene posto in U. In questo caso potrebbe verificarsi la situazione nella quale l'arco che entra è anche l'arco che esce, ma non è una cosa di cui ci preoccupiamo; infatti l'arco arriva da L e finisce in U, di conseguenza abbiamo comunque cambiato la base.
- Se $(p,q) \in U$ allora
  - Se $\theta = \theta^+$ abbiamo che un arco di $\mathcal{C}^+$ si è saturato; questo arco è l'arco uscente (r,s) e viene posto in U.
  - Se $\theta = \theta^-$ abbiamo che un arco di $\mathcal{C}^-$ si è svuotato; questo arco è l'arco uscente (r,s) e viene posto in L.

L'algoritmo del simplesso per flussi ha un funzionamento molto simile a quello dell'algoritmo del simplesso per flussi non capacitato. Si parte sempre da un flusso di base ammissibile, si calcolano i potenziali e si verificano le condizioni di Bellman; se sono soddisfatte l'algoritmo termina, siamo alla soluzione ottima. Altrimenti dobbiamo calcolare quale sia l'arco che viola **Bellman** e che, essendo in un simplesso duale, è anche l'arco entrante

$$(p,q) = \min\Big(\{(i,j) \in L : C_{ij}^{\overline{\pi}} < 0\} \cup \{(i,j) \in U : C_{ij}^{\overline{\pi}} > 0\}\Big) \tag{48}$$

Per semplificare la trattazione analizziamo i due casi in cui ci possiamo trovare. Se andiamo ad aggiungere l'arco entrante al nostro albero otteniamo un ciclo $\mathcal{C}$; consideriamo un flusso ottenuto da $\overline{x}$ spdento flusso sul ciclo $\mathcal{C}$. Se l'arco $(p,q) \in L$, cioè $\overline{x}_{pq} = 0$, il flusso può solo aumentare, quindi fissiamo su $\mathcal{C}$ un verso che sia concorde con (p,q). D'altra parte, se $(p,q) \in U$ l'arco è saturo, dobbiamo quindi fissare un verso che sia discorde all'arco (p,q). Partizioniamo poi il ciclo $\mathcal{C}$ in due sotto-insiemi:

- $\mathcal{C}^+$: l'insieme degli archi **concordi** al verso fissato.
- $\mathcal{C}^-$: l'insieme degli archi **discordi** al verso di fissato.

Quello che dobbiamo fare è spedire θ unità di flusso lungo gli archi che appartengono a $\mathcal{C}^+$ e spedire λ unità di flusso in meno lungo gli archi che appartengono a $\mathcal{C}^-$:

$$x(\lambda) = \begin{cases} \overline{x}_{ij} + \lambda & \text{se } (i,j) \in \mathcal{C}^+ \\ \overline{x}_{ij} - \lambda & \text{se } (i,j) \in \mathcal{C}^- \\ \overline{x}_{ij} & \text{altrimenti} \end{cases} \tag{49}$$

Inoltre, la funzione obiettivo in funzione di λ subisce una diminuzione, nello specifico

$$c^T x(\lambda) = \begin{cases} c^T x + \lambda C_{ij}^{\pi} & \text{se } (p,q) \in L \\ c^T x - \lambda C_{ij}^{\pi} & \text{se } (p,q) \in U \end{cases} \iff c^T x(\lambda) < c^T \overline{x} \quad \forall \lambda > 0 \tag{50}$$

Quindi, spedire θ unità di flusso lungo il ciclo ci fa diminuire la funzione obiettivo. Quello che occorre verificare è che il flusso $x(\lambda)$ rispetta i bilanci ai nodi; affinché il flusso sia ammissibile occorre che

$$0 \leq x(\lambda) \leq u \tag{51}$$

dove u rappresenta la capacità superiore degli archi dell'albero. Questa stessa condizione può essere riformulata in un altro modo; supponiamo innanzitutto di avere tutti gli archi di $\mathcal{C}^+$, su questi archi viene spedita una certa quantità di flusso aggiuntiva. Su ognuno di questi archi è già presente una quantità di flusso $\overline{x}_{ij}$, dobbiamo quindi verificare che il flusso ottenuto considerando anche λ non ecceda la capacità superiore di quel ramo

$$\overline{x}_{ij} + \lambda \leq u_{ij} \iff \lambda <= u_{ij} - \overline{x}_{ij} \tag{52}$$

Di tutto l'insieme dei valori calcolati come differenza tra la capacità superiore del ramo e il flusso presente in quel momento sul ramo, devo considerare il minimo; definisco quindi come θ^+ il minimo dell'insieme dei "*flussi disponibili*" sui diversi archi

$$\theta^+ = \min\{u_{ij} - \overline{x}_{ij} : (i,j) \in \mathcal{C}^+\} \tag{53}$$

Se un arco appartiene a $\mathcal{C}^-$, il valore di λ fa decrescere la quantità di flusso presente su quell'arco, ma siccome affinché il flusso sia ammissibile, deve essere anche positivo, non posso prendere un λ tale da rendere negativo un flusso; di conseguenza devo prendere λ che sia minore e uguale del minore tra i flussi. Definisco il coefficiente θ^-

$$\theta^- = \min\{\overline{x}_{ij} : (i,j) \in {}^-\} \tag{54}$$

Quindi, abbiamo che λ deve rispettare le seguenti condizioni

$$\begin{cases} 0 \leq \lambda \leq \theta^+ \\ 0 \leq \lambda \leq \theta^- \end{cases} \tag{55}$$

che può essere espressa come

$$0 \leq \lambda \leq \theta = \min\{\theta^+, \theta^-\} \tag{56}$$

Per ottenere un flusso di base si sceglie $\lambda = \theta$. Se $\theta = \theta^+$ un arco $(r,s) \in T$ o lo stesso arco $(p,q) \in L$ si satura e viene messo in U, mentre se $\theta = \theta^-$ un arco $(r,s) \in T$ o lo stesso arco $(p,q) \in U$ si svuotano e di conseguenza va messo in L. Infine, se $\theta^+ = \theta^-$ si sceglie, utilizzando la regola anticiclo di Bland, quello minore in ordine lessicografico.

Capitolo 13

Cammini minimi

Consideriamo un grafo (N, A) in cui ad ogni arco (i, j) sia associato un costo c_{ij}. Il costo di un cammino orientato è definito come la somma dei costi degli archi da cui esso è formato. Il **problema dei cammini minimi di radice r** consiste nel determinare, se esiste, un cammino orientato di costo minimo dal nodo r verso ogni altro nodo del grafo. Questo problema può essere formulato come un problema di flusso di costo minimo, infatti possiamo immaginarci che il nodo radice abbia bilancio pari a $-(n-1)$ e che tutti i nodi non radice abbiano bilancio pari a 1.

$$\begin{cases} \min c^T x \\ Ex = b \\ x_{ij} \geq 0 \quad \forall (i,j) \in A \end{cases} \qquad \text{dove} \quad b_i = \begin{cases} -(n-1) & \text{se } i = r \\ 1 & \text{se } i \neq r \end{cases} \tag{1}$$

Quindi, stiamo supponendo che $n-1$ unità di flusso si debbano dirigere dal nodo r ognuna verso un nodo $i \neq r$ minimizzando il costo complessivo di $c^T x$. In questo caso non abbiamo capacità superiori sugli archi, per cui B è una base se e solo se rappresenta un albero di copertura del grafo. Inoltre, B è una base ammissibile se e solo se rappresenta una un albero di copertura del grafo di radice r, ossia, contiene un cammino orientato da r verso ogni altro nodo $i \neq r$.

Una particolarità dei problemi di cammini minimi è l'impossibilità di esistenza dei **casi degeneri**; ogni nodo del cammino ha solo un arco entrante, di conseguenza a meno che non sia una radice, dovranno arrivargli unità di flusso sufficienti da distribuire a tutti i nodi in cui entra, di conseguenza è impossibile che ci sia un arco con flusso nullo. Inoltre, supponendo di avere un albero di copertura (ammissibile) T_r, il vettore π dei potenziali di base associati a T_r è tale che

$$\pi_j - \pi_i = c_{ij} \quad \forall (i,j) \in T_r \tag{2}$$

per cui sappiamo che π_i rappresenta il costo del cammino orientato da r a i.

Infine, essendo il problema riconducibile a un problema di flusso di costo minimo, è sufficiente che siano rispettate le condizioni di Bellman per l'ottimalità

Theorem 0.1. (Condizioni di ottimalità) Sia T_r un albero di copertura orientato di radice r e sia, inoltre, π il vettore dei potenziali associati a T_r. Allora T_r è un albero dei cammini minimi se e solo se T_r verifica le condizioni di Bellman

$$C_{ij}^{\pi} = c_{ij} + \pi_i - \pi_j \geq 0 \tag{3}$$

1 Algoritmo del simplesso per cammini minimi

Quello che abbiamo visto fino ad adesso ci consente di stabilire l'algoritmo del simplesso per cammini, che nella pratica non è altro che un simplesso per flussi con delle semplificazioni

- Le **basi ammissibili** corrispondono ad alberi orientati di radice r.

- Non sono necessarie regole di anticiclo perché non ci sono basi degeneri.
- Se (p, q) è l'arco entrante in base, l'arco uscente sarà, per forza, l'unico arco che entra nel nodo q; d'altronde, tra gli archi discordi al verso del ciclo formato, (i, q) è quello di flusso minimo.

L'algoritmo del simplesso per il problema dei cammini minimi è schematizzato come segue

Simplesso per cammini minimi

1. Trovare un albero T orientato di radice r.
2. Calcolare il potenziale di base $\overline{\pi}^T$ e i costi ridotti $C_{ij}^{\overline{\pi}^T}$ per ogni arco non in base.
3. Se tutti i costi ridotti sono maggiori di 0 ci fermiamo, siamo all'ottimo. Altrimenti scelgo come arco entrante un arco $C_{ij}^{\overline{\pi}^T} < 0$. L'arco (p, q) forma un ciclo $\mathcal{C}$ con gli archi di T.
4. Se tutti gli archi di $\mathcal{C}$ sono concordi con (p, q) ci fermiamo, non esiste l'albero dei camini minimi. Altrimenti scelgo come arco uscente l'unico arco entrante sul nodo q.
5. Aggiorno l'albero.

2 Algoritmo di Dijkstra

Il problema dei cammini minimi, vista la sua importanza, è stato oggetto di moltissimi studi specifici. Tanti algoritmi sono stati inventati, tra cui, il più famoso è **l'algoritmo di Dijkstra**; questo algoritmo è in grado di trovare un albero dei cammini minimi in un grafo in cui i costi degli archi sono non negativi.

Passiamo ad enunciare e descriverlo. Ad ogni iterazione l'algoritmo mantiene un albero di copertura orientato di radice r ed un vettore π di etichette ai nodi, con la proprietà che π_i rappresenta il costo del cammino da r a i contenuto nell'albero, ossia, π è il potenziale di base corrispondente all'albero. All'inizio, l'albero è formato da archi fittizzi (r, i) per ogni $i \neq r$, che collegano la radice a tutti i nodi con costo $c_{ri} = +\infty$; inoltre, il vettore dei predecessori è così definito

$$p_i = \begin{cases} -1 & i \neq r \\ 0 & i = r \end{cases} \tag{4}$$

Mentre le corrispondenti etichette sono

$$\pi_i = \begin{cases} +\infty & i \neq r \\ 0 & i = r \end{cases} \tag{5}$$

Ad ogni iterazione si cercano tutti gli archi che violano la condizione di Belmman e si cambia l'albero e le corrispondenti etichette in modo da soddisfare tali condizioni. L'algoritmo mantiene anche un insieme U di nodi in cui alcuni tra gli archi uscenti dai nodi possono violare le condizioni di Bellman. Inizialmente U contiene tutti i nodi del grafo. Ad ogni iterazione si estrae da U un nodo u con etichetta minima e si controllano gli archi uscenti da u. Se un arco (u, v) viola le condizioni di Bellman significa che il cammino da r a v contenuto nell'albero ha costo maggiore del cammino ottenuto unendo il cammino da r a u e l'arco (u, v). A questo punto l'albero viene modificato assegnando il nodo u come predecessore del nodo v e modificando l'etichetta del nodo v in mood corrispondente

$$\pi_v = \pi_u + c_{uv} \tag{6}$$

L'algoritmo termina quando $U = \emptyset$.

Algoritmo di Dijkstra

Poni

$$U := N, \quad p_i = \begin{cases} -1 & i \neq r \\ 0 & i = r \end{cases}, \quad \pi_i = \begin{cases} +\infty & i \neq r \\ 0 & i = r \end{cases} \tag{7}$$

Fintanto che $U \neq \emptyset$:

1. Estrai da U un nodo u con etichetta π_u minima.
2. Per ogni arco (u, v) uscente da u valutiamo se $\pi_v > \pi_u + c_{uv}$, quindi, valutiamo se il percorso da r a v che abbiamo in questo momento è maggiore del percorso da r a u più il costo per andare da u a v. Se il risultato è affermativo impostiamo come predecessore di v il nodo u $p_v = u$ e cambiamo il costo per arrivare a v $\pi_v = \pi_u + c_{uv}$

Se tutti i costi sono positivi allora l'algoritmo di Dijkstra produce un albero dei camminimi minimi un un numero finito di iterazioni. Un'importante osservazione da fare è la seguente

Se nel grafo esistono archi di costo negativo, non è detto che l'algoritmo sia in grado di trovare l'albero dei cammini minimi.

Capitolo 14

Flusso massimo

Si supponga di avere un grafo (N, A) in cui, per ogni arco $(i, j) \in A$ siano assegnate delle capacità superiori u_{ij}. Il problema del flusso massimo consiste nello spedire la quantità di flusso maggiore da un nodo s a un nodo t rispettando i vincoli di capacità sugli archi. Tale problema può essere formulato come

$$\begin{cases} \max v \\ Ex = b \\ 0 \leq x \leq u \end{cases} \tag{1}$$

Possiamo supporre che i bilanci siano così costruiti

$$b_i = \begin{cases} -v & i = s \\ v & i = t \\ 0 & \text{altrimenti} \end{cases} \tag{2}$$

dove v rappresenta la **quantità di flusso massima** uscente da s e entrante in v.

1 Simplesso per flussi massimi

Il problema del flusso massimo può essere risolto mediante algoritmo del simplesso per flussi. Il primo passo è quello di trasformarlo in problema di costo minimo, lo possiamo fare introducendo un arco (t, s) di costo -1 e di capacità $+\infty$, assegnando costo nullo a tutti gli altri archi e ponendo bilanci nulli a tutti i nodi. Il problema diventa

$$\begin{cases} \min -v \\ \begin{pmatrix} E & \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ -1 \end{pmatrix} \end{pmatrix} \begin{pmatrix} x \\ v \end{pmatrix} = 0, \\ 0 \leq x \leq u, \\ v \geq 0. \end{cases} \tag{3}$$

Procediamo poi applicando il simplesso per flussi. Cerchiamo di capirlo meglio con un esempio pratico: sia dato il seguente grafo

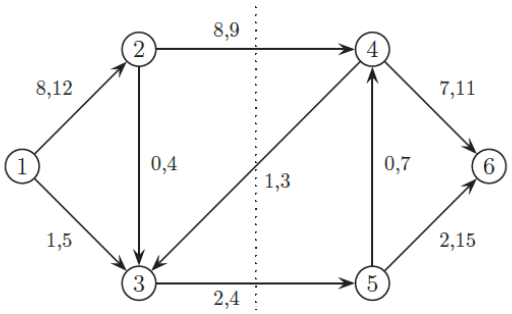
2 Algoritmo di Ford-Fulkerson

Prima di procedere con la descrizione dell'algoritmo di Ford-Fulkerson è necessario definire quali sono le condizioni di ottimalità per il problema di flusso massimo. Il primo concetto che andiamo ad introdurre è quello di taglio di un grafo

Definition 2.1. (Taglio di un grafo) Un taglio (N_s, N_t) è una partizione dell'insieme N dei nodi in due sotto-insiemi

$$N = N_s \cup N_t, \quad N_s \cap N_t = \emptyset \tag{4}$$

Un taglio (N_s, N_t) si dice ammissibile solo se N_s contiene almeno l'origine s e N_t contiene almeno la destinazione t. Prendiamo un esempio



Partizionando il grafo in figura otteniamo $N_s = \{1, 2, 3\}$ e $N_t = \{4, 5, 6\}$, questo taglio è **ammissibile**, in quanto la partizione N_s contiene il **nodo sorgente** e N_t contiene il **nodo destinazione**. Gli archi del taglio sono così definiti

$$\begin{aligned} A^+ &= \{(i,j) \in A : i \in N_s, j \in N_t\} \quad \text{Archi diretti} \\ A^- &= \{(i,j) \in A : i \in N_t, j \in N_s\} \quad \text{Archi inversi} \end{aligned} \tag{5}$$

Quindi, nel primo insieme abbiamo tutti gli archi che vanno dalla *partizione sorgente* a quella *di destinazione* (**diretti**); nel nostro esempio sono gli archi $A^+ = \{(2,4), (3,5)\}$. Nel secondo insieme sono inclusi tutti gli archi che vanno dalla partizione **di destinazione** verso quella **sorgente** (**inversi**) $A^- = \{(4,3)\}$. Definiamo anche una **capacità del taglio**, definita come la capacità degli archi diretti del taglio

$$u(N_s, N_t) = \sum_{(i,j) \in A^+} u_{ij} \tag{6}$$

che nel nostro esempio equivale a 13. Definiamo anche il **flusso del taglio** come differenza tra la somma dei flussi degli archi diretti e somma dei flussi sugli archi inversi

$$x(N_s, N_t) = \sum_{(i,j) \in A^+} x_{ij} - \sum_{(i,j) \in A^-} x_{ij} \tag{7}$$

che nel nostro esempio equivale a 9. Quindi, il taglio ammissibile indicato in figura è costituito da

$$\begin{aligned} &A^+ = \{(2,4), (3,5)\} \quad A^- = \{(4,3)\} \\ &u(N_s, N_t) = 13 \quad x(N_s, N_t) = 9 \end{aligned} \tag{8}$$

Definiamo un'importante relazione che lega flusso del taglio e capacità del taglio

Lemma 2.1. Se x è un flusso ammissibile e (N_s, N_t) è un *taglio ammissibile*, allora

$$v = x(N_s, N_t) \leq u(N_s, N_t) \tag{9}$$

Da questa conclusione possiamo arrivare a un teorema fondamentale

Theorem 2.2. (Teorema del "flusso massimo-taglio minimo") Se esistono un flusso x ammissibile e un taglio (N_s, N_t) ammissibile tali per cui

$$x(N_s, N_t) = u(N_s, N_t) \tag{10}$$

allora x è un **flusso massimo** e (N_s, N_t) è un **taglio minimo**.

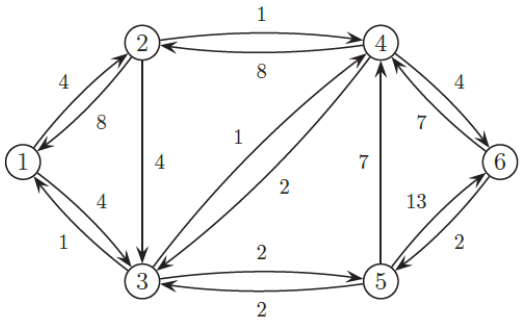
Su quest'ultimo teorema si basa l'algoritmo di Ford-Flukerson. Prima di enunciare l'algoritmo però procediamo dando due definizioni fondamentali

Definition 2.2. (Grafo residuo) Dato un flusso x ammissibile, il grafo residuo relativo al flusso x è un grafo $G(x) = (N, A(x))$ con gli stessi nodi del grafo originale G, mentre gli archi e le loro capacità r_{ij}, dette residue, sono così definiti:

$$\begin{aligned} (i,j) \in A,\ x_{ij} < u_{ij} \quad &\Longrightarrow\ (i,j) \in A(x),\ r_{ij} = u_{ij} - x_{ij}, \\ (i,j) \in A,\ x_{ij} > 0 \quad &\Longrightarrow\ (j,i) \in A(x),\ r_{ij} = x_{ij}. \end{aligned}$$

In altre parole, qualora su un ramo $(i,j) \in A$ è presente un flusso positivo, si crea un arco (j,i) con flusso uguale a quello dell'arco (i,j) e sostituiamo il flusso sull'arco x_{ij} con il residuo r_{ij}.

Prendiamo come esempio il grafo iniziale, il corrispettivo grafo dei residui è così costituito



Un altro concetto su cui si basa l'algoritmo di Ford-Fulkerson è quello di **cammino aumentato**

Definition 2.3. Dato un flusso ammissibile x e un grafo residuo $G(x)$, definiamo **cammino aumentante** rispetto a x un cammino **orientato** da s a t nel grafo $G(x)$

Un cammino aumentante potrebbe essere un cammino non orientato nel grafo di partenza. L'idea alla base dell'algoritmo di Ford-Fulkerson è cercare, ad ogni iterazione, un cammino aumentante rispetto al flusso corrente, allora il flusso viene aggiornato spedendo su tale cammino il flusso possibile pari alla minima capacità residua degli archi che formano il cammino. Se non esistono cammini aumentati, allora il flusso corrente è di valore massimo e l'algoritmo non trova nessun taglio di capacità minima.

Algoritmo di Ford-Fulkerson

Poni $x := 0$, $G(x) := G$. Se esiste un cammino aumentante $\mathcal{C}$ allora

1. Calcola $\delta = \min\{r_{ij} : (i,j) \in \mathcal{C}\}$
2. Per ogni arco $(i,j) \in A$:
   - Se $(i,j) \in \mathcal{C}$ allora $x_{ij} = x_{ij} + \delta$.
   - Se $(j,i) \in \mathcal{C}$ allora $x_{ij} = x_{ij} - \delta$

Per trovare il cammino aumentante è possibile applicare un altro algoritmo, chiamato algoritmo di Edmonds-Karp, in cui si cerca tra tutti i possibili cammini aumentanti e si sceglie quello col minimo numero di archi. L'idea alla base di questo algoritmo è che se nel grafo residuo $G(x)$ esiste almeno un cammino orientato da s a t, tale algoritmo trova in $G(x)$ un cammino da s a t con il numero minimo di archi, altrimenti trova un taglio di capacità minimo. L'algoritmo è una visita a ventaglio del grafo residuo a partire dall'origine s e fa uso dell'insieme Q dei nodi raggiunti nell'esplorazione del grafo e del vettore p dei predecessori. L'insieme Q di comporta come una struttura a *pila*, quindi con una politica del tipo *first-in first-out*, quindi si estraggono i nodi dalla cima della struttura e si inseriscono sul fondo di essa. Inizialmente l'insieme Q contiene solo il nodo sorgente; ad ogni iterazione si estrae il nodo i-esimo da Q e si controllano tutti gli archi uscenti da esso in ordine lessicografico e si inseriscono in coda a Q. L'algoritmo termina in due casi

- Quando il nodo t entra in Q.
- Quando Q si svuota.

Nel secondo caso dobbiamo osservare il vettore dei predecessori; infatti da esso otteniamo il taglio di capacità minima

$$N_s = \{i \in N : p_i \neq -1\} \quad N_t = \{i \in N : p_i = -1\} \tag{11}$$

L'algoritmo è schematizzabile nel seguente modo

Algoritmo di Edmonds-Karp

1. Porre
$$p_i = \begin{cases} -1 & i \neq s \\ 0 & i = s \end{cases}, \quad Q = \{s\} \tag{12}$$
2. Se $Q = \emptyset$ allora ci fermiamo (p fornisce un taglio di capacità minima). Altrimenti estraiamo il primo elemento di Q (a cui ci rirferiremo come i).
3. Se $(i, t) \in A(x)$ allora $p_t = i$ e ci fermiamo, abbiamo trovato un cammino aumentante.
4. In $G(x)$ si analizzano gli archi uscenti da i in ordine lessicografico. Per ogni $(i, j) \in A(x)$
   - Se $p_j = -1$ allora $p_j = i$ e aggiungi j in fondo.
   - Altrimenti non facciamo nulla
5. Tornare al passo 2

Capitolo 15

Problema del trasporto

Supponiamo di avere **m** luoghi di **produzione** e **n** luoghi di **raccolta**. Supponiamo che siano note le capacità produttive o_i di ogni stabilimento (*settimanali*, *mensili* o *giornaliere*), la **domanda** d_j di ogni luogo di raccolta ed il costo di trasporto (**unitario**) c_{ij} da ogni luogo di produzione *i-esimo* a ogni luogo di raccolta *j-esimo*. Definiamo la variabile x_{ij} come la quantità di merce trasportata dal luogo di produzione *i-esimo* verso il luogo di raccolta *j-esimo*. Teniamo inoltre conto che

- Se fissiamo j e sommiamo tutte le x_{ij} otteniamo la quantità totale di merce che arriva al luogo di raccolta *j-esimo*.
- Se fissiamo i e sommiamo tutte le x_{ij} otteniamo la quantità totale di merce spedita dal luogo id produzione *i-esimo*.

Per definire il modello matematico andiamo prima di tutto a determinare la funzione obiettivo; nel nostro caso il problema è quello di *minimizzare il costo totale del trasporto, compatibilmente alla domanda e all'offerta dei rispettivi punti di raccolta e di produzione*. La nostra funzione obiettivo sarà sicuramente di minimo e equivarrà alla somma delle quantità di merce prodotta dal luogo *i-esimo* e spedita al luogo di raccolta *j-esimo* moltiplicato per il costo unitario di spedizione

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{1}$$

Il primo vincolo che dobbiamo imporre è che la quantità di merce spedita al luogo di raccolta *j-esimo* sia maggiore o equivalente alla domanda del luogo stesso

$$\sum_{i=1}^{m} x_{ij} \geq d_j \quad \forall j = 1, \ldots, n \tag{2}$$

Il terzo vincolo invece impone che la quantità di merce spedita da un luogo di produzione *i-esima* non superi la capacità produttiva

$$\sum_{j=1}^{m} x_{ij} \leq o_i \quad \forall i = 1, \ldots, m \tag{3}$$

Ovviamente, dobbiamo anche assicurarci che la quantità di merce spedita sia positiva (la mancanza di questo vincolo potrebbe portare a soluzioni assurde)

$$x_{ij} \geq 0 \quad \forall i = 1, \ldots, m, \forall j = 1, \ldots, n \tag{4}$$

Il modello matematico è così definito

$$\begin{cases} \min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \\ \sum_{i=1}^{m} x_{ij} \geq d_j \quad \forall j = 1, \ldots, n \\ \sum_{j=1}^{m} x_{ij} \leq o_i \quad \forall i = 1, \ldots, m \\ x_{ij} \geq 0 \quad \forall i = 1, \ldots, m, \forall j = 1, \ldots, n \end{cases} \tag{5}$$

Naturalmente, il problema non ha soluzione se la domanda totale supera l'offerta totale

$$\sum_{j=1}^{n} d_j \leq \sum_{i=1}^{m} o_i \tag{6}$$

D'altra parte, se abbiamo un eccesso di produzione possiamo introdurre una località di raccolta fittizia a cui spedire l'eccesso, considerando come costo di spedizione 0.

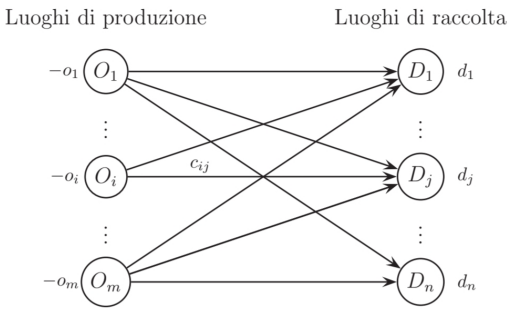
Concludiamo la nostra analisi con una piccola nota; se la domanda non supera l'offerta, a meno di aggiungere un nodo fittizio di raccolta, possiamo supporre che tutti i vincoli del modello precedente siano di uguaglianza

$$\begin{cases} \min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \\ \sum_{i=1}^{m} x_{ij} = d_j \quad \forall j = 1, \ldots, n \\ \sum_{j=1}^{m} x_{ij} = o_i \quad \forall i = 1, \ldots, m \\ x_{ij} \geq 0 \quad \forall i = 1, \ldots, m, \forall j = 1, \ldots, n \end{cases} \tag{7}$$

Il nostro modello è anche compatibile con il modello di trasporto di merce divisibile analizzato nel paragrafo della PL e anche con il modello del trasporto non divisibile della PLI (a patto di aggiungere il vincolo $x_{ij} \in \mathbb{Z}_+$). Osserviamo anche che il nostro problema può essere formulato come un problema di flusso di costo minimo, infatti possiamo supporre che i luoghi di produzione e i luoghi di raccolta siano i nodi del grafo; nello specifico

- I luoghi di produzione $O_1, \ldots, O_m$ sono nodi con bilancio $-o_1, \ldots, -o_m$.
- I luoghi di raccolta $D_1, \ldots D_n$ sono nodi con bilancio $d_1, \ldots, d_n$.

Supponiamo che gli archi che collegano ogni luogo di produzione i con ogni luogo di raccolta j abbiano costo per unità di flusso pari a C_{ij} e capacità superiore pari a $+\infty$. Esistono anche delle



varianti del problema, che potrebbero condurci a modelli matematici diversi:

- Domanda e disponibilità potrebbero non essere note a priori.
- il costo potrebbe non essere lineare ma costante a tratti oppure con un costo fisso iniziale di trasporto.
- Le connessioni con i luoghi di raccolta potrebbero non essere tutte presenti.

Capitolo 16

Problema dell'assegnamento di costo minimo

Supponiamo di dover eseguire n lavori avendo a disposizione n lavoratori, ciascuno dei quali in grado di fare ogni lavoro. Definiamo la **matrice dei costi**, dove ogni elemento c_{ij} rappresenta il costo per far svolgere il lavoro *j-esimo* al lavoratore *i-esimo*.

$$\begin{bmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,n} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n,1} & c_{n,2} & \cdots & c_{n,n} \end{bmatrix} \tag{1}$$

Se ci fosse un lavoratore *i-esimo* che non è in grado di svolgere un lavoro *j-esimo*, sarebbe sufficiente supporre che $c_{ij} \to +\infty$ (nella pratica si traduce nell'impostare un costo estremamente alto, in modo che siamo sicuri che non venga scelto). Definiamo adesso la variabile x_{ij}, la quale ci permette di specificare se facciamo svolgere il lavoro *j-esimo* al lavoratore *i-esimo*

$$x_{ij} = \begin{cases} 1 & \text{il lavoro } j\text{-esimo viene fatto dal lavoratore } i\text{-esimo} \\ 0 & \text{altrimenti} \end{cases} \tag{2}$$

La funzione obiettivo che dobbiamo minimizzare è la seguente

$$\min \sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij} x_{ij} \tag{3}$$

I vincoli che dobbiamo introdurre nel nostro problema sono due, il primo vincolo specifica che un singolo lavoro può essere fatto da un singolo lavoratore

$$\sum_{i=0}^{n} x_{ij} = 1 \quad \forall j = 1, \ldots, n \tag{4}$$

Il secondo vincolo specifica invece che ad un singolo lavoratore non possa essere far fatto fare più di un lavoro

$$\sum_{j=0}^{n} x_{ij} = 1 \quad \forall i = 1, \ldots, n \tag{5}$$

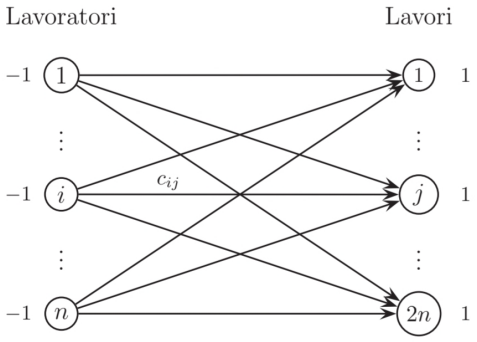
Inoltre, dobbiamo assicurarci che x sia corretto introducendo il vincolo di interezza del problema $x \in \{0,1\}$. Il modello è

$$\begin{cases} \min \sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij} x_{ij} \\ \sum_{i=0}^{n} x_{ij} = 1 \quad \forall j = 1, \dots, n \\ \sum_{j=0}^{n} x_{ij} = 1 \quad \forall i = 1, \dots, n \\ x \in \{0,1\} \end{cases} \tag{6}$$

Però, questo problema ha una caratteristica interessante, rientra nella classe dei problemi PL = PLI, quindi, anche rimuovendo il vincolo di interezza, si ottiene comunque una soluzione ottima a componenti intere. Possiamo quindi anche rimuovere il vincolo e riscrivere il problema come

$$\begin{cases} \min \sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij} x_{ij} \\ \sum_{i=0}^{n} x_{ij} = 1 \quad \forall j = 1, \dots, n \\ \sum_{j=0}^{n} x_{ij} = 1 \quad \forall i = 1, \dots, n \end{cases} \tag{7}$$

Esistono due modi di risolvere il problema. Il primo metodo consiste nel risolverlo utilizzando **linprog**. Il secondo metodo è quello di risolverlo come problema di **flusso minimo non capacitato**. Se intraprendiamo questa strada abbiamo n^2 archi che connettono ogni lavoratore ad ogni lavoro con costo c_{ij}. A questo punto sarà sufficiente applicare il simplesso per flussi non capacitati



e risolvere il problema. Il problema dell'assegnamento ha anche un risvolto molto interessante nel **TSP**. Possiamo utilizzare la soluzione ottima del problema di assegnamento di costo minimo come **valutazione inferiore** per il TSP, sia **simmetrico** che **asimmetrico**. Prendiamo il modello del **TSP asimmetrico**

$$\begin{cases} \min \sum_{(i,j) \in A} c_{ij} x_{ij} \\ \sum_{i \in N, i \neq j} x_{ij} = 1 \ \ \forall j \in N \\ \sum_{j \in N, i \neq j} x_{ij} = 1 \ \ \forall i \in N \\ \sum_{i \in S} \sum_{j \notin S} x_{ij} \geq 1 \ \ \forall S \subset N, S \neq \emptyset \\ x_{ij} \in \{0,1\} \ \ \forall (i,j) \in A \end{cases} \tag{8}$$

Se andassimo a rimuovere il terzo vincolo otterremo esattamente il modello di assegnamento di costo minimo definito all'inizio del paragrafo

$$\begin{cases} \min \sum_{(i,j)\in A} c_{ij}x_{ij} \\ \sum_{i\in N, i\neq j} x_{ij} = 1 \ \ \forall j \in N \\ \sum_{j\in N, i\neq j} x_{ij} = 1 \ \ \forall i \in N \\ \forall (i,j) \in A \end{cases} \equiv \begin{cases} \min \sum_{i=0}^{n}\sum_{j=0}^{n} c_{ij}x_{ij} \\ \sum_{i=0}^{n} x_{ij} = 1 \quad \forall j = 1,\dots,n \\ \sum_{j=0}^{n} x_{ij} = 1 \quad \forall i = 1,\dots,n \end{cases} \tag{9}$$

Parte IV

Programmazione non lineare

Capitolo 17

Teoria della Programmazione non lineare

La programmazione non lineare è una branca dell'ottimizzazione che studia tutti quei problemi scritti nella forma

$$\begin{cases} \min f(x) \\ g(x) \leq 0 \\ h(x) = 0 \end{cases} \tag{1}$$

dove non tutte le funzioni f, g_i con $i = 1, \ldots, m$ e h_i con $i = 1, \ldots, n$ sono **lineari**. Per semplicità supporremmo che tali funzioni siano $\mathcal{C}^\infty$ su un insieme aperto che contiene la regione ammissibile

$$\Omega = \{x \in \mathbb{R}^n : g(x) \leq 0, \quad h(x) = 0\} \tag{2}$$

La ricerca dei nostri punti di minimo procede cercando di trovare delle proprietà che caratterizzino tali punti. I problemi della *Programmazione Non Lineare* si possono dividere in due tipologie, **problemi non vincolati**, quindi, problemi in cui la funzione obiettivo è definita come

$$f : \mathbb{R}^n \to \mathbb{R} \tag{3}$$

e **problemi vincolati**, quindi, problemi in cui la funzione obiettivo è nella forma

$$f : D \to \mathbb{R}, \quad D \subset \mathbb{R}^n \tag{4}$$

Il problema che affronteremo noi durante il corso è quello di riuscire a trovare i punti di minimo delle funzioni; nell'appendice che si trova sotto abbiamo richiamato alcuni algoritmi di analisi matematica II e uno dei problemi che si è presentato è **la distinzione tra punti di massimo e minimo locali e globali**. Prendiamo un attimo un esempio; sia data la funzione

$$f(x_1, x_2) = 4x_1^3 - 5x_2x_1 + 4x_2^2 \tag{5}$$

andiamo a calcolare il gradiente

$$\nabla f(x_1, x_2) = \begin{bmatrix} 12x_1^2 - 5x_2 \\ -5x_1 + 8x_2 \end{bmatrix} \tag{6}$$

supponiamo di voler calcolare i punti stazionari della funzione

$$\nabla f(x_1, x_2) = \begin{bmatrix} 12x_1^2 - 5x_2 \\ -5x_1 + 8x_2 \end{bmatrix} = 0 \tag{7}$$

come si osserva facilmente, non abbiamo un sistema di equazioni lineari e, nonostante siamo in $\mathbb{R}^2$, diventa estremamente complesso calcolarci le soluzioni. Potremmo provare a utilizzare l'algoritmo di enumerazione totale

Algoritmo di enumerazione totale

1. Calcolo il gradiente.
2. Risolvo il sistema $\nabla f(x) = 0$.
3. Provo la verifica della *matrice hessiana* su tutte le soluzioni.
4. Prendo, tra i risultati minimo (o massimi) locali, quello che più di tutti minimizza (o massimizza) la funzione obiettivo.

l'algoritmo appena descritto è estremamente inefficiente; durante il corso studieremo algoritmi che risultano essere molto più efficaci nella risoluzione di questi problemi.

1 Classificazione delle funzioni

Durante questa parte del corso affronteremo diversi problemi con diverse funzioni; in generale, però, possiamo classificare le funzioni che studieremo in tre famiglie

- **Funzioni *quadratiche***.
- **Funzioni *coercive***.
- **Funzioni *convesse***

1.1 Funzioni quadratiche

Una funzione $f : \mathbb{R}^n \to \mathbb{R}$ si definisce come **quadratica** se e solo se presenta la seguente struttura

$$< x, Ax > + < c, x > \tag{8}$$

quindi, se ha una parte di **secondo grado** e una **parte lineare**. Ad esempio, la funzione

$$f(x) = 3x_1^2 + 4x_1x_2 + x_2^2 + 5x_1 - 6x_2 \tag{9}$$

è **quadratica**, in quanto ha una parte di secondo grado $3x_1^2 + 4x_1x_2 + x_2^2$ e una parte lineare $5x_1 - 6x_2$. Se invece prendessimo la funzione

$$f(x) = 4x_1^3 - 5x_1x_2 \tag{10}$$

non potremmo dire che è **quadratica**, infatti vi è un termine che ha grado 3. Stessa cosa vale anche per la funzione

$$f(x) = x_1^2x_2 - 3x_2^2 \tag{11}$$

in quanto, anche se composto da due variabili diverse, il primo termine ha comunque grado 3.

Quando abbiamo funzioni **quadratiche**, il **gradiente** associato è composto solo da funzioni lineari e di conseguenza, il sistema

$$\nabla f(x) = 0 \tag{12}$$

è un **sistema lineare**.

1.2 Funzioni coercive

Una funzione si definisce **coerciva** se e solo se, per ogni successione $\{x_k\}$ tale che $||x_k|| \to \infty$, si ha che

$$\lim_{||x_k|| \to +\infty} f(x_k) = +\infty \tag{13}$$

In altre parole, affinché una funzione $f(x)$ sia coerciva, il suo valore deve crescere senza limiti quando la norma di x tende all'infinito, indipendentemente dalla direzione lungo cui ci si sposta.

Questo implica che la funzione non possa avere "minimi" in nessuna direzione: ovunque ci si sposti nello spazio, la funzione diverge al crescere di $||x||$.

Un esempio di funzione coerciva è la funzione quadratica $f(x) = ||x||^2 = x_1^2 + x_2^2 + \cdots + x_n^2$, dove $x = (x_1, x_2, \ldots, x_n)$ è un vettore in $\mathbb{R}^n$. Infatti, per questa funzione si ha

$$\lim_{||x||\to\infty} ||x||^2 = \infty.$$

In generale, una funzione quadratica di forma $f(x) = x^T Ax$ è coerciva se e solo se la matrice A che la definisce è definita positiva, cioè se tutti i suoi autovalori sono positivi.

Per verificare se una funzione è coerciva, bisogna considerare il comportamento della funzione quando $||x|| \to \infty$ lungo tutte le possibili direzioni. Ad esempio, per la funzione $f(x) = x_1^2 + 2x_2^2$, mentre $||x|| \to \infty$ lungo la direzione di x_1 o x_2, si ha sempre che $f(x) \to \infty$, quindi la funzione è coerciva.

1.3 Funzioni convesse

Una funzione si dice convessa se e solo se

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda(f(x_1) + (1-\lambda)f(x_2)) \quad \forall\lambda \in [0,1], \forall x_1, x_2 \in \mathbb{R}^n \tag{14}$$

Una funzione è convessa se e solo se la matrice **hessiana** è **semidefinita positiva** $Hf(x) \geq 0 \quad \forall x$; graficamente invece, una matrice viene definita essiana se il grafico considerato rispetto a un intervallo $[x_1, x_2]$ rimane sempre sotto la corda definita nello stesso intervallo. Il problema del primo metodo è la necessità di calcolarsi tutte le matrici Hessiane. Esiste inoltre un teorema fondamentale sulle funzioni convesse

Theorem 1.1. I punti stazionari di funzioni convesse, se esistono, sono dei minimi assoluti.

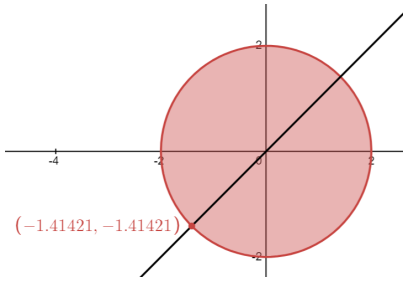
Quindi, una funzione convessa non ammette punti di sella e ne, tantomeno, punti di massimo, ma solamente punti di minimo assoluto (se esiste almeno un punto stazionario)

2 Esistenza di ottimi globali

In questo paragrafo analizzeremo alcune condizioni che garantiscono l'esistenza di punti di minimo globali

Theorem 2.1. Se la regione Ω è **limitata**, allora esiste un minimo globale di f su Ω.

Ad esempio, se prendessimo la funzione $f(x) = x_1 + x_2$ definita sulla regione $\Omega = \{x \in \mathbb{R}^2 : x_2^2 + x_2^2 - 4 \leq 0\}$, che per inciso, corrisponde ad una circonferenza di raggio 2 centrata in 0, si troverebbe facilmente che $(-\sqrt{2}, -\sqrt{2})$ è un punto di minimo globale. Lo si può anche osservare graficamente



Quando la regione non è limitata dobbiamo procedere in modo diverso, dobbiamo infatti ricondurci al teorema di Weierstrass aggiungendo ipotesi sugli insiemi di livello di f.

Theorem 2.2. Se esiste uno scalare α tale che il sottolivello $L_\alpha = \{x \in \Omega : f(x) \leq \alpha\}$ è *non vuoto* e *limitato*, allora esiste il un minimo globale di f su Ω

Prendiamo come esempio la funzione $e^{x_1+x_2}$ definita sulla regione $\Omega = \{x \in \mathbb{R}^2 : x_1 - x_2 \leq 0, \quad -2x_1 + x_2 \leq 0\}$ ammette un minimo globale anche se la regione non è **limitata**, infatti se andiamo a calcolare il sottolivello L_α otteniamo

$$f(x) \leq \alpha \iff e^{x_1+x_2} \leq \alpha \iff x_1 + x_2 \leq \ln(\alpha) \tag{15}$$

dalla quale ricaviamo che

$$L_\alpha = \{x \in \mathbb{R}^2 : x_1 - x_2 \leq 0, \quad -2x_1 + x_2 \leq 0, \quad x_1 + x_2 \leq \ln(\alpha)\} \tag{16}$$

Visto che, per $\alpha > 1$, la regione è **non vuota** e **limitata**, allora la funzione assume minimo globale su Ω. Inoltre, da queste considerazioni possiamo giungere a un'altra conclusione

Lemma 2.3. Se una funzione f è **coerciva**, allora esiste un minimo globale di f su Ω.

La coericività è solo una condizione sufficiente, ma non necessaria per l'esistenza di un minimo; se una funzione è **coerciva** ammette sicuramente un minimo sulla regione Ω, d'altra parte, se la funzione non è **coerciva** esistono casi in cui lo ammette e casi in cui non lo ammette. Prendiamo due esempi che ci permettano di capire meglio questo concetto; date le funzioni

$$\begin{aligned} f_1(x) &= \sin(x_1 x_2) \\ f_2(x) &= x_1 - x_2 \end{aligned} \tag{17}$$

Entrambe le funzioni non sono coercive, la prima funzione in quanto limitata tra $[-1, 1]$. La prima funzione, nonostante non sia coerciva ammette dei punti di minimo globali su $\mathbb{R}^2$ in tutti i punti (x_1, x_2) in cui $x_1 x_2$ è un multiplo intero di π; la seconda funzione non ammette minimi, in quanto, se prendessimo la successione $\{x_k\} = (0, k)$, con $k \to +\infty$ si avrebbe che

$$\lim_{k\to\infty} f(x_k) = -\infty \tag{18}$$

Una conclusione a cui possiamo giungere guardando la funzione appena studiata è che, per dimostrare che una funzione non ammette minimo su Ω è sufficiente dimostrare che è inferiormente illimitata sullo stesso intervallo; la dimostrazione è semplice, se una funzione ammette minimo globale su Ω, ed è quindi limitata inferiormente su Ω, esiste uno scalare M tale che $f(x) \geq M$ per ogni $x \in \Omega$. Per dimostrare che una funzione non è limitata su un intervallo Ω è sufficiente trovare una successione $\{x_k\} \subset \Omega$ tale per cui

$$\lim_{k\to+\infty} f(x_k) = -\infty \tag{19}$$

Prendiamo come esempio la funzione $f(x) = e^{x_1+x_2}$ definita su $\Omega = \mathbb{R}^2$: essa è limitata inferiormente su Ω, in quanto $f(x) \geq 0$ per ogni $x \in \Omega$, ma non ammette minimo globale sullo stesso insieme, in quanto

$$\inf_{x\in\mathbb{R}^2} f(x) = 0 \overset{\text{e}}{\longleftrightarrow} \nexists \ \overline{x} \in \mathbb{R}^2 : f(\overline{x}) = 0 \tag{20}$$

3 Domini della PNL

Affrontiamo adesso un tema che sarà di fondamentale importanza per la fruizione del corso; il tema dei **domini della PNL**. In generale, i domini Ω dei problemi di PNL sono definiti come

$$\Omega = \{x \in \mathbb{R}^n : g(x) \leq 0, \quad h(x) = 0\} \tag{21}$$

Dove $g(x)$ e $h(x)$ sono funzioni definite da $\mathbb{R}^n \to \mathbb{R}^m$ e $f(x)$ è definita da $\mathbb{R}^n \to \mathbb{R}$. Inoltre, durante tutto il corso assumeremo che le funzioni siano continue di classe C^∞; così facendo possiamo anche supporre che tutti i domini siano chiusi e che, per definizione, contengano anche il bordo. Un'altra considerazione che dobbiamo fare, sull'entità dei domini, riguarda se siano **limitati** o **illimitati**:

- Nei domini **limitati**, per teorema di Weierstrass, abbiamo assicurata l'esistenza di un punto di minimi globale.

- Nei domini **illimitati** non possiamo dire nulla di certo.

Durante questa parte di corso affronteremo solo domini limitati, ma, nota bene, all'orale è possibile che venga fatta qualche domanda **generale** sui domini illimitati. Inoltre, lavoreremo sempre su domini convessi; esiste una condizione che ci permette di verificare se un dominio è convesso

$$\begin{cases} g_i(x) \text{ è convesso} & \forall i = 1, \dots, q \\ h_j(x) \text{ è lineare} & \forall j = 1, \dots, k \end{cases} \iff \Omega \text{ è convesso} \tag{22}$$

Verificare se gli h_j sono lineari è facile; verificare se le g_i sono convesse richiede qualche passaggio in più

Algoritmo per la determinazione della convessità

1. Si calcola la matrice **hessiana**.
2. Si calcolano gli autovalori.
3. Se la matrice è semi-definita positiva e quindi, ha autovalori $\lambda \geq 0$, allora h_j è convessa.

Inoltre, quasi sempre i minimi che andiamo a trovare si trovano sul bordo e sono, quindi, dei **punti di frontiera**; matematicamente, un punto di frontiera è un punto, tale per cui, preso un intorno sufficientemente piccolo (una **palla**), esiste dei punti della palla che sono esterni all'insieme e dei punti della palla che sono interni all'insieme. Quindi, ricapitolando, i domini su cui lavoreremo sono sempre domini **chiusi** e **limitati** con **vincoli regolari**. Esistono anche delle condizioni, dette *condizioni di regolarità dei vincoli* che ci permettono di capire se i vincoli sono regolari:

Theorem 3.1. (Condizioni di regolarità)

1. Se g_i e h_j sono, **tutte**, funzioni lineari, allora i vincoli sono regolari in ogni punto $x \in \Omega$.
2. Se le funzioni g_i sono **tutte** convesse, le funzioni h_j sono tutte lineari e esiste un punto $\overline{x}$ tale per cui $g(\overline{x}) < 0$ e $h(\overline{x}) = 0$, allora i vincoli sono regolari in ogni punto $x \in \Omega$.
3. Se $\overline{x} \in \Omega$ ed i vettori
$$\begin{cases} \nabla g_i(\overline{x}) & \forall i = 1, \dots, n \\ \nabla h_j(\overline{x}) & \forall j = 1, \dots, n \end{cases} \tag{23}$$
sono tutti **linearmente indipendenti**, allora i vincoli sono regolari nel punto $\overline{x}$.

Facciamo qualche esempio per capire meglio la questione della regolarità dei vincoli. Assumiamo di avere la funzione

$$g_1(x) = x_1^2 + x_2^2 \leq 1 \tag{24}$$

L'insieme che stiamo analizzando è **chiuso** e **illimitato**; andandolo a disegnare graficamente si potrebbe notare che è anche convesso. Proviamo però a verificarlo tramite la condizione definita in precedenza, in altre parole, dimostriamo che $Hg(x)$ è semidefinita positiva. Calcoliamo per prima cosa il gradiente

$$\nabla g_1(x) = (2x_1, 2x_2) \tag{25}$$

e, successivamente, la matrice hessiana

$$Hg(x) = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \tag{26}$$

Calcoliamo adesso gli autovalori applicando la definizione

$$\det(Hg_1(x) - I\lambda) = (2 - \lambda)(2 - \lambda) \iff \lambda_1 = \lambda_2 = 2 \tag{27}$$

La matrice è **definita positiva** e quindi, implicitamente, anche **semi**-definita positiva; possiamo concludere che $g_1(x)$ è un insieme convesso. Proviamo adesso con la funzione

$$h_1(x) = x_1^2 + x_2^2 - 1 \tag{28}$$

Ci accorgiamo, anche abbastanza facilmente, che questa non è una funzione lineare e di conseguenza, per il primo vincolo, neanche convessa. Possiamo però vedere se è regolare controllando se soddisfa le due condizioni restanti.

4 Condizioni di ottimalità

In questo capitolo andiamo a enunciare quelle che sono le condizioni necessarie per l'ottimalità.

Theorem 4.1. Se $\overline{x}$ è un punto di **minimo locale** su Ω ed i vincoli sono **regolari** per $\overline{x}$, allora esistono due vettori, rispettivamente, $\overline{\lambda} \in \mathbb{R}^m$ e $\mu \in \mathbb{R}^p$ tali che $(\overline{x}, \overline{\lambda}, \overline{\mu})$ soddisfa il sistema di **Lagrange-Kuhn-Tucker**

$$\begin{cases} \nabla f(\overline{x}) + \sum_{i=1}^{m} \overline{\lambda}_i \nabla g_i(\overline{x}) + \sum_{j=1}^{p} \overline{\mu}_j \nabla h_j(\overline{x}) = 0 \\ \overline{\lambda}_i g_i(\overline{x}) = 0 \quad \forall\, i = 1, \dots, m \\ \overline{\lambda} \geq 0 \\ g(\overline{x}) \leq 0 \\ h(\overline{x}) = 0 \end{cases} \tag{29}$$

Se, invece, $\overline{x}$ è un punto di **massimo**, allora esistono due vettori, rispettivamente, $\overline{\lambda} \in \mathbb{R}^m$ e $\mu \in \mathbb{R}^p$ tali che $(\overline{x}, \overline{\lambda}, \overline{\mu})$ soddisfa il sistema di **Lagrange-Kuhn-Tucker**

$$\begin{cases} \nabla f(\overline{x}) + \sum_{i=1}^{m} \overline{\lambda}_i \nabla g_i(\overline{x}) + \sum_{j=1}^{p} \overline{\mu}_j \nabla h_j(\overline{x}) = 0 \\ \overline{\lambda}_i g_i(\overline{x}) = 0 \quad \forall\, i = 1, \dots, m \\ \overline{\lambda} \leq 0 \\ g(\overline{x}) \leq 0 \\ h(\overline{x}) = 0 \end{cases} \tag{30}$$

Notiamo alcune caratteristiche, il sistema che abbiamo appena descritto è un sistema di $n+m+p$ incognite. Inoltre, tutte le soluzioni del sistema sono dei punti stazionari, quindi punti tali per cui $\nabla f(\overline{x}) = 0$. Una considerazione degna di nota è che, se risolviamo il sistema conoscendo dei valori di $\overline{x}$, anche se otteniamo che $\lambda \geq 0$, non possiamo essere sicuri che il punto sia di minimo; sappiamo solo che il punto **non è di massimo**. Dimostriamolo con un esempio, consideriamo la funzione $f(x) = x_1 + x_2$ sulla regione $\Omega = \{x \in \mathbb{R}^2 : -x_1^2 - x_2^2 + 2 \leq 0\}$. L'unica soluzione che si ottiene risolvendo il sistema LKT è $x = (1, 1)$ con $\lambda = 1/2$; per le condizioni appena descritte dovrebbe essere un punto di minimo, se però provassimo a prendere la successione

$$x_k = \left(1 - \frac{1}{k}, \sqrt{2 - \left(1 - \frac{1}{k}\right)^2}\right) \in \Omega \tag{31}$$

ci accorgiamo che essa converge a $(1, 1)$ e che $f(x_k) < f(1, 1)$. Il sitema LKT, da solo, non è un sistema in grado di trovare i punti di minimo; manca ancora una condizione, detta *condizione di ottimalità del primo ordine*.

Proviamo a fare due supposizioni sul punto $\overline{x}$. Supponiamo che il punto $\overline{x}$ sia interno, quindi abbiamo che $g_i(\overline{x}) < 0$ per ogni vincolo. Affinché la seconda condizione delle condizioni KKT sia verificata, è necessario che $\overline{\lambda}_i = 0$, in quanto

$$\overline{\lambda}_i \cdot g_i(\overline{x}) = 0 \quad \Rightarrow \quad \overline{\lambda}_i = 0$$

In altre parole, il prodotto tra uno scalare diverso da 0 e un altro scalare è uguale a 0 solo se il secondo scalare è 0. La conseguenza è che, se $\overline{\lambda}_i = 0$ e quindi il punto è interno, non abbiamo alcuna restrizione indotta dai vincoli attivi e non possiamo concludere nulla riguardo alla natura del punto (massimo o minimo).

Quando, invece, ci troviamo su un bordo del dominio, la presenza dei moltiplicatori di Lagrange $\overline{\lambda}_i$ ci permette di determinare quando fermarci nel muoverci lungo la direzione del gradiente della funzione obiettivo. Infatti, i moltiplicatori $\overline{\lambda}_i$ ci indicano in che misura i vincoli attivi influenzano la direzione di ottimizzazione e, di conseguenza, il punto in questione.

Un'ultima casistica su cui facciamo un breve accenno è quella dei domini illimitati; quando abbiamo un dominio illimitato ci troviamo in una situazione non ottimale, infatti non vi è alcuna

garanzia sull'esistenza di un punto di massimo o di minimo, cosa che invece abbiamo, grazie a Weierstrass, nei domini limitati (supponendo che le funzioni sia continue).

Prima di enunciare la *condizione di ottimalità del primo ordine*, proviamo a fare un breve esercizio di applicazione del sistema LKT. Data la seguente tabella e le rispettive funzioni

Componenti di x	**Moltiplicatori** λ
x_1	λ_1
x_2	λ_2

Tabella 17.1: Tabella con x e λ

$$f(x) = \tag{32}$$

Calcolare i coefficienti o i punti stessi, andando a ipotizzare cosa potrebbero essere. Diamo alcune definizioni utili per la scrittura della condizione

Definition 4.1. (Dominio convesso) Dato Ω, diciamo che è un **dominio convesso** se e solo se $g_i(x)$ sono **convesse** per ogni $i = 1, \dots, m$ e $h_i(x)$ sono **lineari** per ogni $j = 1, \dots, p$.

Definition 4.2. (Problema convesso) Sia Ω un **dominio convesso**; un problema nella forma

$$\begin{cases} \min f(x) \\ x \in \Omega \end{cases} \tag{33}$$

è detto **problema convesso** se e solo se $f(x)$ è **convessa**. In un problema convesso non esistono né minimi locali, né punti di sella.

Definition 4.3. (Problema concavo) Sia Ω un **dominio convesso**; un probema nella forma

$$\begin{cases} \min f(x) \\ x \in \Omega \end{cases} \tag{34}$$

è detto **problema concavo** se e solo se $f(x)$ è **concava**. In un problema concavo non esistono né punti di massimo locale, ne punti di sella.

Passiamo adesso a enunciare la **condizione di ottimalità del primo ordine**.

Theorem 4.2. (Condizione sufficiente di ottimalità del primo ordine) Siano $f(x)$ **convessa** (**concava**), $g_i(x)$ **convessa** per ogni $i = 1, \dots, m$ e $h_j(x)$ *lineare* per ogni $j = 1, \dots, p$. Se $(\overline{x}, \overline{\lambda}, \overline{\mu})$ è una soluzione del sistema LKT, allora $\overline{x}$ è un **minimo** (**massimo**) globale di $f(x)$ su Ω.

Questo teorema permette di ottenere un risultato fortissimo; a patto che siano rispettate determinate condizioni sul dominio e sulla funzione, possiamo classificare direttamente i punti di massimo e di minimo senza dover verificare con la matrice **hessiana**. Se il problema è concavo allora le soluzioni del sistema LKK sono punti di massimo, mentre se il problema è convesso, allora, sono punti di minimo.

Quelle che abbiamo definito adesso sono delle condizioni sufficienti per l'ottimalità; infatti è sufficiente che vengano rispettate le condizioni affinché il punto possa essere classificato come minimo. Esistono, però, anche delle corrispettive condizioni necessarie per l'ottimalità

Theorem 4.3. (Condizione necessaria di ottimalità) Sia $\overline{x} \in \Omega$ **massimo** (**minimo**), allora esiste $\lambda \leq 0$ ($\lambda \geq 0$) e esiste μ, tali per cui

$$\begin{cases} \nabla f(\overline{x}) + \sum_{i=1}^{m} \overline{\lambda}_i \nabla g_i(\overline{x}) + \sum_{j=1}^{p} \overline{\mu}_j \nabla h_j(\overline{x}) = 0 \\ \overline{\lambda}_i g_i(\overline{x}) = 0 \quad \forall\, i = 1, \dots, m \\ \overline{\lambda} \leq 0 \\ g(\overline{x}) \leq 0 \\ h(\overline{x}) = 0 \end{cases} \tag{35}$$

Esiste anche un caso particolare di problemi definiti su domini convessi, sono tutti i problemi in cui il dominio è un poliedro. In un poliedro abbiamo i vincoli $g_i(x)$ non definiti e i vincoli $h_j(x)$ lineari; di conseguenza rispetta la definizione di dominio convesso. In questi casi abbiamo delle condizioni di ottimalità che semplificano ancora di più la classificazione dei punti

Theorem 4.4. (Condizioni di) Sia Ω un poliedro e $f(x) \in C^1$; allora

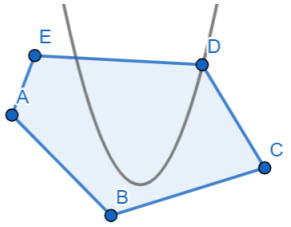
- Se $f(x)$ è convessa, allora, uno dei vertici del poliedro è un **massimo globale**.
- Se $f(x)$ è concava, allora, uno dei vertici del poliedro è un **minimo globale**.

Queste condizioni ci danno solo una certezza, che se viene rispettata la condizione allora esiste un punto di minimo o di massimo; il problema è che non avendo moltiplicatori λ non siamo in grado di capire dal sistema se il punto è di max o di min, dobbiamo quindi procedere con una verifica usando la matrice **hessiana**.

Un fatto utile da osservare è che, se la funzione $f(x)$ è convessa e il dominio Ω è un poliedro, il massimo è necessariamente **sul bordo**. Ciò deriva da una proprietà particolare che abbiamo studiato nelle prime lezioni sulle funzioni della PNL

Theorem 4.5. Se $f(x)$ è convessa, allora il punto $\overline{x}$ tale per cui $\nabla f(\overline{x}) = 0$ è un **punto di minimo**.

Di conseguenza, la funzione tenderà a crescere costantemente e, di conseguenza, il massimo si troverà dove la funzione interseca il bordo del poliedro. Lo possiamo visualizzare graficamente con il seguente esempio



5 Metodo delle restrizioni

Questo metodo permette di determinare se il punto che risolve il sistema delle condizioni di Karush-Kuhn-Tucker (KKT) è un punto di **massimo**, **minimo** locale o un punto di sella. L'osservazione preliminare che dobbiamo fare è la seguente:

Lemma 5.1. Se $\overline{x}$ è un punto di **massimo/minimo** della funzione sul dominio $\Omega = \{x \in \mathbb{R}^n : g(x) \leq 0,\ h(x) = 0\}$, allora $\overline{x}$ è un punto di massimo/minimo della funzione sulla restrizione ϕ.

L'idea del *metodo delle restrizioni* è che, se un punto è un minimo locale, allora è un minimo locale anche per ogni possibile restrizione. Pertanto, sarà sufficiente trovare una restrizione per cui il punto diventi un massimo o un punto di sella, per far cadere l'ipotesi del teorema e classificare il punto che stiamo analizzando come un punto di sella. Consideriamo come esempio il seguente problema:

Esempio 1

Trovare i ***massimi*** *e* ***minimi*** *della funzione*

$$f(x) = -x_1^2 - 2x_2^2 + 8x_2 \tag{36}$$

sull'insieme

$$\{x \in \mathbb{R}^2 : -x_1^2 - x_2^2 + 1 \leq 0, \quad x_1^2 - x_2 - 2 \leq 0\} \tag{37}$$

Il primo passo è analizzare il dominio della funzione. Il dominio è costituito dall'esterno di una sfera di raggio 1, intersecato con l'interno di una parabola:

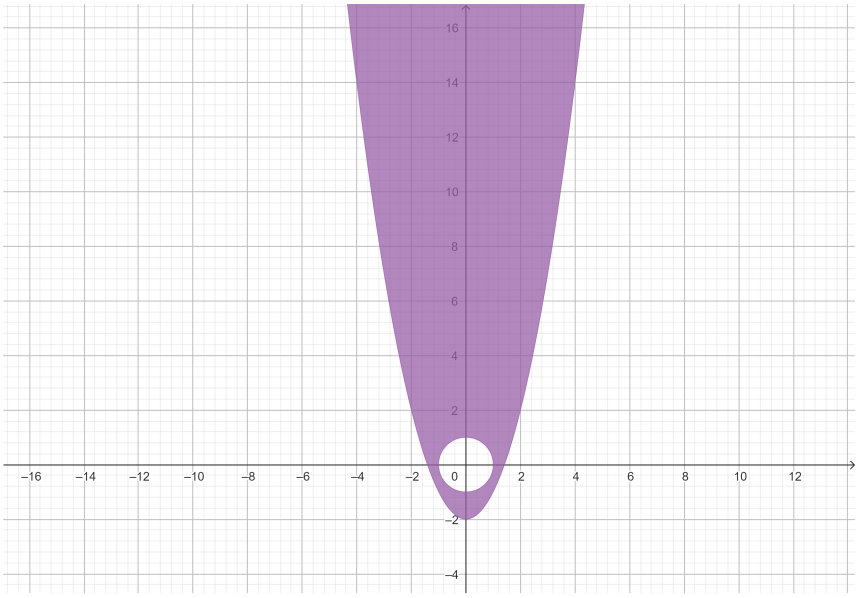


Figura 17.1: Dominio dell'esempio

Il dominio è illimitato, quindi non possiamo applicare il Teorema di Weierstrass. D'altra parte, la funzione è **concava**, quindi, se la funzione ammette un punto stazionario, tale punto $\overline{x} \in \Omega$, con $\nabla f(\overline{x}) = 0$, è un punto di **massimo globale**. Il gradiente della funzione è:

$$\nabla f(x) = [-2x_1; -4x_2 + 8] \tag{38}$$

Questo si annulla per $(0, 2)$. Oltre a questa considerazione, visto che il dominio è illimitato, non possiamo fare altre osservazioni utili. Supponiamo ora di aver verificato che il dominio è regolare per i punti che ci interessano. Prendiamo come esempio il punto:

$$x^* = \left(\frac{\sqrt{15}}{2}, \frac{7}{4}\right) \tag{39}$$

I cui moltiplicatori Lagrange (LKT) sono $(0, 1)$. Poiché i moltiplicatori sono ≥ 0, il punto non può essere un massimo locale, ma potrebbe essere un minimo locale o un punto di sella. Proviamo a verificarlo con il metodo delle restrizioni. In particolare, prendiamo la restrizione $(0, t)$:

$$f(0, t) = -2t^2 + 8t \tag{40}$$

La funzione è concava e presenta un unico punto stazionario:

$$f'(0, t) = -4t + 8 \implies -4t + 8 = 0 \longrightarrow t = \frac{1}{2} \tag{41}$$

Quindi, abbiamo trovato una restrizione per cui il punto x^* non è un punto di minimo. Pertanto, possiamo classificarlo come un punto di sella.

Capitolo 18

Algoritmi iterativi per la risoluzione

Supponiamo di avere un punto ammissibile x_0, considerare la restrizione della funzione f ad una semiretta di direzione d_k uscente dal punto x

$$\phi(t) = f(x_k + td_k), \quad \text{con } t \geq 0 \tag{1}$$

La maggior parte degli algoritmi che studieremo sono detti *metodi di discesa*; in questi metodi le *direzioni* d^k *ottime* sono quelle per cui

$$\phi(x^{k+1}) < \phi(x^k) \tag{2}$$

Cosa che è vera se e solo se d_k è una *direzione di discesa* e, di conseguenza, se

$$\frac{d\phi(x)}{dx} = \phi^{(1)}(x) < 0 \tag{3}$$

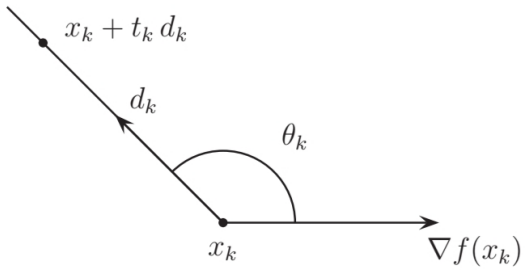
Inoltre, visto che

$$\phi^{(1)}(x) = d^k \nabla\phi^{(1)}(x) \tag{4}$$

affinché sia rispettata la condizione $\phi^{(1)}(x) < 0$ sia rispettata, dobbiamo assicurarci che

$$d_k \cdot \nabla\phi^{(1)}(x) = ||d_k||\ ||\nabla\phi^{(1)}(x)|| \cos(\theta_k) < 0 \tag{5}$$

Situazione che può essere verificata solo quando θ_k è maggiore di un angolo retto.



L'idea alla base dei metodi che studieremo durante questo capitolo è quella di costruire una successione del tipo

$$x^{k+1} = x^k + t_k d^k \tag{6}$$

Sostanzialmente, prendo il punto precedente e lo uso come punto di partenza della semiretta che congiunge x_k a x^{k+1} lungo la direzione d^k. Il coefficiente t_k, detto **passo**, può essere ottenuto minimizzando la restrizione $\phi(t)$ definita in questa sezione.

Quando iteriamo gli algoritmi per l'ottimizzazione delle funzioni dobbiamo anche definire dei criteri di STOP. Rispetto alla *programmazione lineare* non abbiamo criteri così facilmente definibili; dobbiamo scegliere noi tra diverse possibilità:

- Determinare un numero massimo di step.
- Impostare un tempo masimo
- Impostare un limite Δ tale per cui

$$|f(x^{k+1}) - f(x^k)| < \Delta \tag{7}$$

- Impostare un limite Δ_1 tale per cui

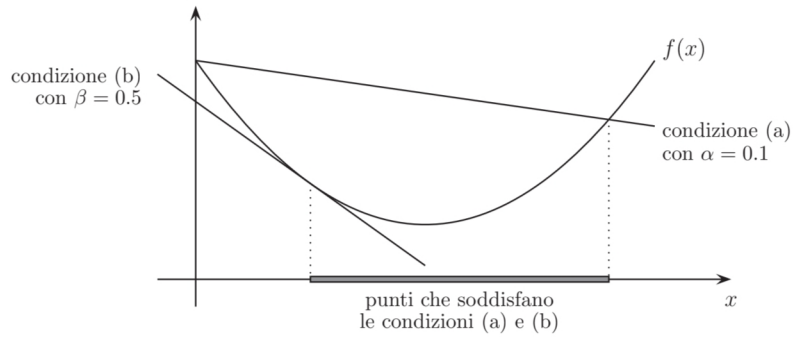
$$|\nabla f(x^k)| < \Delta \tag{8}$$

La ricerca del minimo di una funzione è un processo che è, computazionalmente parlando, estremamente costoso. Esistono dei metodi, detti *metodi inesatti*, nati con lo scopo di ovviare a questa problematica. Supponiamo, preliminarmente, che $f'(0) < 0$. Il primo metodo si basa sulla ricerca di un punto $\overline{x} > 0$ che soddisfi le **condizioni di Armijo-Goldstein-Wolfe**;

Condizioni di Armijo-Goldstein-Wolfe

- $f(\overline{x}) \leq f(0) + \alpha \nabla f(0) x$
- $\nabla f(\overline{x}) \geq \beta \nabla f(0)$
- $0 < \alpha < \beta < 1$

α e β sono due parametri. La prima condizione garantisce che $\overline{x}$ non sia troppo grande, in tal modo evitiamo che il membro destro tenda a $-\infty$ e che quindi la condizione non sia soddisfatta. La seconda condizione garantisce invece che $\overline{x}$ non sia troppo piccolo: in tal caso si avrebbe $f'(\overline{x}) \simeq f'(0)$ e la condizione non potrebbe sussistere.



Esiste anche un risultato che ci garantisce l'esistenza di un intervallo in cui vengono soddisfatte queste condizioni

Theorem 0.1. Se f è **limitata inferiormente** e $\nabla f(0) < 0$, allora esiste un intervallo (a, b) tale che le condizioni di **Armijo-Goldstein-Wolfe** sono soddisfatte per ogni $x \in (a, b)$.

1 Metodo del gradiente a passo fissato

Il metodo del gradiente a passo fissato è l'algoritmo di ottimizzazione per funzioni non vincolate più utilizzato nell'ambito del **Machine Learning**. L'algoritmo è estremamente intuitivo: si fissa una direzione d_k, che può essere di discesa ($-\nabla f(x)$ ad esempio) o di salita ($\nabla f(x)$ ad esempio), si fissa un **passo costante** $t_k = \eta \in \mathbb{R}$ e si procede lungo la direzione del gradiente in x_k, controllando che il gradiente della funzione **converga verso 0**.

Algoritmo del gradiente a passo fissato

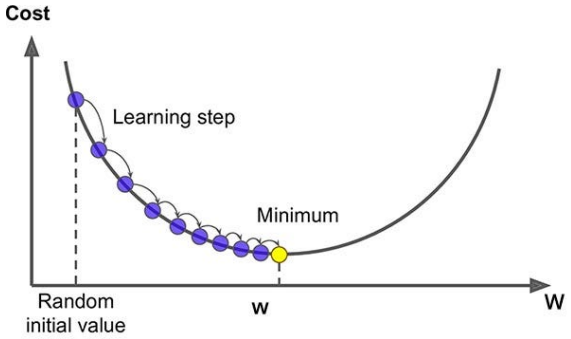
1. Scegliere $x_k \in \mathbb{R}^n$, $t_k = \eta \in \mathbb{R}$.
2. Finché $\nabla f(x_k) \neq 0$, si varia $x_{k+1} = x_k - \eta \nabla f(x_k)$ (se minimizziamo) o $x_{k+1} = x_k + \eta \nabla f(x_k)$ (se massimizziamo), e si incrementa k.

Ovviamente, se abbiamo una funzione convessa cerchiamo il minimo, se abbiamo una funzione concava cerchiamo il massimo. Il motivo è presto detto: per definizione, in una funzione **convessa** (**concava**), i punti stazionari sono tutti punti di **minimo** (**massimo**).

La scelta di η non è da ritenersi casuale, infatti, occorre scegliere η in modo che i passi successivi non vadano a oltrepassare il punto $\nabla f(x_k) = 0$. In generale, la scelta del passo η deve soddisfare la seguente condizione:

$$0 \leq \eta \leq \frac{2}{L} \longleftrightarrow L \geq \left| \frac{\partial^2 f(x)}{\partial x_i \partial x_j} \right| \tag{9}$$

Questa condizione garantisce che il gradiente non cambi troppo velocemente e che il passo scelto permetta all'algoritmo di convergere in modo stabile senza oscillare troppo intorno al punto $\nabla f(x_k) = 0$.



2 Metodo del gradiente a passo ideale

Questo metodo è forse uno dei metodi più classici che esistono per ottimizzare le funzioni non lineari che siano almeno di classe C^1 e che non siano sottoposte a vincoli. Come direzione di ricerca scegliamo quella opposta al gradiente, ossia $d_k = -\nabla f(x_k)$ e la chiamiamo **direzione di massima discesa** nel punto x_k. Calcoliamo adesso il nostro passo, definito come

$$t_k = \arg\min_{t \geq 0} f(x_k + t d_k) = \arg\min_{t \geq 0} f(x_k - t \nabla f(x_k)) \tag{10}$$

Dove $f(x_k + t d_k)$ è una funzione a una singola variabile. Il calcolo dei minimi di una funzione in una variabile è uno dei prerequisiti del corso, dovrebbe essere quindi relativamente facile. D'altra parte, nel libro, vengono enunciati dei metodi iterativi che permettono di calcolare il passo, ne enunciamo uno per completezza; questo algoritmo prende il nome di **metodo di Newton**

Metodo di Newton

- Scegliere $x_0 \in \mathbb{R}$ e porre $k = 0$.
- Finché $f'(x_k) \neq 0$ si calcola

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)} \tag{11}$$

Si incrementa k: $k = k + 1$ e si torna al secondo passo.

Rispetto al precedente algoritmo abbiamo un miglioramento non da poco, infatti, non abbiamo più un passo costante che rimane uguale ad ogni iterazione, ma abbiamo la possibilità di scegliere, ad ogni iterazione, il miglior passo possibile. Anche in questo caso, il criterio di STOP dell'algoritmo, è che $\nabla f(x_k) = 0$.

Algoritmo del gradiente a passo ideale

1. Scegliere $x_0 \in \mathbb{R}^n$ e porre $k = 0$.
2. Fintanto che $\nabla f(x_k) \neq 0$
   (a) Calcola una soluzione t_k del problema

$$t_k = \arg\min_{t>0} f(x_k - t\nabla f(x_k)) \tag{12}$$

   (b) Aggiornare $x_{k+1} = x_k - t_k \nabla f(x_k)$ e $k = k + 1$.

Possiamo fare un'osservazione molto importante sul metodo del gradiente: gli svantaggi di questo metodo sono costituiti dalla difficoltà di cercare, ad ogni iterazione, il passo esatto e dal suo andamento a zig zag dovuto al fatto che due direzioni successive sono tra loro ortogonali. Anche in questo caso abbiamo un teorema di convergenza.

Theorem 2.1. Se f è **coerciva**, comunque si sceglie un punto iniziale x_0, il metodo del gradiente con ricerca esatta del passo, o **trova un punto stazionario di f** dopo un numero finito di iterazioni, o trova una successione limitata $\{x_k\}$ tale che **ogni suo punto di accumulazione è un punto stazionario di f**

3 Metodo del gradiente con back-tracking

Come accennato nella sezione sul backtracking e sulle condizioni di Armijo-Goldstein-Wolfe (AGW), il calcolo del minimo ad ogni iterazione, sebbene preciso, ha un alto costo computazionale. Una variante dell'algoritmo del gradiente utilizza invece le condizioni AGW per determinare il passo in modo più efficiente. Sebbene questa variante comporti una convergenza più lenta rispetto alla ricerca del passo ottimale, ha il vantaggio di ridurre significativamente il costo computazionale, rendendo l'algoritmo più adatto a problemi con funzioni complesse o dataset di grandi dimensioni.

Come nei due casi precedenti, anche in questo abbiamo un criterio di stop, l'algoritmo si ferma quando $\nabla f(x_k) = 0$ Andiamo a descrivere i vari passi dell'algoritmo

Metodo del gradiente con Back-tracking

- Fissa $\alpha, \gamma \in (0, 1)$ e $\bar{t} > 0$. Scegliere il punto di partenza $x_k \in \mathbb{R}^n$ e porre $k = 0$.
- Finché $\nabla f(x_k) \neq 0$ allora si calcola il più piccolo naturale m tale per cui

$$f(x_k - \gamma^m \bar{t} \nabla f(x_k)) \leq f(x_k) - \alpha \gamma^m \bar{t} ||\nabla f(x_k)||^2 \tag{13}$$

successivamente si aggiorna le variabili $t_k = \gamma^m \bar{t}$, $x_{k+1} = x_k - t_k \nabla f(x_k)$ e $k = k + 1$.

Anche in questo caso abbiamo un teorema di convergenza

Theorem 3.1. Se f è **coerciva**, comunque si sceglie un punto iniziale x_0, il metodo del gradiente con **backtracking**, o **trova un punto stazionario di f** dopo un numero finito di iterazioni, o trova una successione limitata $\{x_k\}$ tale che **ogni suo punto di accumulazione è un punto stazionario di f**

Possiamo riassumere le principali differenze tra **algoritmo del gradiente a passo ideale** e **algoritmo del gradiente con backtracking** nella seguente tabella.

Caratteristica	**Ricerca con Passo Ideale**	**Condizioni di Armijo/Goldstein/Wolfe**
Complessità computazionale	Alta, richiede minimizzare la funzione lungo la direzione del gradiente ad ogni iterazione	Bassa, il passo è determinato con un semplice criterio di discesa sufficiente
Velocità di convergenza	Più rapida, in quanto calcola il passo ottimale per ogni iterazione	Più lenta, poiché non calcola il passo ottimale, ma usa un criterio approssimato
Stabilità	Potenzialmente meno stabile (se la funzione è irregolare o ha molte oscillazioni)	Maggiore stabilità, evita passi troppo lunghi o troppo corti
Applicabilità pratica	Più difficile da implementare e poco usato in applicazioni pratiche complesse	Molto usato in applicazioni pratiche, come l'ottimizzazione nel machine learning
Tipo di controllo sul passo	Esatto e ottimale per ogni iterazione	Approssimato, con condizioni di discesa sufficiente e controllo della curvatura

Tabella 18.1: Confronto tra la ricerca con passo ideale e le condizioni di Armijo/Goldstein/Wolfe

4 Metodo di Newton a passo costante

Tra tutti i metodi visti fino ad ora, l'algoritmo di Newton è il più veloce. Ad ogni iterazione, questo metodo risolve un sistema di equazioni lineari ottenuto facendo un approssimazione lineare di $\nabla f(x)$ in un intorno di x_k, ossia

$$\nabla f(x) \simeq \nabla f(x_k) + Hf(x_k)(x - x_k) \tag{14}$$

Se la matrice $Hf(x_k)$ è invertibile, allora x_{k+1} è il punto in cui si annulla tale approssimazione, ossia

$$\begin{cases} x_{k+1} = x_k + [Hf(x_k)]^{-1}\nabla f(x_k) & \text{se massimizzo} \\ x_{k+1} = x_k - [Hf(x_k)]^{-1}\nabla f(x_k) & \text{se minimizzo} \end{cases} \tag{15}$$

Anche in questo caso il criterio di STOP è che $\nabla f(x_k) = 0$. Definiamo formalmente i passi dell'algoritmo

Metodo di Newton a passo costante

- Scegliere un punto $x_k \in \mathbb{R}^n$ e porre $k = 0$.
- Finché $\nabla f(x_k) \neq 0$ fai
  1. $d_k = -[Hf(x_k)]^{-1}\nabla f(x_k)$
  2. $x_{k+1} = x_k + d_k$
  3. $k = k + 1$

Osserviamo che, tra tutti i metodi studiati è il più veloce, d'altra parte però, se il punto scelto è distante dal punto di **massimo/minimo**, è possibile che l'algoritmo impazzisca; inoltre, l'ipotesi alla

base di tutto l'algoritmo è che $Hf(x_k)$ sia **invertibile**. Definiamo adesso il teorema fondamentale sull'algoritmo

Theorem 4.1. Supponiamo che esista un minimo locale $\overline{x}$ tale per cui la matrice $Hf(\overline{x})$ è **definita positiva**. Allora esiste un intorno U di $\overline{x}$ tale che, comunque si scelga $x_k \in U$, la successione $\{x_k\}$ generata dal metodo di Newton converge a $\overline{x}$ e la convergenza è quadratica.

Capitolo 19

Algoritmi iterativi per la risoluzione di problemi vincolati

1 Metodo di Frank-Wolfe

Il metodo di Frank-Wolfe è un algoritmo iterativo che si applica per trovare un punto stazionario di un problema di massimo vincolato. Supponiamo di avere un problema nella forma

$$\begin{cases} \min f(x) \\ Ax \leq b \end{cases} \tag{1}$$

Dove A è una matrice $m \times n$ e $b \in \mathbb{R}^m$. Supponiamo di partire da un punto ammissibile x_k. Approssimiamo la nostra funzione in un intorno di x_k con gli sviluppi di Taylor del primo ordine

$$f(x) \simeq f(x_k) + \nabla f(x_k)(x - x_k) + o|x - x_k| \tag{2}$$

Possiamo escludere il coefficiente $o|x - x_k|$ e anche i coefficenti costanti, rimaniamo quindi con $\nabla f(x_k)^T x$. Risolviamo quindi il problema lineare associato in x_k

$$\begin{cases} \min \nabla f(x_k)^T x \\ Ax \leq b \end{cases} \tag{3}$$

Il problema in questione è un problema di programmazione lineare, possiamo quindi applicare l'algoritmo del simplesso o la direttiva **linprog**. Indichiamo come y_k la soluzione del problema lineare. Se anche il punto x_k risolve il problema linearizzato

$$\nabla f(x_k)^T y_k = \nabla f(x_k)^T x_k \tag{4}$$

allora x_k è un punto stazionario e, di conseguenza, l'algoritmo termina. Altrimenti, se

$$\nabla f(x_k)^T y_k < \nabla f(x_k)^T x_k \tag{5}$$

abbiamo che

$$\nabla f(x_k)(y_k - x_k) < 0 \tag{6}$$

ciò implica che la direzione $y_k - x_k$ sia una direzione di discesa per f. In tal caso il metodo trova un nuovo punto x_{k+1} uguale al minimo della funzione $f(x)$ ristretta al segmento di estremi x_k e y_k. Ossia, il passo t_k è la soluzione del sistema

$$\begin{cases} \min f(x_k + t(y_k - x_k)) \\ t \in [0, 1] \end{cases} \tag{7}$$

Possiamo riassumere il metodo nei seguenti passi

Algoritmo di Frank-Wolfe

- Scegli un punto ammissibile x_0 e poni $k = 0$.
- Calcola una soluzione y_k del sistema linearizzato

$$\begin{cases} \min \nabla f(x_k)^T x \\ Ax \leq b \end{cases} \tag{8}$$

- Se $\nabla f(x_k)(y_k - x_k) = 0$ allora ci fermiamo, abbiamo trovato un punto stazionario. Altrimenti calcoliamo una soluzione t_k del problema

$$\begin{cases} \min f(x_k + t(y_k - x_k)) \\ t \in [0, 1] \end{cases} \tag{9}$$

- Poniamo $x_{k+1} = x_k + t_k(y_k - x_k)$ e $k = k + 1$. Torniamo al passo 2.

L'idea alla base del metodo di **Frank-Wolfe** è quella di prendere un punto ammissibile x_k di partenza. A questo punto dobbiamo determinare la direzione di massima discesa d_k. Per trovarlo, l'idea è quella di trovare il vertice in cui la funzione assume il valore minimo possibile, risolvendo il problema

$$\begin{cases} \min \nabla f(x_k)^T x \\ Ax \leq b \end{cases} \tag{10}$$

Una volta risolto il problema, cosa che possiamo fare in tanti metodi, dalle rette di iso-costo alla direttiva linprog, possiamo calcolare la nostra direzione di discesa $d_k = y_k - x_k$. Ci muoviamo lungo la semiretta che collega il nostro punto x_k al vertice y_k e andiamo a calcolare il passo t_k risolvendo il problema linearizzato

$$t_k = \arg\min_{t \in [0,1]} f(x_k + t(y_k - t_k)) \tag{11}$$

Successivamente posso aggiornare gli indici

$$\begin{aligned} x_{k+1} &= x_k + t_k(y_k - x_k) \\ k &= k + 1 \end{aligned} \tag{12}$$

Domande orale

Se il dominio non fosse convesso, cosa possiamo dedurre ? *Se il dominio è* ***non convesso****, non possiamo dire nulla sull'entità dei punti che troviamo attraverso l'algoritmo, neanche se risolvono LKKT. Fatta eccezione per funzioni* ***concave*** *(****convesse****), infatti se il punto stazionario, che per definizione è il* ***minimo*** *(****massimo****), è nel poliedro, abbiamo una soluzione.*,
Se la funzione è convessa (concava), cosa possiamo dire sull'algoritmo ? *Se abbiamo una funzione* ***convessa*** *(****concava****), visto che per teorema l'algoritmo converge sempre a un punto stazionario, allora converge sempre al* ***minimo globale*** *(****massimo globale****)*
Come calcolo il minimo di una funzione convessa ? Calcolo il gradiente, vedo dove si annulla, se il punto è incluso nel poliedro l'ho trovato.

Esiste, come nel caso del metodo del **gradiente**, un teorema a riguardo del metodo Frank-Wolfe

Theorem 1.1. Supponiamo che la regione ammissibile sia limitata oppure che la funzione f sia coerciva. Allora, comunque si scelga il punto iniziale x_0, il metodo **frank-wolfe** o trova un punto stazionario dopo un numero finito di iterazioni, o trova una successione $\{x_k\}$ tale per cui, ogni punto di accumulazione, è un punto stazionario.

2 Metodo del gradiente proiettato

Il metodo del gradiente proiettato, così come anche il metodo di Frank-Wolfe, è un metodo utilizzato per trovare un punto stazionario di un problema vincolato. Alla k-esima iterazione valutiamo il punto corrente x_k

- Se il punto è **interno** alla regione ammissibile, scegliamo come $d_k = -\nabla f(x_k)$.
- Se il punto è sul **bordo** della regione ammissibile, scegliamo come d_k la proiezione ortogonale di $-\nabla f(x_k)$ sul **sotto-spazio vettoriale** definito dai vincoli attivi nel punto x_k.

Procediamo a trattare cosa voglia dire **proiezione ortogonale sul sottospazio vettoriale dei vincoli attivi**. Se prendiamo uno spazio vettoriale V e un sottospazio vettoriale W tale che $W \subset V$, e prendiamo un vettore $v \in V$ con $v \notin W$, diciamo che la proiezione ortogonale $P_W(v)$ è il vettore $w \in W$ che rappresenta la versione di v "più vicina" a W. In altre parole, è il punto di W che si trova a distanza minima da v, ed è come se proiettassimo v perpendicolarmente su W.

Lemma 2.1. La proiezione ortogonale di un vettore $y \in \mathbb{R}^n$ sul sottospazio $S = \{x \in \mathbb{R}^n : Mx = 0\}$ è data dal vettore Hy, dove la matrice $H = I - M^T(MM^T)^{-1}M$

L'algoritmo del gradiente proiettato può essere sintetizzato nei seguenti passi

Algoritmo del gradiente proiettato

- Si sceglie un punto ammissibile x_0 e si pone $k = 0$.
- Sia $\mathcal{A} = \{i \in A_i x_k = b_i\}$ e M la sottomatrice di A avente per righe i vettori A_i con $i \in \mathcal{A}$. Possiamo fare un paragone azzardato e paragonare M alla matrice di base che utilizzavamo nella programmazione lineare.
- Calcola la matrice $H = I - M^T(MM^T)^{-1}M$ e la direzione $d_k = -H\nabla f(x_k)$.
- Se $d_k \neq 0$ si calcola una soluzione $\hat{t}_k$ del problema

$$\begin{cases} \max t \\ A(x_k + td_k) \leq b \end{cases} \tag{13}$$

  si calcola successivamente una soluzione t_k del problema

$$\min_{0 \leq t_k \leq \hat{t}_k} f(x_k + td_k) \tag{14}$$

  si aggiorna, infine, i valori $x_{k+1} = x_k + t_k d_k$ e $k = k + 1$, e si torna al passo 2.
- Altrimenti, se $d_k = 0$, calcoliamo $\lambda = -(MM^T)^{-1}M\nabla f(x_k)$. Se $\lambda \geq 0$ ci fermiamo, altrimenti, calcoliamo

$$\lambda_j = \min_{i \in \mathcal{A}} \lambda_i \tag{15}$$

  si elimina da M la riga A_j w si torna al terzo passo.

Analizziamo passo passo l'algoritmo. La prima cosa da fare è scegliere un punto ammissibile x_k, per farlo è sufficiente prendere un punto tale per cui $Ax_k \leq b$.

Domanda orale

Come si trova un punto ammissibile di partenza ? *Si costruisce un algoritmo di duale ausiliario e risolvendolo si ottiene il nostro* ***punto ammissibile di partenza****.*

Successivamente, dobbiamo identificare i vincoli attivi. Intuitivamente, è come se stessimo cercando le direzioni in cui rischiamo di uscire dal poliedro, perché ci troviamo sul bordo. A tal fine, definiamo l'insieme degli indici dei vincoli attivi come:

$$\mathcal{A} = \{i : A_i x_k = b_i\} \tag{16}$$

A partire da questi indici costruiamo una matrice $M = \{A_i : i \in \mathcal{A}\}$. Calcoliamo adesso la **matrice di proiezione**

$$H = I - M(MM^T)^{-1}M \tag{17}$$

questa matrice ci permette di trovare una proiezione ortogonale della nostra direzione di massima discesa sulla regione ammissibile. Calcoliamo adesso la direzione di massima discesa d_k

$$d_k = -H\nabla f(x_k) \tag{18}$$

Se il punto è interno al poliedro abbiamo $\mathcal{A} = \emptyset$, di conseguenza

$$H = I \quad \text{e di conseguenza} \quad d_k = -\nabla f(x_k) \tag{19}$$

Il motivo del segno negativo è che la proiezione ortogonale del gradiente indica la direzione in cui la funzione cresce restando nella regione ammissibile. Poiché dobbiamo minimizzare la funzione, dobbiamo muoverci nella direzione opposta a quella del gradiente. Se il prodotto scalare tra il gradiente e la direzione di discesa è zero, significa che non c'è modo di muoversi lungo quella direzione. In questo caso dobbiamo verificare se il punto x_k è un **punto stazionario**, l'unico strumento che abbiamo per farlo è il **sistema LKKT**. Quindi, procediamo calcolandoci i moltiplicatori λ, definiti come

$$\lambda = -(MM^T)^{-1}M\nabla f(x_k) \tag{20}$$

Se tutti i componenti sono positivi allora abbiamo soddisfatto il sistema, abbiamo un punto **stazionario** e possiamo fermare l'iterazione.

Domanda orale

Chi mi garantisce che MM^T sia invertibile ? *Per definizione, il prodotto di una matrice per la sua trasposta, è sempre* ***invertibile****.*

Altrimenti prendiamo l'indice della componente minima

$$\begin{aligned} &\lambda_j = \min_{i \in \mathcal{A}} \lambda_i \\ &j = \{i \in \mathcal{A} : \lambda_j = \lambda_i\} \end{aligned} \tag{21}$$

Eliminiamo la riga j-esima dalla matrice M e si procede a ricalcolare la proiezione ortogonale. Se invece $d_k \neq 0$ dobbiamo calcolare il passo esatto, lo possiamo fare come in ogni altro metodo del gradiente appena studiato

$$t_k = \underset{0 \leq t_k \leq \hat{t}_k}{\arg\min} f(x_k + td_k) \tag{22}$$

Osservando attentamente notiamo la presenza di un fattore $\hat{t}_k$, questo coefficiente è il passo massimo, quindi la massima distanza che possiamo percorrere lungo la direzione d_k senza uscire dal poliedro. La possiamo ricavare risolvendo il seguente problema

$$\begin{cases} \max \hat{t}_k \\ A(x_k + \hat{t}_k d_k) \leq b \end{cases} \tag{23}$$

L'ultima cosa che rimane da fare è aggiornare i vari valori

$$\begin{aligned} &t_{k+1} = x_k + t_k d_k \\ &k = k + 1 \end{aligned} \tag{24}$$

Durante la spiegazione abbiamo usato due ragionamenti che sono sintetizzabili in un teorema

Theorem 2.2. Ad ogni passo k-esimo dell'algoritmo, valgono i seguenti fatti

- Se $d_k \neq 0$ allora d_k è una direzione di discesa per f.
- Se $d_k = 0$ e $\lambda \geq 0$, allora (x_k, λ) è una soluzione del sistema LLKT.

Infine, enunciamo l'ultimo teorema, analogo a tutti i teoremi già definiti in precedenza

Theorem 2.3. Supponiamo che la regione ammissibile sia limitata, oppure che la funzione f sia **coerciva**. Allora, comunque si scelga un punto iniziale x_0, il metodo del gradiente proiettato o trova un punto stazionario dopo un numero finito di iterazioni, oppure genera una successione $\{x_k\}$ tale che, ogni suo punto di accumulazione è un punto stazionario.

Parte V

Appendici

Appendice A

Richiami di Analisi II

In questo capitolo andremmo a richiamare i prerequisiti necessari di Analisi Matematica II necessari alla comprensione del paragrafo sulla **Programmazione non lineare**.

Una delle prime cose che occorre trattare è il concetto di continuità e derivabilità delle funzioni

1 Derivabilità e continuità delle funzioni

Siano dati un sottoinsieme $X \in \mathbb{R}^n$, un vettore $x \in X$ ed una funzione $f : \mathbb{R}^n \to \mathbb{R}$. Preso uno scalare $l \in \mathbb{R}$ tale che $l \in [-\infty, +\infty]$ diciamo che **f tende a l** quando **y tende a x** se e solo se, per ogni successione $\{x_k\} \subseteq X\backslash\{x\}$ che converge ad x la successione $\{f(x_k)\}$ converge verso l.

$$\lim_{y \to x} f(y) = l \iff \forall \{x_k\} \to x \subseteq X\backslash\{x\}, \quad \{f(x_k)\} \to l \tag{1}$$

Sfruttando quanto appena detto possiamo dare una definizione di continuità della funzione

Definition 1.1. (Continuità di una funzione) Data una funzione $f : \mathbb{R}^n \to \mathbb{R}$, un vettore $x \in \mathbb{R}^n$ e un vettore $y \in \mathbb{R}^n$, possiamo dire che la funzione è **continua in x** se e solo se

$$\lim_{y \to x} f(y) = f(x) \tag{2}$$

Sia $f(x_0, \ldots, x_n)$ una funzione in $\mathbb{R}^n$, si assuma di voler calcolare la funzione derivata rispetto a una variabile x_i, si definisce come ***derivata parziale*** rispetto a x_i

$$\frac{\partial f}{\partial x^i} f(x_0, \ldots, x_n) = \lim_{h \to 0} \frac{f(x_0, x_i + h, \ldots, x_n) - f(x_0, x_i, \ldots, x_n)}{h} \tag{3}$$

il limite del rapporto incrementale rispetto alla variabile x_i.

Quando si calcola una derivata parziale valgono tutte le regole della derivazione in $\mathbb{R}$, considerando però le restanti variabili come se fossero costanti. Si assuma di avere una funzione in due variabili

$$f(x, y) = 3x^2y^4 \tag{4}$$

nel caso si volesse definire la derivata rispetto alla variabile x, si procederebbe calcolando

$$\frac{\partial f}{\partial x} 3x^2y^4 = 6xy^4 \tag{5}$$

Se una funzione in $\mathbb{R}^n$ ha tutte le derivate definite per un punto x_0 si dirà che

*la funzione è **derivabile** in* x_0

analogamente, se una funzione ammette tutte le derivate parziali in un insieme di punti E si dirà che

*la funzione è **derivabile** in E*

Si assuma di avere un punto $(x_0, \ldots, x_n) \in D$ e si assuma di avere una funzione *derivabile in* x_0. Si definisce **gradiente della funzione** il vettore che ha per coordinate le derivate parziali rispetto a ogni variabile

$$\nabla f = \left(\frac{\partial f}{\partial x_0}, \frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_i} \right) \tag{6}$$

Calcolando il gradiente in un punto specifico, si possono ottenere informazioni sulla direzione di massima crescita della funzione in quel punto e su come questa crescita è influenzata dalle variazioni rispetto a x e a y. Se la componente lungo x è più grande di quella lungo y, la funzione cresce più rapidamente in direzione x; se la componente lungo y è più grande, la funzione cresce più rapidamente in direzione y.

L'ultimo costrutto teorico che definiamo è il concetto di matrice Hessiana. La matrice **hessiana** è la matrice $Hf(x) \in \mathbb{R}^{n \times n}$ costituita da tutte le derivate seconde; è una matrice con n^2 componenti (dove n indica la dimensione del dominio).

Definition 1.2 (Matrice Hessiana)**.** Sia $f : A \subseteq \mathbb{R}^n \to \mathbb{R}$ una funzione, si definisce come **matrice hessiana** $Hf(x)$ di f la matrice composta

$$Hf(x) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1^2}(x) & \frac{\partial f_1}{\partial x_1 \partial x_2}(x) & \ldots & \frac{\partial f_1}{\partial x_1 \partial x_n}(x) \\ \frac{\partial f_1}{\partial x_2 \partial x_1}(x) & \frac{\partial f_1}{\partial x_2^2}(x) & \ldots & \frac{\partial f_1}{\partial x_2 \partial x_n}(x) \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_1}{\partial x_1 \partial x_n}(x) & \frac{\partial f_1}{\partial x_2 \partial x_n}(x) & \ldots & \frac{\partial f_1}{\partial x_n^2}(x) \end{bmatrix} \tag{7}$$

La matrice **hessiana**, per teorema, è una **matrice simmetrica** e per tanto ha **autovalori reali**. Passiamo a dare una veloce carrellata di definizioni

Definition 1.3. (Punto stazionario) Data una funzione $f : \mathbb{R}^n \to \mathbb{R}$ e un punto $x_0 \in \mathbb{R}^n$. Si definisce x_0 come **punto stazionario** se e solo se

$$\nabla f(x_0) = 0 \tag{8}$$

Definition 1.4. (Vettori linearmente indipendenti) Siano $x_1, x_2, \ldots, x_m \in \mathbb{R}^n$ dei vettori definiti su uno spazio a n dimensioni, siano inoltre $a_1, a_2, \ldots, a_m \in \mathbb{R}$ dei coefficienti reali. Possiamo dire che i vettori sono **linearmente indipendenti** se l'unica combinazione lineare che vale 0 è ottenuta ponendo tutti i coefficienti $a_1, a_2, \ldots, a_n$ a 0.

$$\sum_{i=1}^{m} a_i x_i = 0 \iff a_i = 0 \quad \forall i = 1, \ldots, m \tag{9}$$

Definition 1.5. (Matrice definita e semi-definita) Sia data una matrice $A \in M^{n \times m}$; diciamo allora che

- Se A è **definita positiva**, il prodotto scalare rispetta la seguente condizione:

$$< A \cdot x, x > > 0 \quad \forall x \in \mathbb{R}^n \tag{10}$$

- Se A è **semi-definita positiva**, il prodotto scalare rispetta la seguente condizione:

$$< A \cdot x, x > \geq 0 \quad \forall x \in \mathbb{R}^n \tag{11}$$

- Se A è **definita negativa**, il prodotto scalare rispetta la seguente condizione:

$$< A \cdot x, x > < 0 \quad \forall x \in \mathbb{R}^n \tag{12}$$

- Se A è **semi-definita negativa**, il prodotto scalare rispetta la seguente condizione:

$$< A \cdot x, x > \leq 0 \quad \forall x \in \mathbb{R}^n \tag{13}$$

Possiamo anche utilizzare gli autovalori per definire se una matrice rispetta una di queste definizioni; supponiamo di calcolare gli autovalori di A

$$\det(A - \lambda I) \tag{14}$$

Se gli autovalori sono positivi la matrice è **definita positiva**, se oltre che positivi possono essere anche nulli, allora è **semi-definita positiva**. Invertendo il segno si ottengono anche le definizioni per matrice **definita negativa** e **semi-definita negativa**.

Una volta assunte queste definizioni, possiamo passare a enunciare le definizioni relative all'individuazione di minimi locali per funzioni *n-dimensionali*.

Theorem 1.1. (Teorema di Fermat) Data $f : \mathbb{R}^n \to \mathbb{R}$, $f \in C^\infty$ e dato un punto $\overline{x} \in \mathbb{R}^n$ di **minimo** locale, possiamo dire che $\overline{x}$ è anche un **punto stazionario**.

- Il fatto che sia un punto di **minimo** è condizione sufficiente perché $\overline{x}$ sia un punto stazionario; non è però condizione necessaria, in quanto anche i punti di sella e di massimo sono punti stazionari pur non essendo punti di minimo.
- Il fatto che $\overline{x}$ sia un punto stazionario è condizione necessaria affinché sia un sia anche un punto di **minimo**.

Theorem 1.2. Data $f : \mathbb{R}^n \to \mathbb{R}$, $f \in C^\infty$ se $\overline{x} \in \mathbb{R}^n$ è un punto di minimo locale di f (e quindi anche punto stazionario), allora possiamo dire che

$$Hf(\overline{x}) \geq 0 \tag{15}$$

la *matrice hessiana* è **semi-definita positiva**. Come anche nel caso precedente

- Se ho un punto di minimo $\overline{x}$ è automatico che $Hf(\overline{x}) \geq 0$.
- Se $Hf(\overline{x}) \geq 0$ non è detto che $\overline{x}$ sia un punto di minimo.

Theorem 1.3. Data $f : \mathbb{R}^n \to \mathbb{R}$, $f \in C^2$ e dato un punto $\overline{x}$, se $\nabla f(\overline{x}) = 0$ e $Hf(\overline{x}) > 0$ allora $\overline{x}$ è un **punto di minimo locale**.

Concludiamo la nostra analisi su massimi e minimi locali parlando dei teoremi per l'individuazione di massimi locali

Theorem 1.4. (Teorema di Fermat) Data $f : \mathbb{R}^n \to \mathbb{R}$, $f \in C^\infty$ e dato un punto $\overline{x} \in \mathbb{R}^n$; se $\overline{x}$ è un punto di massimo locale, allora è anche un **punto stazionario**.

Theorem 1.5. Data $f : \mathbb{R}^n \to \mathbb{R}$, $f \in C^\infty$ e dato un punto $\overline{x} \in \mathbb{R}^n$; se $\overline{x}$ è un punto di massimo locale di f e $\nabla f(\overline{x}) = 0$ allora

$$Hf(\overline{x}) \leq 0 \tag{16}$$

In questo caso la matrice è **semi-definita negativa**.

Theorem 1.6. Data $f : \mathbb{R}^n \to \mathbb{R}$, $f \in C^2$ e dato un punto $\overline{x}$, se $\nabla f(\overline{x}) = 0$ e $Hf(\overline{x}) < 0$ allora $\overline{x}$ è un **punto di massimo locale**.

Un altro concetto fondamentale è quello di **restrizione**. Questo concetto risulta particolarmente utile durante lo studio delle funzioni coercive

Definition 1.6. Data una funzione $f : \mathbb{R}^n \to \mathbb{R}$ definiamo come **restrizione della funzione** $\gamma : \mathbb{R} \to \mathbb{R}$ ottenuta restringendo il dominio della funzione f. Quella che utilizzeremo noi è la *restrizione a una semiretta*, dove scriviamo l'equazione della semiretta in formato parametrico e successivamente la sostituiamo alla funzione

Prendiamo un esempio pratico

$$f(x_1, x_2) = x_1^2 - x_2^2 \longleftrightarrow \begin{cases} x_1 = t \\ x_2 = 3 + t \end{cases} \iff \gamma(t) = t^2 - (3+t)^2 = -9 - 6t \tag{17}$$

Appendice B

Matlab

1 Direttiva *linprog*

Se volessimo automatizzare il processo di risoluzione di un problema di PL potremmo utilizzare un applicativo, come ad esempio Matlab. Matlab risolve problemi nella forma

$$\begin{cases} \min c^T x \\ Ax \leq b_1 \\ Aeqx = b_2 \\ lb \leq x \leq ub \end{cases} \tag{1}$$

La direttiva che permette di trovare la soluzione al problema su prende il nome di **LINPROG**; questa direttiva accetta 7 parametri

- Il vettore c che descrive il nostro modello.
- I vettori e le matrici che descrivono il poliedro
  - A e b sono, rispettivamente, la matrice e il vettore che descrivono il nostro poliedro risolvendo il sistema lineare
    $$Ax \leq b \tag{2}$$
  - Aeq e beq sono, rispettivamente, la matrice e il vettore che descrivono il nostro poliedro risolvendo il sistema lineare
    $$A_{eq}x = b_{eq} \tag{3}$$
- Un *lower-bound* delle variabili. Quindi una **limitazione inferiore** delle variabili.
- Un *upper-bound* delle variabili. Quindi una **limitazione superiore** delle variabili.

I parametri appena descritti si concretizzano in codice matlab come nell'esempio sottostante

```
c = [];
A = [];
Aeq = [];
b = [];
beq = [];
lb = [];
ub = [];
x = linprog(c,A, Aeq, b, beq, lb, ub);
```

Occorre fare una piccola trattazione sui vettori di *lower-bound*(**lb**) e *upper-bound*(**ub**):

- Se solo singole x sono limitate è sufficiente inserire i valori nella matrice e utilizzare uno spazio dove le x non sono limitate.

$$\begin{cases} x_1 \geq 0 \\ x_2 \geq 0 \\ x_3 \leq 8 \end{cases} \tag{4}$$

in linguaggio matlab avremmo

```
v_b = [ , , 8];
v_a = [0, 0 , ]
```

2 Direttiva *intlinprog*

Analoga alla direttiva linprog che utilizziamo nei problemi di PL, esiste anche una direttiva utilizzata per risolvere i problemi di PLI. Questa direttiva prende il nome di **intlinprog**. I parametri che dobbiamo specificare sono gli stessi della direttiva **linprog**, con l'aggiunta di un vettore **intcon** che indica gli indici delle variabili che vogliamo a coefficienti interi

```
c = [];
A = [];
Aeq = [];
b = [];
beq = [];
v_b = [];
v_a = [];
x = intlinprog(c,intcon, A, Aeq, b, beq, lb, ub);
```

3 Direttiva *quadprog*

Minimizziamo la funzione

$$\begin{cases} \min \frac{1}{2} xQx + c^T x \\ Ax \leq b \\ A_{\text{eq}} x = b_{\text{eq}} \\ lb \leq x \leq ub \end{cases} \tag{5}$$

Applichiamo la direttiva

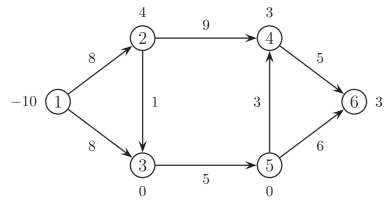
```
quadprog(Q, c, A, b, Aeq, beq, lb, ub)
```

Quadprog funziona bene per il minimo di una convessa e il massimo di una concava. Non è affidabile nel resto dei casi.

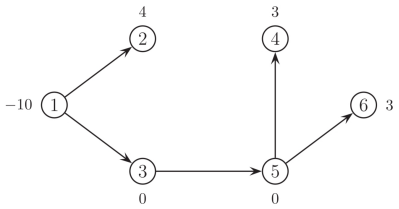
Appendice C

Calcolo del flusso di base

Dato il seguente grafo



Supponiamo di prendere un albero di copertura qualsiasi; nel nostro esempio avremmo $T = \{(1,2),(1,3),(3,5),(5,4),(5,6)\}$.

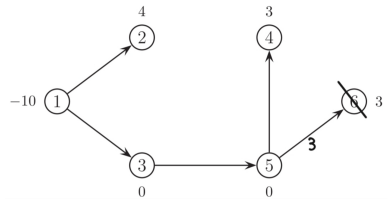


Quello che dobbiamo fare è determinare il flusso relativo al nostro albero di copertura.

Partiamo analizzando l'equazione di bilancio al nodo 6. Il bilancio del nodo è uguale a 3 e non ci sono archi uscenti, di conseguenza, affinché l'equazione venga soddisfatta, il flusso entrate deve essere uguale a 3 (la differenza tra flusso entrante e flusso uscente deve essere uguale al bilancio)

$$\overline{x} = \begin{bmatrix} (1,2) & (1,3) & (2,3) & (2,4) & (3,5) & (4,6) & (5,4) & (5,6) \\ & & 0 & 0 & & 0 & & 3 \end{bmatrix} \tag{1}$$

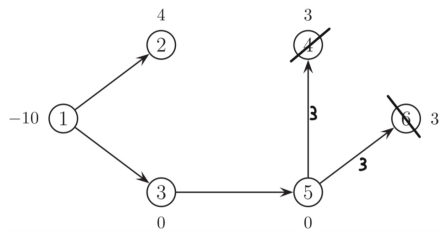
Possiamo quindi "*chiudere*" il nodo corrispettivo.



Passiamo ora all'equazione del nodo 4. Analogamente al nodo 6, non abbiamo alcun arco uscente e solo un arco entrante. Dunque, per far sì che l'equazione di bilancio venga rispettata, il flusso entrante sul nodo 4 deve essere uguale al bilancio, quindi 3.

$$\overline{x} = \begin{bmatrix} (1,2) & (1,3) & (2,3) & (2,4) & (3,5) & (4,6) & (5,4) & (5,6) \\ & & 0 & 0 & & 0 & 3 & 3 \end{bmatrix} \tag{2}$$

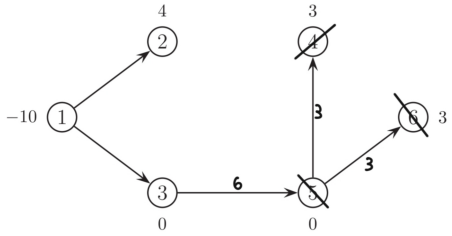
Possiamo, come nel caso precedente, "chiudere" il nodo 4.



Passiamo adesso ad analizzare il nodo 5. Il bilancio sul nodo è 0 e escono $3+3=6$ unità di flusso; di conseguenza, affinché l'equazione di bilancio sia rispettata è necessario che entrino 6 unità di flusso dal nodo 3

$$\overline{x} = \begin{bmatrix} (1,2) & (1,3) & (2,3) & (2,4) & (3,5) & (4,6) & (5,4) & (5,6) \\ & & 0 & 0 & 6 & 0 & 3 & 3 \end{bmatrix} \tag{3}$$

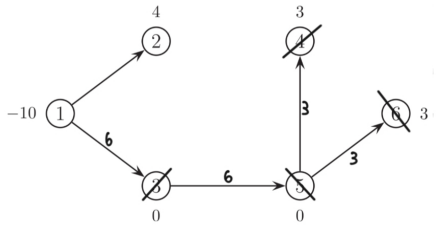
Possiamo, come nel caso precedente, "chiudere" il nodo 5.



Ripetiamo lo stesso procedimento anche per il nodo 3, dove abbiamo un bilancio pari a 0 e 6 unità di flusso uscente; affinché l'equazione di bilancio sia rispettata è necessario che entrino 6 unità di flusso dal nodo 1

$$\overline{x} = \begin{bmatrix} (1,2) & (1,3) & (2,3) & (2,4) & (3,5) & (4,6) & (5,4) & (5,6) \\ & 6 & 0 & 0 & 6 & 0 & 3 & 3 \end{bmatrix} \tag{4}$$

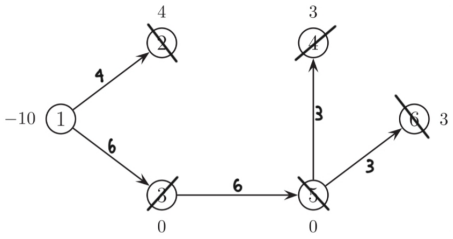
Possiamo, come nel caso precedente, "chiudere" il nodo 3. Analizziamo il nodo 2. Il bilancio sul



nodo 2 è 4, il flusso uscente è 0; di conseguenza, per rispettare l'equazione di bilancio, è sufficiente che il flusso entrante dal nodo 1 sia 4

$$\overline{x} = \begin{bmatrix} (1,2) & (1,3) & (2,3) & (2,4) & (3,5) & (4,6) & (5,4) & (5,6) \\ 4 & 6 & 0 & 0 & 6 & 0 & 3 & 3 \end{bmatrix} \tag{5}$$

Possiamo, come nel caso precedente, "chiudere" il nodo 2.



Siamo quindi riusciti ad ottenere un flusso di base ammissibile, senza calcoli matematici complessi, pari a:

$$\overline{x} = \begin{bmatrix} (1,2) & (1,3) & (2,3) & (2,4) & (3,5) & (4,6) & (5,4) & (5,6) \\ 4 & 6 & 0 & 0 & 6 & 0 & 3 & 3 \end{bmatrix} \tag{6}$$

Ovviamente, essendo il flusso relativo all'albero di copertura, il flusso sugli archi non di base è nullo.