

Problemi di ottimizzazione combinatoria



Indice

Perché l'ottimizzazione combinatoria?

Problemi di ottimizzazione combinatoria

Categorie di problemi di ottimizzazione combinatoria

Contare le soluzioni

Esplosione combinatoria e forza bruta

Bound primali, bound duali e gap

Cenni di complessità computazionale

Errori tipici

In sintesi

Obiettivi

Al termine di questo capitolo saremo in grado di:

- ▶ riconoscere un problema di ottimizzazione combinatoria, la categoria delle sue decisioni e l'oggetto combinatorio delle sue soluzioni (un sottoinsieme, un ordinamento, una partizione, un assegnamento);
- ▶ contare le soluzioni candidate (2^n sottoinsiemi, $n!$ ordinamenti, $(n-1)!/2$ ordinamenti ciclici, numeri di Bell per le partizioni) e stimare il tempo di calcolo di un'enumerazione a forza bruta;
- ▶ definire bound primali, bound duali e gap, e usarli per certificare che una soluzione è ottima;
- ▶ collocare un problema nelle classi P, NP e NP-hard, e distinguere un algoritmo pseudo-polinomiale da uno polinomiale.

Prerequisiti

- ▶ Dai *Preliminari*: problemi di ottimizzazione (regione ammissibile, funzione obiettivo, soluzione ottima e valore ottimo), coefficienti binomiali e partizioni di un insieme finito
- ▶ Non serve alcuna conoscenza di programmazione intera: i bound di questo capitolo vengono da argomenti combinatori



Perché l'ottimizzazione combinatoria?

Introduzione: Perché l'ottimizzazione combinatoria?

- ▶ Molte decisioni non scelgono un numero, ma un **oggetto discreto**: un insieme di progetti, un ordine di visita, un modo di suddividere oggetti fra contenitori
- ▶ In questa sezione vediamo dove nascono queste decisioni e perché contano nella pratica
- ▶ Incontriamo i sei problemi attorno a cui è costruito il corso, ciascuno attraverso una breve storia

Problemi di ottimizzazione: un richiamo

- ▶ Ricordiamo che, dati un insieme F e una funzione f definita su F , il **problema di ottimizzazione** P associato in forma di massimo (minimo) chiede di trovare un punto $\tilde{\mathbf{x}}(P) \in F$ tale che

$$f(\tilde{\mathbf{x}}(P)) \geq f(\mathbf{x}) \quad (\leq), \quad \forall \mathbf{x} \in F$$

- ▶ Il valore $z(P) = f(\tilde{\mathbf{x}}(P))$ è il **valore ottimo**
- ▶ In molti problemi decisionali le soluzioni ammissibili non sono punti di un continuo ma **oggetti discreti**:
 - ▶ un insieme di progetti da finanziare;
 - ▶ un ordine in cui visitare i clienti;
 - ▶ un modo di suddividere oggetti fra contenitori.

Dove nascono i problemi (1/4)

- I problemi di ottimizzazione combinatoria nascono ogni volta che le decisioni riguardano la **selezione**, l'**ordinamento**, il **raggruppamento** o l'**assegnamento** di oggetti discreti. Le principali aree applicative, con problemi tipici:

Area	Problemi tipici
Pianificazione della produzione	che cosa, quando e quanto produrre: soddisfacimento della domanda su più periodi, lot sizing , pianificazione della capacità, bilanciamento delle linee di assemblaggio
Scheduling	ordine e tempi dei lavori sulle macchine (makespan, ritardo), scheduling di progetti, scheduling della manutenzione

Dove nascono i problemi (2/4)

Area	Problemi tipici
Localizzazione di impianti	dove aprire magazzini, stabilimenti, servizi di emergenza, e quali clienti serve ciascuno
Packing	caricare oggetti su veicoli, container o pallet; selezionare il carico di maggior valore per un singolo container
Taglio	tagliare barre, lastre, rotoli o pannelli nei pezzi richiesti con il minimo sfrido (acciaio, vetro, tessuti, legname, carta)
Logistica	instradamento di veicoli, gestione della flotta, gestione delle scorte, progettazione della rete di distribuzione

Dove nascono i problemi (3/4)

Area	Problemi tipici
Trasporti	turnazione degli equipaggi, instradamento degli aerei, orari ferroviari, carico delle merci
Personale	pianificazione della forza lavoro e dei turni, turni degli infermieri, abbinamento degli equipaggi
Progettazione di reti	reti di telecomunicazione, energetiche, di trasporto e della catena di fornitura
Finanza	selezione di portafoglio con vincoli di cardinalità, replica di un indice, allocazione del capitale

Dove nascono i problemi (4/4)

Area	Problemi tipici
Telecomunicazioni	assegnamento delle frequenze, instradamento, espansione della capacità, allocazione della banda
Energia	unit commitment (quali centrali accendere), pianificazione della generazione, gestione della rete
Sanità	scheduling degli interventi chirurgici, pianificazione della radioterapia, scambio di reni e abbinamento degli organi, gestione dei posti letto
Altre aree	pianificazione delle colture, calendari sportivi, allocazione di macchine virtuali ai server, pianificazione dei pattugliamenti

Uno strumento per le decisioni

L'**ottimizzazione combinatoria** è uno strumento fondamentale per le decisioni.

I pacchi che riceviamo, l'elettricità che usiamo e i voli che prendiamo sono pianificati risolvendo problemi di ottimizzazione combinatoria.

I sei problemi del corso

Il corso è costruito attorno a **sei problemi**.

Ciascuno è introdotto qui con una breve storia;
le definizioni formali e le varianti arrivano più avanti nel corso.

Il problema dello zaino 0/1

- ▶ Un'azienda ha un budget di c euro e n progetti candidati; il progetto j costa w_j e rende un profitto p_j
- ▶ Quali progetti finanziare?

Il problema del bin packing

- ▶ Pacchi di pesi diversi devono essere spediti con camion identici di capacità c
- ▶ Quanti camion servono?
- ▶ Lo stesso problema si presenta quando si collocano macchine virtuali su server identici

Il problema della colorazione dei vertici

- ▶ Sei esami devono essere fissati nella sessione d'esame. Due esami che hanno almeno uno studente in comune non possono svolgersi nello stesso giorno
- ▶ Qual è il numero minimo di giorni?

Il problema del commesso viaggiatore

- ▶ Un corriere parte dal deposito, deve visitare $n - 1$ clienti una volta ciascuno e tornare
- ▶ Qual è il giro più corto?

Scheduling su macchine parallele identiche

- ▶ Sette lavori di stampa devono essere inviati a tre stampanti identiche. Ogni lavoro gira su una stampante senza interruzioni
- ▶ Come distribuire i lavori in modo che finiscano tutti il prima possibile?

Il problema di lot sizing capacitato

- ▶ Una fabbrica conosce la domanda del suo prodotto per i prossimi m periodi (settimane)
- ▶ Produrre in un periodo richiede un **setup** (un costo fisso per preparare la linea) ed è limitato dalla capacità della linea; ciò che si produce in anticipo resta a scorta, con un costo di mantenimento
- ▶ In quali periodi produrre, e quanto?

Decisioni discrete:

- ▶ Le soluzioni ammissibili sono oggetti discreti: un insieme di progetti, un ordine di visita, una suddivisione di oggetti fra contenitori

Applicazioni:

- ▶ Selezionare, ordinare, raggruppare e assegnare: produzione, scheduling, localizzazione, packing, taglio, logistica, trasporti, personale, reti, finanza, telecomunicazioni, energia, sanità

Sei problemi:

- ▶ Zaino, bin packing, colorazione dei vertici, commesso viaggiatore, scheduling su macchine parallele, lot sizing

Problemi di ottimizzazione combinatoria



Introduzione: Problemi di ottimizzazione combinatoria

- ▶ In questa sezione diamo la definizione formale di **problema di ottimizzazione combinatoria**: un insieme base, una famiglia di soluzioni ammissibili e una funzione obiettivo
- ▶ Distinguiamo un **problema** dalle sue **istanze**, e introduciamo una piccola istanza che usiamo in tutto il capitolo
- ▶ Elenchiamo le caratteristiche principali di questi problemi: facili da enunciare, spesso molto difficili da risolvere

Definizione 1: problema di ottimizzazione combinatoria

Dati:

1. un insieme finito $\mathcal{U} = \{1, 2, \dots, n\}$, detto **insieme base**,
2. una famiglia $\mathcal{F} \subseteq 2^{\mathcal{U}}$ di sottoinsiemi di $\mathcal{U}$, detti **soluzioni ammissibili**,
3. una **funzione obiettivo** $f : \mathcal{F} \rightarrow \mathbb{R}$,

il **problema di ottimizzazione combinatoria** associato chiede di trovare una soluzione ammissibile $\tilde{S} \in \mathcal{F}$ tale che

$$f(\tilde{S}) \geq f(S), \quad \forall S \in \mathcal{F} \quad (\text{forma di massimo}),$$

oppure tale che

$$f(\tilde{S}) \leq f(S), \quad \forall S \in \mathcal{F} \quad (\text{forma di minimo}).$$

Regione ammissibile e funzione obiettivo

- ▶ La regione ammissibile $\mathcal{F}$ è un insieme **finito**, poiché $|2^{\mathcal{U}}| = 2^n$
- ▶ Di conseguenza, una soluzione ottima esiste sempre quando $\mathcal{F} \neq \emptyset$
- ▶ Nella maggior parte dei problemi la funzione obiettivo è **lineare** (additiva): ogni elemento $j \in \mathcal{U}$ ha un peso c_j e

$$f(S) = \sum_{j \in S} c_j$$

- ▶ Alcuni problemi hanno un obiettivo diverso, per esempio la più grande fra diverse somme, che è un massimo e non una somma

Problemi e istanze

- ▶ Un **problema** è una domanda generale con dati simbolici
- ▶ Un'istanza I del problema si ottiene fissando i numeri
- ▶ Indichiamo con $z(I)$ il valore ottimo dell'istanza I

Fissiamo ora i numeri di una piccola istanza:
la useremo in tutto il capitolo.

Esempio

- ▶ Consideriamo l'insieme base $\mathcal{U} = \{1, 2, 3, 4\}$, i pesi $\mathbf{c} = (7, 6, 4, 3)$ e l'obiettivo additivo $f(S) = \sum_{j \in S} c_j$, da massimizzare
- ▶ Le soluzioni ammissibili sono gli 11 sottoinsiemi

$$\mathcal{F} = \{\emptyset, \{1\}, \{2\}, \{3\}, \{4\}, \{1, 3\}, \{1, 4\}, \\ \{2, 3\}, \{2, 4\}, \{3, 4\}, \{2, 3, 4\}\}$$

- ▶ Per esempio, $f(\{1, 3\}) = 7 + 4 = 11$ e $f(\{2, 3, 4\}) = 6 + 4 + 3 = 13$, mentre il sottoinsieme $\{1, 2\}$ non è ammissibile
- ▶ La chiamiamo la **piccola istanza**

Come sono date le soluzioni ammissibili

- ▶ Nella piccola istanza $\mathcal{F}$ è **elencata**
- ▶ Nei problemi del corso non è mai elencata: è descritta da una **regola**, per esempio un vincolo di capacità
- ▶ La famiglia $\mathcal{F}$ è troppo grande per essere scritta

Che cos'è l'ottimizzazione combinatoria

L'**ottimizzazione combinatoria** studia i problemi di ottimizzazione su **strutture discrete** e **oggetti combinatori**, come grafi, sottoinsiemi, permutazioni, sequenze, partizioni, coperture e impaccamenti.

Caratteristiche principali

Le **caratteristiche principali** dei problemi di ottimizzazione combinatoria sono:

1. l'insieme delle soluzioni ammissibili è **finito**;
2. il numero di soluzioni ammissibili è tipicamente **enorme**: cresce in modo esponenziale (o più rapido) con la dimensione dell'istanza;
3. trovare una soluzione ottima può essere **computazionalmente molto difficile** (si vedano i cenni di complessità computazionale più avanti in questo capitolo);
4. servono algoritmi specializzati per esplorare lo spazio delle soluzioni senza enumerarlo.

Riepilogo

Definizione:

- ▶ Un insieme base finito $\mathcal{U}$, una famiglia di soluzioni ammissibili $\mathcal{F} \subseteq 2^{\mathcal{U}}$, una funzione obiettivo f , in forma di massimo o di minimo
- ▶ Una soluzione ottima esiste quando $\mathcal{F} \neq \emptyset$; l'obiettivo è di solito additivo

Problemi e istanze:

- ▶ Un'istanza I fissa i numeri; $z(I)$ è il suo valore ottimo
- ▶ La piccola istanza: $\mathcal{U} = \{1, 2, 3, 4\}$, $\mathbf{c} = (7, 6, 4, 3)$, 11 sottoinsiemi ammissibili

Caratteristiche:

- ▶ Regione ammissibile finita ma enorme: servono algoritmi specializzati

Categorie di problemi di ottimizzazione combinatoria

Introduzione: Categorie di problemi di ottimizzazione combinatoria

- ▶ Le decisioni di un problema di ottimizzazione combinatoria rientrano in cinque **categorie**
- ▶ A ogni categoria sono associati un **oggetto combinatorio**, che è la forma di una soluzione, e un tipo naturale di **variabili decisionali**
- ▶ Vediamo poi le principali aree applicative e collochiamo i problemi del corso nelle loro categorie

Categorie di decisioni (1/3)

1. **IN/OUT (inclusione/esclusione).** Ogni elemento è selezionato oppure no: una soluzione è un **sottoinsieme** degli elementi, descritto da variabili binarie $x_j \in \{0, 1\}$, con $x_j = 1$ se e solo se l'elemento j è selezionato.
 - ▶ Esempi: selezionare i progetti da finanziare con un budget limitato, scegliere i prodotti di un portafoglio, decidere quali clienti visitare in un percorso.
2. **Conteggio (quantità discrete).** La decisione è quante unità di ogni elemento usare: una soluzione è un vettore di **quantità intere**, descritto da variabili intere $x_j \in \mathbb{Z}_{\geq 0}$.
 - ▶ Esempi: quante unità di un prodotto produrre, quanti operatori assegnare a un turno, quanti server attivare in un data center.

Categorie di decisioni (2/3)

3. **Assegnamento.** Ogni elemento è assegnato a esattamente una fra diverse risorse distinte: una soluzione è una **funzione** dagli elementi alle risorse, descritta da variabili binarie $x_{ij} \in \{0, 1\}$, uguali a 1 se e solo se l'elemento j è assegnato alla risorsa i .
- ▶ Esempi: assegnare compiti a macchine o persone, abbinare studenti a progetti, allocare veicoli ai percorsi di consegna.
4. **Ordinamento e sequenze.** La decisione è l'ordine in cui gli elementi sono lavorati o visitati: una soluzione è una **permutazione** $\sigma = (\sigma_1, \dots, \sigma_n)$ degli elementi, o un ordinamento ciclico per un giro.
- ▶ Esempi: l'ordine in cui sono lavorati i lavori, la sequenza delle operazioni su una macchina, l'ordine di visita dei clienti.

5. **Configurazione e partizionamento.** Gli elementi sono suddivisi in gruppi, ciascuno dei quali soddisfa una condizione come una capacità: una soluzione è una **partizione** degli elementi in gruppi intercambiabili.
- ▶ Esempi: suddividere un insieme di elementi in gruppi, impaccare oggetti in contenitori, tagliare materiali minimizzando lo sfrido.

Raccogliamo le cinque categorie in una tabella.

Categorie, oggetti e variabili

Categoria	Oggetto combinatorio	Variabili naturali
IN/OUT	sottoinsieme	$x_j \in \{0, 1\}$
conteggio	vettore di quantità intere	$x_j \in \mathbb{Z}_{\geq 0}$
assegnamento	funzione (classi etichettate)	$x_{ij} \in \{0, 1\}$, una risorsa i per ogni j
ordinamento e sequenze	permutazione, ordinamento ciclico	$x_{ij} \in \{0, 1\}$, uguale a 1 se j segue i
configurazione e partizionamento	partizione (classi non etichettate)	$x_{ij} \in \{0, 1\}$, elemento j nel gruppo i

Leggere la tabella

- ▶ La tabella riassume le cinque categorie, i loro oggetti combinatori e le loro variabili naturali
- ▶ Con un insieme base opportuno ogni categoria rientra nella definizione di problema di ottimizzazione combinatoria:
 - ▶ per un assegnamento l'insieme base è l'insieme delle coppie (elemento, risorsa);
 - ▶ per un ordinamento l'insieme delle coppie di elementi consecutivi.
- ▶ Assegnamento e partizionamento differiscono solo nelle risorse: risorse distinte danno classi etichettate, risorse identiche danno gruppi intercambiabili (li contiamo nella prossima sezione)

Il ruolo della categoria

La **categoria** delle decisioni determina
l'**oggetto combinatorio** delle soluzioni,
il **numero di soluzioni candidate**
e le **variabili decisionali** naturali.

Aree applicative (1/3)

- I problemi di ottimizzazione combinatoria nascono in molte aree applicative. Le principali, con le loro decisioni tipiche e alcuni esempi, sono:

Area	Decisioni tipiche	Esempi
pianificazione della produzione	che cosa, quando e quanto produrre, con quali risorse	lot sizing, pianificazione della capacità, bilanciamento delle linee di assemblaggio
scheduling	ordine e tempi delle operazioni sulle risorse	scheduling dei lavori sulle macchine, scheduling di progetti, scheduling della manutenzione

Aree applicative (2/3)

Area	Decisioni tipiche	Esempi
localizzazione di impianti	dove aprire gli impianti e quali clienti serve ciascuno	localizzazione di magazzini, di stabilimenti, di servizi di emergenza
packing	come sistemare oggetti in contenitori	selezione del carico, carico di più veicoli, carico dei pallet
taglio	come tagliare il materiale grezzo nei pezzi richiesti	taglio di barre e lastre, minimizzazione dello sfrido, taglio a ghigliottina
logistica e catena di fornitura	percorsi, flotte e livelli di scorta	instradamento di veicoli, gestione della flotta, gestione delle scorte

Aree applicative (3/3)

Area	Decisioni tipiche	Esempi
trasporti	equipaggi, veicoli e orari di un sistema di mobilità	turnazione degli equipaggi, instradamento degli aerei, orari ferroviari
gestione del personale	assegnamento di persone a compiti, turni e ruoli	pianificazione della forza lavoro, turni degli infermieri, pianificazione dei turni
progettazione di reti	quali nodi e archi di una rete attivare	reti di telecomunicazione, energetiche e di trasporto

Combinare le categorie

- ▶ Un problema reale combina spesso diverse categorie
- ▶ Per esempio, la **localizzazione di impianti**:
 - ▶ decide quali impianti aprire (IN/OUT);
 - ▶ assegna ogni cliente a un impianto aperto (assegnamento).

Collochiamo ora i problemi del corso nelle loro categorie.

I problemi del corso per categoria (1/3)

La tabella colloca i problemi del corso nelle loro categorie; ogni nome rimanda alla sua pagina.

Problema	Soluzione	Categoria
problema dello zaino 0/1 (KP)	sottoinsieme di oggetti	IN/OUT
problema del subset sum (SSP)	sottoinsieme di oggetti	IN/OUT
problema dello zaino limitato (BKP)	numero di copie di ogni oggetto	conteggio
problema dello zaino illimitato (UKP)	numero di copie di ogni oggetto	conteggio
problema dello zaino multidimensionale (MDKP)	sottoinsieme di oggetti	IN/OUT

I problemi del corso per categoria (2/3)

Problema	Soluzione	Categoria
problema degli zaini multipli (MKP)	ogni oggetto in uno zaino o in nessuno	assegnamento
problema dello zaino a scelta multipla (MCKP)	un oggetto per classe	IN/OUT
problema dello zaino con setup (KPS)	sottoinsieme di classi e di oggetti	IN/OUT
problema del bin packing (BPP)	partizione degli oggetti nei contenitori	partizionamento
problema del cutting stock (CSP)	numero di rotoli tagliati con ogni schema	configurazione, conteggio
problema della colorazione dei vertici (VCP)	partizione dei vertici in insiemi stabili	partizionamento

I problemi del corso per categoria (3/3)

Problema	Soluzione	Categoria
problema della clique massima (MCP)	sottoinsieme di vertici	IN/OUT
problema dell'insieme stabile massimo (MSSP)	sottoinsieme di vertici	IN/OUT
problema del commesso viaggiatore (TSP)	ordinamento ciclico delle città	ordinamento
problema del commesso viaggiatore asimmetrico (ATSP)	ordinamento delle città su un giro	ordinamento
scheduling su macchine parallele identiche ($P \parallel C_{\max}$)	assegnamento dei lavori alle macchine	assegnamento
problema di lot sizing capacitato (CLS)	periodi di setup e quantità prodotte	IN/OUT, conteggio

Riepilogo

Cinque categorie di decisioni:

- ▶ IN/OUT (sottoinsieme), conteggio (quantità intere), assegnamento (funzione), ordinamento (permutazione, ordinamento ciclico), configurazione e partizionamento (partizione)
- ▶ La categoria dà l'oggetto combinatorio, il numero di candidate e le variabili naturali

Assegnamento e partizionamento:

- ▶ Risorse distinte danno classi etichettate, risorse identiche danno gruppi intercambiabili

Applicazioni e problemi:

- ▶ Un problema reale combina spesso diverse categorie, come nella localizzazione di impianti

Contare le soluzioni



Introduzione: Contare le soluzioni

- ▶ Prima di risolvere un problema ci chiediamo **quante** soluzioni candidate ha
- ▶ La risposta dipende solo dall'oggetto combinatorio: sottoinsiemi, ordinamenti o partizioni
- ▶ Questi numeri ci dicono fin dove può arrivare un'enumerazione completa, e perché diventa presto impossibile

Proposizione 1

Un insieme con n elementi ha 2^n sottoinsiemi.

- ▶ La dimostrazione identifica ogni sottoinsieme con un vettore binario

Dimostrazione

- ▶ Un sottoinsieme $S \subseteq \{1, 2, \dots, n\}$ è identificato dal vettore binario $\mathbf{x} \in \{0, 1\}^n$ con $x_j = 1$ se e solo se $j \in S$
- ▶ Ognuna delle n componenti assume 2 valori in modo indipendente, quindi i vettori sono

$$2 \cdot 2 \cdots 2 = 2^n$$

- ▶ Equivalentemente, il numero di sottoinsiemi con k elementi è $\binom{n}{k}$, e abbiamo:

$$\sum_{k=0}^n \binom{n}{k} = 2^n$$



Sottoinsiemi e ipercubo unitario

- ▶ Geometricamente, i 2^n vettori binari sono i vertici dell'ipercubo unitario $[0, 1]^n$:
 - ▶ il quadrato per $n = 2$ (4 vertici);
 - ▶ il cubo per $n = 3$ (8 vertici).
- ▶ Ogni decisione IN/OUT su n elementi è un sottoinsieme: i 2^n sottoinsiemi sono le sue soluzioni candidate

Ordinamenti e ordinamenti ciclici

- Gli n elementi di un insieme si possono ordinare in

$$n! = n \cdot (n - 1) \cdots 2 \cdot 1$$

modi (**permutazioni**)

- Un **ordinamento ciclico** dispone gli n elementi lungo un cerchio, come in un giro attraverso n luoghi: l'elemento di partenza e il verso di percorrenza non contano

Contare gli ordinamenti ciclici

Proposizione 2

Per $n \geq 3$ elementi, il numero di ordinamenti ciclici distinti, dove una sequenza, le sue rotazioni e la sua inversa sono lo stesso ordinamento ciclico, è

$$\frac{(n-1)!}{2}.$$

- La dimostrazione fissa l'elemento di partenza e poi toglie il doppio conteggio dovuto al verso

Dimostrazione

- ▶ Poiché l'elemento di partenza non cambia un ordinamento ciclico, possiamo partire sempre dall'elemento 1
- ▶ I restanti $n - 1$ elementi si possono ordinare in $(n - 1)!$ modi
- ▶ Ogni ordinamento ciclico si ottiene due volte, una per ogni verso, per esempio

$$(1, 3, 5, 2, 4) \quad \text{e} \quad (1, 4, 2, 5, 3)$$

- ▶ Quindi il numero di ordinamenti ciclici distinti è $(n - 1)!/2$



Esempio

- ▶ Per $n = 4$ ci sono $3!/2 = 3$ ordinamenti ciclici:

$$(1, 2, 3, 4), \quad (1, 2, 4, 3), \quad (1, 3, 2, 4)$$

- ▶ Ogni altra sequenza è una rotazione o un'inversa di uno di essi
- ▶ Per esempio, $(2, 1, 4, 3)$ è l'inversa di $(1, 2, 3, 4)$ fatta partire dall'elemento 2

Partizioni

- ▶ Una **partizione** di un insieme finito A è una collezione di sottoinsiemi non vuoti e a due a due disgiunti di A la cui unione è A
- ▶ Le decisioni di raggruppamento, come la suddivisione di oggetti in gruppi, hanno partizioni come soluzioni

Quante partizioni ha un insieme con n elementi?

Numeri di Bell

- ▶ Il numero di partizioni di un insieme con n elementi è il **numero di Bell** $B(n)$, definito dalla ricorsione:

$$B(0) = 1, \quad B(n) = \sum_{k=0}^{n-1} \binom{n-1}{k} B(k), \quad \forall n \geq 1$$

- ▶ La ricorsione conta le partizioni in base al blocco che contiene l'elemento n :

- ▶ se questo blocco ha altri $n - 1 - k$ elementi, essi si possono scegliere in

$$\binom{n-1}{n-1-k} = \binom{n-1}{k}$$

modi;

- ▶ i restanti k elementi si partizionano in $B(k)$ modi.

I primi numeri di Bell

- ▶ I primi numeri di Bell sono:

n	0	1	2	3	4	5	6	7	8	9	10
$B(n)$	1	1	2	5	15	52	203	877	4140	21147	115975

- ▶ Come 2^n e $(n-1)!/2$, i numeri di Bell crescono molto rapidamente con n

Assegnamenti ed etichette intercambiabili

- ▶ Un **assegnamento** di n elementi a m etichette è una funzione dagli elementi alle etichette: ce ne sono m^n
- ▶ Se le etichette sono intercambiabili (risorse identiche), due assegnamenti che differiscono solo per una ridenominazione delle etichette sono la stessa soluzione
- ▶ Le soluzioni distinte sono le **partizioni** degli elementi in al più m blocchi

Esempio

Per un insieme con $n = 5$ elementi, i numeri degli oggetti combinatori sono:

oggetto combinatorio	numero
sottoinsiemi	$2^5 = 32$
ordinamenti (permutazioni)	$5! = 120$
ordinamenti ciclici	$4!/2 = 12$
partizioni	$B(5) = 52$
assegnamenti a 3 etichette	$3^5 = 243$
partizioni in al più 3 blocchi	$1 + 15 + 25 = 41$

- L'ultima riga conta le partizioni di 5 elementi in 1, 2 o 3 blocchi (rispettivamente 1, 15 e 25); le altre 10 + 1 partizioni delle 52 hanno 4 o 5 blocchi

Soluzioni candidate e ammissibili

- ▶ Questi numeri contano le soluzioni **candidate**
- ▶ Quelle **ammissibili** possono essere molte meno: nella piccola istanza solo 11 dei 16 sottoinsiemi sono ammissibili

Il numero di candidate è il lavoro di un'enumerazione completa:
è l'argomento della prossima sezione.

Formule di conteggio:

- ▶ 2^n sottoinsiemi di n elementi (i vertici dell'ipercubo unitario)
- ▶ $n!$ ordinamenti e, per $n \geq 3$, $(n-1)!/2$ ordinamenti ciclici
- ▶ $B(n)$ partizioni, calcolate con la ricorsione dei numeri di Bell
- ▶ m^n assegnamenti a m etichette; con etichette intercambiabili, le partizioni in al più m blocchi

Soluzioni candidate e ammissibili:

- ▶ Le formule contano le candidate: le soluzioni ammissibili possono essere molte meno

Esplosione combinatoria e forza bruta

Introduzione: Esplosione combinatoria e forza bruta

- ▶ Una regione ammissibile finita si può sempre esplorare per intero: è l'algoritmo a **forza bruta**
- ▶ Lo descriviamo e lo eseguiamo sulla piccola istanza
- ▶ Poi misuriamo quanto rapidamente cresce il numero di candidate, e quanto dura un'enumerazione completa su un calcolatore: è l'**esplosione combinatoria**

Enumerazione completa

- ▶ Poiché la regione ammissibile è finita, un problema di ottimizzazione combinatoria si può sempre risolvere per **enumerazione completa**:
 - ▶ elenchiamo tutte le soluzioni candidate;
 - ▶ scartiamo quelle non ammissibili;
 - ▶ teniamo la migliore.
- ▶ Questo algoritmo si chiama **forza bruta**

L'algoritmo a forza bruta

Input: un'istanza di un problema di ottimizzazione combinatoria in forma di massimo.

Output: una soluzione ottima e il valore ottimo.

Passo 0: Inizializzazione. Porre $\bar{S} \leftarrow$ nessuna e $LB \leftarrow -\infty$.

Passo 1: Enumerazione. Per ogni soluzione candidata S (sottoinsieme, ordinamento, partizione):

- (a) se S non è ammissibile, scartarla;
- (b) se S è ammissibile e $f(S) > LB$, porre $\bar{S} \leftarrow S$ e $LB \leftarrow f(S)$.

Passo 2: Output. Restituire $\tilde{S} = \bar{S}$ e $z(I) = LB$. STOP

Esempio (1/2)

Per la piccola istanza i $2^4 = 16$ sottoinsiemi di $\mathcal{U}$ sono:

S	$f(S)$	ammissibile	S	$f(S)$	ammissibile
$\emptyset$	0	sì	$\{2, 3\}$	10	sì
$\{1\}$	7	sì	$\{2, 4\}$	9	sì
$\{2\}$	6	sì	$\{3, 4\}$	7	sì
$\{3\}$	4	sì	$\{1, 2, 3\}$	17	no
$\{4\}$	3	sì	$\{1, 2, 4\}$	16	no
$\{1, 2\}$	13	no	$\{1, 3, 4\}$	14	no
$\{1, 3\}$	11	sì	$\{2, 3, 4\}$	13	sì
$\{1, 4\}$	10	sì	$\{1, 2, 3, 4\}$	20	no

Esempio (2/2)

- Fra gli 11 sottoinsiemi ammissibili, il migliore è

$$\tilde{S} = \{2, 3, 4\}, \quad z(I) = 13$$

- È l'unica soluzione ottima: anche il sottoinsieme $\{1, 2\}$ ha valore 13, ma non è ammissibile

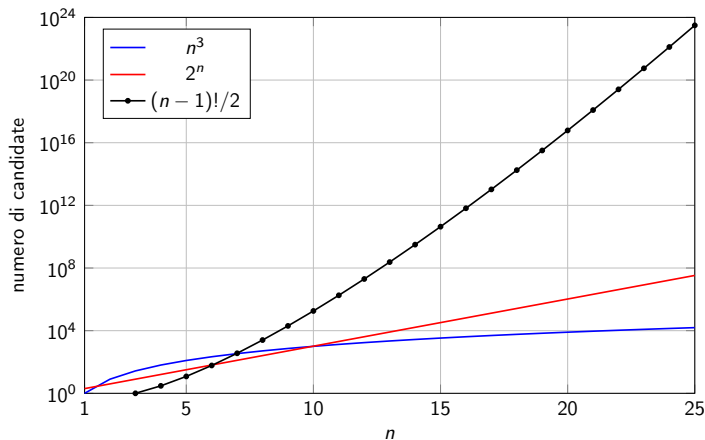
La forza bruta funziona per questa piccola istanza.
Come cresce il numero di candidate con n ?

Crescita esponenziale

- Il numero di candidate cresce molto rapidamente con la dimensione n dell'istanza. La tabella confronta la crescita polinomiale e quella esponenziale:

n	n^2	n^3	2^n	$(n-1)!/2$	$B(n)$
5	25	125	32	12	52
10	100	1 000	1 024	181 440	115 975
15	225	3 375	32 768	$\approx 4.36 \cdot 10^{10}$	$\approx 1.38 \cdot 10^9$
20	400	8 000	1 048 576	$\approx 6.08 \cdot 10^{16}$	$\approx 5.17 \cdot 10^{13}$
25	625	15 625	$\approx 3.36 \cdot 10^7$	$\approx 3.10 \cdot 10^{23}$	$\approx 4.64 \cdot 10^{18}$
30	900	27 000	$\approx 1.07 \cdot 10^9$	$\approx 4.42 \cdot 10^{30}$	$\approx 8.47 \cdot 10^{23}$

Crescita esponenziale: scala logaritmica



Tempi di calcolo: sottoinsiemi

- Supponiamo che un calcolatore valuti 10^9 soluzioni candidate al secondo, e calcoliamo il tempo di un'enumerazione completa (un anno = 365.25 giorni):

n	2^n	tempo a 10^9 al secondo	tempo a 10^{12} al secondo
10	1 024	≈ 1.02 microsecondi	≈ 1.02 nanosecondi
20	$\approx 1.05 \cdot 10^6$	≈ 1.05 millisecondi	≈ 1.05 microsecondi
30	$\approx 1.07 \cdot 10^9$	≈ 1.07 secondi	≈ 1.07 millisecondi
40	$\approx 1.10 \cdot 10^{12}$	≈ 18.3 minuti	≈ 1.10 secondi
50	$\approx 1.13 \cdot 10^{15}$	≈ 13.0 giorni	≈ 18.8 minuti
60	$\approx 1.15 \cdot 10^{18}$	≈ 36.5 anni	≈ 13.3 giorni
70	$\approx 1.18 \cdot 10^{21}$	$\approx 3.74 \cdot 10^4$ anni	≈ 37.4 anni
80	$\approx 1.21 \cdot 10^{24}$	$\approx 3.83 \cdot 10^7$ anni	$\approx 3.83 \cdot 10^4$ anni

Tempi di calcolo: ordinamenti ciclici e partizioni

ordinamenti ciclici: $(n - 1)!/2$			
n	numero	10^9 al secondo	10^{12} al secondo
10	181 440	≈ 181 microsecondi	≈ 181 nanosecondi
15	$\approx 4.36 \cdot 10^{10}$	≈ 43.6 secondi	≈ 43.6 millisecondi
20	$\approx 6.08 \cdot 10^{16}$	≈ 1.93 anni	≈ 16.9 ore
25	$\approx 3.10 \cdot 10^{23}$	$\approx 9.83 \cdot 10^6$ anni	$\approx 9.83 \cdot 10^3$ anni

partizioni: $B(n)$		
n	numero	10^9 al secondo
10	115 975	≈ 116 microsecondi
15	$\approx 1.38 \cdot 10^9$	≈ 1.38 secondi
20	$\approx 5.17 \cdot 10^{13}$	≈ 14.4 ore
25	$\approx 4.64 \cdot 10^{18}$	≈ 147 anni

Un calcolatore 1000 volte più veloce

- ▶ Un calcolatore 1000 volte più veloce non cambia il quadro
- ▶ **Sottoinsiemi:** poiché $2^{10} = 1024 > 1000$, nello stesso tempo enumera i sottoinsiemi di soli 10 elementi in più, circa
- ▶ **Ordinamenti ciclici:** il guadagno è ancora minore
 - ▶ passare da n a $n + 1$ elementi moltiplica il loro numero per n ;
 - ▶ $15 \cdot 16 \cdot 17 = 4080 > 1000$, quindi da 15 elementi in su il calcolatore più veloce guadagna meno di 3 elementi.

L'enumerazione non basta

- ▶ L'enumerazione completa è possibile solo per **istanze molto piccole**
- ▶ Calcolatori più veloci non aiutano: un numero esponenziale di candidate vanifica qualunque accelerazione costante
- ▶ Servono approcci più intelligenti, che **escludono** intere parti dello spazio delle soluzioni senza esplorarle

Forza bruta:

- ▶ Elenca tutte le candidate, scarta quelle non ammissibili, tiene la migliore: è esatta
- ▶ Sulla piccola istanza esamina 16 sottoinsiemi e trova $\tilde{S} = \{2, 3, 4\}$, $z(I) = 13$

Esplosione combinatoria:

- ▶ 2^n , $(n-1)!/2$ e $B(n)$ crescono molto più rapidamente di qualunque polinomio
- ▶ A 10^9 candidate al secondo, 2^{60} sottoinsiemi richiedono circa 36.5 anni
- ▶ Un'accelerazione costante fa guadagnare solo pochi elementi

Bound primali, bound duali e gap

Introduzione: Bound primali, bound duali e gap

- ▶ Gli approcci più intelligenti si basano su un'idea semplice: invece di calcolare direttamente $z(I)$, lo **racchiudiamo** fra due numeri
- ▶ Uno viene da una soluzione ammissibile, l'altro da un argomento che vale per *tutte* le soluzioni ammissibili
- ▶ La distanza fra i due numeri, il **gap**, misura quanto siamo lontani da una prova di ottimalità

Definizione 2: bound primali e duali

Consideriamo un'istanza I di un problema di ottimizzazione combinatoria con soluzioni ammissibili $\mathcal{F}$, funzione obiettivo f e valore ottimo $z(I)$.

1. Un **bound primale** è il valore $f(\bar{S})$ di una soluzione ammissibile $\bar{S} \in \mathcal{F}$.
2. Un **bound duale** è un numero β tale che

$$f(S) \leq \beta, \quad \forall S \in \mathcal{F} \quad (\text{forma di massimo}),$$

oppure tale che

$$f(S) \geq \beta, \quad \forall S \in \mathcal{F} \quad (\text{forma di minimo}).$$

Lower bound e upper bound

- ▶ Nella **forma di massimo** un bound primale è un **lower bound** e un bound duale è un **upper bound**:

$$\text{LB} = f(\bar{S}) \leq z(I) \leq \text{UB}$$

- ▶ Nella **forma di minimo** i ruoli si scambiano: un bound primale è un **upper bound** $\text{UB} = f(\bar{S})$ e un bound duale è un **lower bound** LB , con

$$\text{LB} \leq z(I) \leq \text{UB}$$

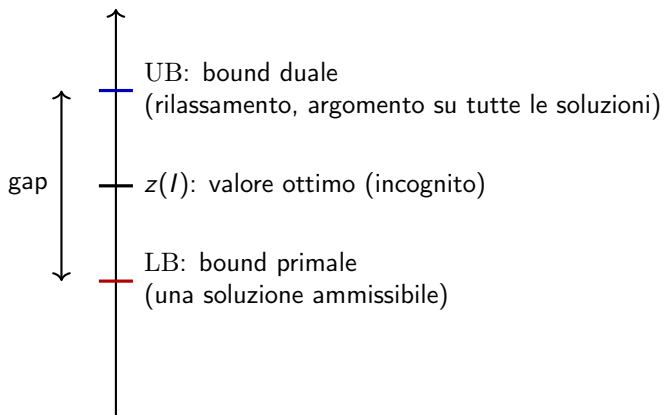
Da dove vengono i bound

- ▶ I bound primali vengono da soluzioni ammissibili, per esempio trovate da **algoritmi euristici**
- ▶ I bound duali vengono da **rilassamenti** (per esempio il rilassamento lineare) o da argomenti combinatori

La prossima figura mostra i due bound nella forma di massimo:
il valore ottimo è racchiuso fra di essi.

I bound nella forma di massimo

valore obiettivo (massimizzazione)



Definizione 3: gap

Siano $LB \leq z(I) \leq UB$ un lower bound e un upper bound di un'istanza I , uno primale e uno duale. Il **gap assoluto** è $UB - LB$ e, per $LB \neq 0$, il **gap relativo** (o semplicemente **gap**) è

$$gap(I) = \frac{UB - LB}{|LB|}.$$

Leggere il gap

- ▶ Il denominatore è sempre il **lower bound**:
 - ▶ nella forma di massimo è il valore della soluzione ammissibile, come nel gap di ottimalità del modulo *Programmazione lineare intera*, capitolo «**Bound primali e duali**»;
 - ▶ nella forma di minimo è il bound duale.
- ▶ Con questa scelta il gap non è mai minore dell'errore relativo $|f(\bar{S}) - z(I)|/z(I)$ della soluzione ammissibile $\bar{S}$: un gap del 5% garantisce una soluzione entro il 5% dall'ottimo
- ▶ Se il gap è nullo, la soluzione ammissibile $\bar{S}$ è ottima e il bound duale è un **certificato** della sua ottimalità

Osservazione 1

Se $f(S) \in \mathbb{Z}$ per ogni $S \in \mathcal{F}$, allora nella forma di massimo anche $\lfloor \beta \rfloor$ è un bound duale, e nella forma di minimo anche $\lceil \beta \rceil$ è un bound duale.

- La dimostrazione usa solo la definizione di parte intera

Dimostrazione

- ▶ Nella forma di massimo, per ogni $S \in \mathcal{F}$ abbiamo

$$f(S) \leq \beta \quad \text{e} \quad f(S) \in \mathbb{Z}$$

- ▶ Il più grande intero non maggiore di β è $\lfloor \beta \rfloor$, quindi

$$f(S) \leq \lfloor \beta \rfloor, \quad \forall S \in \mathcal{F}$$

- ▶ La forma di minimo è analoga



Esempio (1/3)

Consideriamo di nuovo la piccola istanza (massimizzazione), e supponiamo di non avere enumerato i suoi sottoinsiemi.

- ▶ *Bound primale.* Partiamo da $\emptyset$ ed esaminiamo gli elementi per peso non crescente, aggiungendo ciascuno se l'insieme resta ammissibile:
 - ▶ $\{1\}$ è ammissibile, $\{1, 2\}$ no;
 - ▶ $\{1, 3\}$ è ammissibile, $\{1, 3, 4\}$ no.

La soluzione ammissibile $\bar{S} = \{1, 3\}$ dà $LB = 11$

- ▶ *Un primo bound duale.* I pesi sono positivi, quindi

$$f(S) \leq f(\mathcal{U}) = 20, \quad \forall S \in \mathcal{F}$$

Esempio (2/3)

- *Un bound duale migliore.* Nessun sottoinsieme ammissibile contiene sia 1 sia 2, quindi un sottoinsieme ammissibile prende al più uno dei pesi 7 e 6:

$$f(S) \leq \max\{7, 6\} + 4 + 3 = 14, \quad \forall S \in \mathcal{F}$$

Il gap assoluto è $14 - 11 = 3$ e il gap relativo è $\frac{3}{11} \approx 27.3\%$

- *Un bound primale migliore.* La soluzione ammissibile $\{2, 3, 4\}$ dà $LB = 13$: il gap scende a

$$\frac{1}{13} \approx 7.7\%$$

Esempio (3/3)

- *Chiusura del gap.* L'unico sottoinsieme ammissibile con 3 elementi è $\{2, 3, 4\}$, e un sottoinsieme con al più 2 elementi ha valore al più $7 + 6 = 13$. Quindi

$$UB = 13 = LB$$

- Il gap è nullo, e questo argomento è un certificato che $\{2, 3, 4\}$ è ottima, senza elencare i 16 sottoinsiemi
- Se invece un argomento desse il bound duale 13.5, anche $\lfloor 13.5 \rfloor = 13$ sarebbe un bound duale, poiché tutti i valori $f(S)$ sono interi (per l'osservazione sui valori obiettivo interi)

I bound nel resto del corso

- ▶ Nel resto del corso, ogni capitolo su un problema registra, per le sue istanze, i migliori bound primali e duali trovati fino a quel punto
- ▶ Ogni metodo migliora l'uno o l'altro lato
- ▶ Sulla piccola istanza siamo passati da $LB = 11$, $UB = 14$ a $LB = UB = 13$

Bound primali e duali:

- ▶ Un bound primale è il valore di una soluzione ammissibile; un bound duale vale per tutte le soluzioni ammissibili
- ▶ Massimo: $LB = f(\bar{S}) \leq z(I) \leq UB$; minimo: i ruoli si scambiano

Gap:

- ▶ Gap assoluto $UB - LB$, gap relativo $(UB - LB)/|LB|$
- ▶ Un gap nullo certifica che la soluzione ammissibile è ottima
- ▶ Con valori obiettivo interi un bound duale si può arrotondare

Piccola istanza: da $LB = 11$, $UB = 14$ a $LB = UB = 13$

Cenni di complessità computazionale

Introduzione: Cenni di complessità computazionale

- ▶ La forza bruta richiede tempo esponenziale. Esiste un algoritmo che risolve un problema di ottimizzazione combinatoria in tempo polinomiale?
- ▶ La complessità computazionale classifica i problemi in base a questa domanda
- ▶ Qui diamo solo le definizioni che servono nel corso

Dimensione di un'istanza e algoritmi polinomiali

- ▶ La **dimensione** di un'istanza è il numero di bit necessari per scriverla
 - ▶ un'istanza con n numeri, il più grande dei quali è M , ha dimensione dell'ordine di $n \log_2 M$;
 - ▶ un numero M si scrive con circa $\log_2 M$ bit, non con M simboli.
- ▶ Un algoritmo è **polinomiale** se il suo tempo di esecuzione è limitato da un polinomio nella dimensione dell'istanza
- ▶ La **versione decisionale** di un problema di ottimizzazione pone una domanda sì/no, per esempio: “esiste una soluzione ammissibile di valore almeno k ?”

Definizione 4: le classi P e NP

1. **P** è la classe dei problemi decisionali che si possono risolvere con un algoritmo polinomiale.
2. **NP** è la classe dei problemi decisionali le cui istanze “sì” ammettono un certificato che si può verificare in tempo polinomiale (per esempio, una soluzione ammissibile di cui ammissibilità e valore si calcolano in tempo polinomiale).
3. Un problema decisionale in NP è **NP-completo** se ogni problema in NP si può ridurre a esso in tempo polinomiale. Un problema di ottimizzazione è **NP-hard** se la sua versione decisionale è NP-completa (più in generale, se un algoritmo polinomiale per esso darebbe un algoritmo polinomiale per un problema NP-completo).

P è uguale a NP?

- ▶ Non è noto se $P = NP$
- ▶ Non si conosce alcun algoritmo polinomiale per un problema NP-completo, ed è opinione diffusa che non ne esista nessuno
- ▶ Per un problema NP-hard non ci aspettiamo un algoritmo esatto polinomiale

Polinomiale nel valore dei numeri

- ▶ Alcuni problemi NP-hard si possono risolvere in tempo polinomiale in n e **nel valore** dei numeri dell'istanza
- ▶ Per esempio, un algoritmo che richiede tempo $O(nM)$, dove M è il numero più grande dell'istanza, non è polinomiale: M è esponenziale nel suo numero di bit $\log_2 M$

Definizione 5: algoritmo pseudo-polinomiale, problemi NP-hard in senso debole e forte

1. Un algoritmo è **pseudo-polinomiale** se il suo tempo di esecuzione è limitato da un polinomio nella dimensione dell'istanza e nel numero più grande che vi compare.
2. Un problema NP-hard è **NP-hard in senso forte** se resta NP-hard quando tutti i suoi numeri sono limitati da un polinomio nella dimensione dell'istanza (equivalentemente, quando i numeri sono scritti in unario). Altrimenti, se ammette un algoritmo pseudo-polinomiale, si dice **NP-hard in senso debole** (NP-hard in senso ordinario).

- Un problema NP-hard in senso forte non ha algoritmi pseudo-polinomiali, a meno che $P = NP$

NP-hard non significa irrisolvibile

- ▶ Che un problema sia NP-hard non significa che un'istanza non si possa risolvere: significa che non ci aspettiamo un algoritmo che risolva *ogni* istanza in tempo polinomiale
- ▶ In pratica combiniamo:
 - ▶ **metodi esatti** che chiudono il gap fra bound primali e duali;
 - ▶ **euristiche** che trovano rapidamente buone soluzioni;
 - ▶ **algoritmi di approssimazione** che garantiscono un gap limitato nel caso peggiore.

La complessità dei problemi del corso

- ▶ I problemi studiati più avanti nel corso sono tutti NP-hard:
 - ▶ il problema dello zaino 0/1 è NP-hard **in senso debole**;
 - ▶ il bin packing, la colorazione dei vertici, la clique massima e il problema del commesso viaggiatore sono NP-hard **in senso forte**.
- ▶ Ogni capitolo discute la complessità del suo problema

Dimensione e algoritmi polinomiali:

- ▶ La dimensione di un'istanza è il suo numero di bit: un numero M occupa circa $\log_2 M$ bit

Classi:

- ▶ P, NP, NP-completo; un problema di ottimizzazione è NP-hard se la sua versione decisionale è NP-completa
- ▶ Algoritmi pseudo-polinomiali; problemi NP-hard in senso debole e in senso forte

In pratica:

- ▶ Metodi esatti, euristiche e algoritmi di approssimazione, combinati

Errori tipici

Introduzione: Errori tipici

- ▶ In questa sezione raccogliamo gli errori tipici sulle nozioni del capitolo
- ▶ Riguardano il conteggio delle soluzioni, il verso e l'arrotondamento dei bound, e il significato di NP-hard
- ▶ Riconoscerli aiuta a leggere correttamente il resto del corso

Osservazione 2: errori tipici (1/3)

- ▶ **Confondere soluzioni candidate e ammissibili.** I numeri 2^n , $(n-1)!/2$ e $B(n)$ contano le candidate; le soluzioni ammissibili possono essere molte meno, per esempio 11 sottoinsiemi su 16 nella piccola istanza.
- ▶ **Scambiare i due versi dei bound.** In un problema di massimo un bound primale è un lower bound e un bound duale è un upper bound; in un problema di minimo i ruoli si scambiano.

Osservazione 3: errori tipici (2/3)

- ▶ *Arrotondare un bound duale quando l'obiettivo non è intero. L'arrotondamento dell'osservazione sui valori obiettivo interi richiede $f(S) \in \mathbb{Z}$ per ogni soluzione ammissibile.*
- ▶ *Contare due volte soluzioni equivalenti. Un ordinamento ciclico e il suo inverso sono lo stesso ordinamento ciclico, e due assegnamenti che differiscono per una ridenominazione di etichette intercambiabili sono la stessa soluzione.*

Osservazione 4: errori tipici (3/3)

- ▶ *Leggere NP-hard come irrisolvibile. Essere NP-hard è un'affermazione sul caso peggiore: molte istanze di dimensione pratica si risolvono, e il resto del corso mostra come.*

Conteggio:

- ▶ Le candidate non sono le soluzioni ammissibili; le soluzioni equivalenti si contano una volta sola

Bound:

- ▶ Primale e duale scambiano il loro verso fra massimo e minimo
- ▶ Un bound duale si arrotonda solo se ogni valore obiettivo è intero

Complessità:

- ▶ NP-hard è un'affermazione sul caso peggiore, non su ogni istanza

In sintesi

Introduzione: In sintesi

- ▶ Raccogliamo le idee chiave del capitolo: definizione, categorie, conteggio, bound e complessità
- ▶ Chiudiamo con il quadro riassuntivo della piccola istanza, che registra i suoi bound primali e duali
- ▶ Ogni capitolo su un problema terrà un quadro di questo tipo per le sue istanze

Riepilogo (1/3)

- ▶ Un problema di ottimizzazione combinatoria ha un insieme base finito, una famiglia di sottoinsiemi ammissibili e una funzione obiettivo; una soluzione ottima esiste quando esiste una soluzione ammissibile
- ▶ La categoria delle decisioni dà l'oggetto combinatorio di una soluzione: un sottoinsieme, un vettore di quantità intere, un assegnamento, un ordinamento o una partizione
- ▶ Le soluzioni candidate sono 2^n sottoinsiemi, $n!$ ordinamenti, $(n-1)!/2$ ordinamenti ciclici e $B(n)$ partizioni: la forza bruta funziona solo su istanze molto piccole

Riepilogo (2/3)

- ▶ Una soluzione ammissibile dà un bound primale, un rilassamento o un argomento combinatorio dà un bound duale; un gap nullo certifica che la soluzione ammissibile è ottima
- ▶ I problemi del corso sono NP-hard: per essi non ci aspettiamo un algoritmo esatto polinomiale

Quadro riassuntivo della piccola istanza.

piccola istanza (max)	lower bound	upper bound	da
primi bound	11	14	scelta greedy, uno dei pesi 7 e 6
bound finali	13	13	soluzione $\{2, 3, 4\}$, argomento sulla cardinalità