



SAPIENZA
UNIVERSITÀ DI ROMA

Ottimizzazione combinatoria

Catalogo dei problemi di ottimizzazione combinatoria

Tutti i problemi definiti nel corso, raggruppati per famiglia: per ciascuno la definizione, le ipotesi e che cosa fa il corso sul problema, tecnica per tecnica.

Fabio Furini

fabiofurini.github.io

Dipartimento di Ingegneria informatica, automatica e gestionale — Sapienza Università di Roma

Versione online e materiale scaricabile:
fabiofurini.github.io/ottimizzazione-combinatoria/
github.com/fabiofurini/ottimizzazione-combinatoria

Anno accademico 2026–2027

Indice

1	Problemi di zaino	3
1.1	Problema dello zaino 0/1 (KP)	4
1.2	Problema del subset sum (SSP)	6
1.3	Problema dello zaino limitato (BKP)	7
1.4	Problema dello zaino illimitato (UKP)	8
1.5	Problema dello zaino multidimensionale (MDKP)	9
1.6	Problema degli zaini multipli (MKP)	10
1.7	Problema dello zaino a scelta multipla (MCKP)	11
1.8	Problema dello zaino con setup (KPS)	12
2	Problemi di packing	13
2.1	Problema del bin packing (BPP)	13
2.2	Problema del cutting stock (CSP)	15
3	Problemi di colorazione	17
3.1	Problema della colorazione dei vertici (VCP)	17
4	Problemi di cricca	20
4.1	Problema della cricca massima (MCP)	20
4.2	Problema dell'insieme stabile massimo (MSSP)	22
4.3	Problema della cricca massima pesata sui vertici (VWMCP)	23
4.4	Problema della cricca massima pesata sui lati (EWMCP)	23
5	Problemi di routing	24
5.1	Problema del commesso viaggiatore (TSP)	24
5.2	Problema del commesso viaggiatore metrico (TSP metrico)	26
5.3	Problema del commesso viaggiatore asimmetrico (ATSP)	27
5.4	Problema del ciclo hamiltoniano (HCP)	27
6	Problemi di scheduling	28
6.1	Scheduling su macchine parallele identiche ($P C_{\max}$)	28
6.2	Scheduling su un numero fissato di macchine identiche ($Pm C_{\max}$)	30
6.3	Scheduling su macchine parallele con velocità diverse ($Q C_{\max}$)	31
6.4	Scheduling su macchine parallele scorrelate ($R C_{\max}$)	31

7	Problemi di lot sizing	32
7.1	Lot sizing capacitato (CLS)	33
7.2	Problema del lot sizing senza capacità (ULS)	34
7.3	Lot sizing capacitato con capacità costante (CLS-C)	35
7.4	Lot sizing con più prodotti e tempi di setup (MLS)	35
7.5	Lot sizing con scorta iniziale (LS-SI)	36
7.6	Lot sizing con straordinario (LS-OT)	36
7.7	Lot sizing con domanda non soddisfatta (LS-UD)	37
8	Problemi misti	38
8.1	Problema del taglio delle aste (RCP)	38
8.2	Problema del cambio monete (CCP)	39

Problemi di zaino

- Ogni oggetto j ha un profitto p_j e un peso w_j ; una soluzione seleziona un sottoinsieme, o un multiinsieme, degli oggetti i cui pesi soddisfano uno o più vincoli di capacità.
- L'obiettivo è massimizzare il profitto totale degli oggetti selezionati; nel problema del subset sum il peso totale deve essere uguale a un valore dato.
- I membri della famiglia differiscono per il numero di copie di ogni oggetto, per il numero di capacità e di zaini, e per vincoli aggiuntivi sugli oggetti selezionati.

Problema	Sigla	Differenza
Problema dello zaino 0/1	KP	Il problema centrale della famiglia.
Problema del subset sum	SSP	L'SSP si risolve come il KP con $p_j = w_j$ per ogni $j \in \mathcal{J}$: la risposta è sì se e solo se il valore ottimo di questo KP è uguale a c .
Problema dello zaino limitato	BKP	Il BKP si riduce al KP quando $b_j = 1$ per ogni oggetto, e all'UKP quando ogni limite b_j è abbastanza grande da non essere vincolante.
Problema dello zaino illimitato	UKP	L'UKP è il BKP con $b_j = \hat{b}_j = \lfloor c/w_j \rfloor$: ogni oggetto si può prendere tante volte quante entra nello zaino.
Problema dello zaino multidimensionale	MDKP	L'MDKP è il KP con k vincoli di capacità, uno per ogni risorsa; con $k = 1$ è il KP.
Problema degli zaini multipli	MKP	L'MKP è il KP con m zaini; ogni soluzione ammissibile dell'MKP è ammissibile per il KP con capacità $\sum_{k \in \mathcal{K}} c_k$, e questo dà un bound duale.
Problema dello zaino a scelta multipla	MCKP	L'MCKP è il KP con il vincolo aggiuntivo di scegliere esattamente un oggetto per ogni classe, quindi un'istanza può essere non ammissibile.
Problema dello zaino con setup	KPS	Il KPS è il KP in cui scegliere il primo oggetto di una classe costa il setup f_h e consuma il peso di setup s_h ; una classe senza oggetti scelti non costa nulla.

1.1 Problema dello zaino 0/1 (KP)

Definizione 1.1: Problema dello zaino 0/1

Dati:

1. uno zaino di *capacità* $c \in \mathbb{Z}_{\geq 1}$,
2. un insieme $\mathcal{I} = \{1, 2, \dots, n\}$ di n oggetti,
3. per ogni oggetto $j \in \mathcal{I}$, un *profitto* $p_j \in \mathbb{Z}_{\geq 1}$ e un *peso* $w_j \in \mathbb{Z}_{\geq 1}$,

il **problema dello zaino 0/1** (KP) chiede di decidere quali oggetti inserire nello zaino, in modo che il peso totale degli oggetti selezionati non superi la capacità dello zaino e il profitto totale degli oggetti selezionati sia massimo.

Ipotesi

- Facciamo le seguenti ipotesi:
 1. Il peso totale di tutti gli oggetti supera la capacità dello zaino (altrimenti si potrebbero selezionare tutti gli oggetti). Formalmente:

$$\sum_{j \in \mathcal{I}} w_j > c$$

2. Ogni oggetto, da solo, entra nello zaino (altrimenti non può essere selezionato). Formalmente:

$$w_j \leq c, \quad \forall j \in \mathcal{I}$$

1.1.1 Che cosa fa il corso su questo problema

Tecnica	Dove
Definizione	Cap. 2 «Problemi di zaino»
Formulazioni	Cap. 9 «Formulazioni dei problemi di zaino»
Programmazione dinamica	Cap. 16 «Principi della programmazione dinamica»; Cap. 17 «PD per il problema dello zaino 0/1»
Euristiche	Cap. 29 «Algoritmi euristici: definizioni»; Cap. 30 «Euristiche costruttive»; Cap. 31 «Ricerca locale»; Cap. 32 «Metaeuristiche»
Euristiche MIP	Cap. 33 «Euristiche MIP per i problemi classici»
Approssimazione	Cap. 34 «Principi degli algoritmi di approssimazione»; Cap. 35 «Algoritmi di approssimazione per lo zaino»
Algoritmi ad hoc	Cap. 41 «Zaino: branch-and-bound sull'elemento critico»
Esercizi	Cap. 2 (6 esercizi); Cap. 9 (5 esercizi); Esercizio 16.1; Cap. 17 (5 esercizi); Cap. 29 (2 esercizi); Cap. 30 (3 esercizi); Cap. 31 (3 esercizi); Cap. 32 (5 esercizi); Cap. 33 (4 esercizi); Esercizio 34.4; Cap. 35 (15 esercizi); Cap. 41 (4 esercizi)

Proprietà

- NP-difficile in senso debole: la programmazione dinamica lo risolve in $\Theta(nc)$ passi, un tempo pseudo-polinomiale (Cap. 1).
- Bound di cardinalità: se entrano al più $\bar{k}$ oggetti, $z(I)$ è al più la somma dei $\bar{k}$ profitti più grandi (Cap. 2).
- Ordinare per efficienza non basta: sull'istanza del Cap. 2 si ottiene profitto 11, mentre l'ottimo è 15 (Cap. 2).

Formulazioni

- Il modello ILP_{KP} ha una variabile binaria per oggetto e un vincolo di capacità (Cap. 9).
- Il rilassamento LP_{KP} si risolve ordinando gli oggetti per efficienza: si prendono gli oggetti prima dell'oggetto critico s , e s solo in parte (Cap. 9).
- $z(LP_{KP}) \leq 2 z(ILP_{KP})$, e il rapporto 2 è raggiunto al limite (Cap. 9).
- Arrotondando per difetto $z(LP_{KP})$ si ottiene il bound di Dantzig UB_D ; il bound di Martello–Toth UB_{MT} considera l'oggetto critico fuori e dentro lo zaino (Cap. 9).

Programmazione dinamica

- La ricorrenza $v(j, k)$ su oggetti e capacità richiede $\Theta(nc)$ passi (Cap. 17).
- Le liste degli stati non dominati tengono solo gli stati raggiungibili, al più $\min\{2^j, c + 1\}$ per stadio (Cap. 17).
- La PD sui profitti richiede $\Theta(nP)$ passi; è il nucleo dell'FPTAS (Cap. 17).
- L'UKP e il BKP hanno le loro ricorrenze (Cap. 18, Cap. 19).
- Così anche l'SSP, l'MCKP e il KPS, in $\Theta(nc)$ passi; per l'MDKP e l'MKP la tabella cresce esponenzialmente con il numero di capacità (Cap. 20, Cap. 21, Cap. 22, Cap. 23, Cap. 24).

Euristiche

- Greedy per profitto, per peso e per efficienza: sull'istanza a quattro oggetti del Cap. 29 (ottimo 17) danno 17, 10 e 16 (Cap. 30).
- Ricerca locale con flip e swap: first improvement e best improvement raggiungono 17 per strade diverse (Cap. 31).
- Una penalità sul peso in eccesso permette alla ricerca di attraversare soluzioni non ammissibili; con $\alpha > \max_j p_j$ i suoi ottimi locali sono ammissibili (Cap. 31).
- Simulated annealing, GRASP, algoritmi genetici e memetici; su KP20 l'algoritmo memetico raggiunge l'ottimo 521 in ogni esecuzione (Cap. 32).

Euristiche MIP

- Arrotondare per difetto la soluzione del rilassamento lineare dà sempre una soluzione ammissibile, perché il modello ha un solo vincolo di packing (Cap. 33).
- Sull'istanza del Cap. 29 il diving frazionario si ferma a 16 e RINS fallisce, mentre local branching e proximity search raggiungono l'ottimo 17 e ne dimostrano l'ottimalità (Cap. 33).

Approssimazione

- L'algoritmo greedy non ha una garanzia costante; Ext-Greedy è un algoritmo 2-approssimato, e il bound è tight (Cap. 35).
- H^ℓ garantisce $\frac{\ell+2}{\ell+1}$ in tempo $O(n^{\ell+1})$, e quindi dà un PTAS (Cap. 35).
- Il Basic-FPTAS scala i profitti e garantisce $\frac{1}{1-\epsilon}$ in tempo $O(n^2/\epsilon)$ (Cap. 35).
- Nessun algoritmo polinomiale ha una garanzia assoluta fissata, a meno che non risolva il KP in modo esatto (Cap. 35).

Algoritmi ad hoc

- Il rilassamento lineare di un nodo si risolve con le formule di Dantzig, senza il semplice, e ha una sola variabile frazionaria: l'oggetto critico, su cui si fa branching (Cap. 41).

1.2 Problema del subset sum (SSP)

Definizione 1.2: Problema del subset sum

Dati:

1. un valore obiettivo $c \in \mathbb{Z}_{\geq 1}$,
2. un insieme $\mathcal{J} = \{1, 2, \dots, n\}$ di n oggetti,
3. per ogni oggetto $j \in \mathcal{J}$, un peso $w_j \in \mathbb{Z}_{\geq 1}$,

il **problema del subset sum** (SSP) chiede se esiste un sottoinsieme di oggetti il cui peso totale è esattamente uguale al valore obiettivo c .

Ipotesi

- Facciamo le seguenti ipotesi:
 1. Il peso totale di tutti gli oggetti è almeno pari al valore obiettivo (altrimenti non può esistere alcun sottoinsieme ammissibile). Formalmente:

$$\sum_{j \in \mathcal{J}} w_j \geq c$$

2. Il peso di ogni oggetto non supera il valore obiettivo (altrimenti l'oggetto non può appartenere ad alcun sottoinsieme ammissibile). Formalmente:

$$w_j \leq c, \quad \forall j \in \mathcal{J}$$

1.2.1 Che cosa fa il corso su questo problema

Tecnica	Dove
Definizione	Cap. 2 «Problemi di zaino»
Formulazioni	Cap. 9 «Formulazioni dei problemi di zaino»
Programmazione dinamica	Cap. 16 «Principi della programmazione dinamica»; Cap. 20 «PD per il problema del subset sum»
Esercizi	Esercizio 2.2; Esercizio 9.6; Esercizio 16.2; Cap. 20 (5 esercizi)

1.3 Problema dello zaino limitato (BKP)

Definizione 1.3: Problema dello zaino limitato

Dati:

1. uno zaino di *capacità* $c \in \mathbb{Z}_{\geq 1}$,
2. un insieme $\mathcal{I} = \{1, 2, \dots, n\}$ di n oggetti,
3. per ogni oggetto $j \in \mathcal{I}$, un *profitto* $p_j \in \mathbb{Z}_{\geq 1}$, un *peso* $w_j \in \mathbb{Z}_{\geq 1}$ e un *numero di copie disponibili* $b_j \in \mathbb{Z}_{\geq 1}$,

il **problema dello zaino limitato** (BKP) chiede di decidere quante copie di ogni oggetto inserire nello zaino, in modo che il peso totale di tutte le copie selezionate non superi la capacità dello zaino, che nessun oggetto sia preso più volte delle sue copie disponibili e che il profitto totale di tutte le copie selezionate sia massimo.

Ipotesi

- Facciamo le seguenti ipotesi:
 1. Il peso totale di tutte le copie disponibili di tutti gli oggetti supera la capacità dello zaino (altrimenti si potrebbero selezionare tutte le copie). Formalmente:

$$\sum_{j \in \mathcal{I}} b_j w_j > c$$

2. Ogni oggetto, da solo, entra nello zaino (altrimenti non può essere selezionato). Formalmente:

$$w_j \leq c, \quad \forall j \in \mathcal{I}$$

3. Il peso totale di tutte le copie disponibili di un singolo oggetto non supera la capacità dello zaino (altrimenti quell'oggetto non potrebbe mai essere preso nel numero massimo di copie consentito). Formalmente:

$$b_j w_j \leq c, \quad \forall j \in \mathcal{I}$$

1.3.1 Che cosa fa il corso su questo problema

Tecnica	Dove
Definizione	Cap. 2 «Problemi di zaino»
Formulazioni	Cap. 9 «Formulazioni dei problemi di zaino»
Programmazione dinamica	Cap. 16 «Principi della programmazione dinamica»; Cap. 19 «PD per il problema dello zaino limitato»
Esercizi	Esercizio 2.3; Esercizio 9.7; Cap. 19 (5 esercizi)

1.4 Problema dello zaino illimitato (UKP)

Definizione 1.4: Problema dello zaino illimitato

Dati:

1. uno zaino di *capacità* $c \in \mathbb{Z}_{\geq 1}$,
2. un insieme $\mathcal{I} = \{1, 2, \dots, n\}$ di n oggetti,
3. per ogni oggetto $j \in \mathcal{I}$, disponibile in un numero illimitato di copie, un *profitto* $p_j \in \mathbb{Z}_{\geq 1}$ e un *peso* $w_j \in \mathbb{Z}_{\geq 1}$,

il **problema dello zaino illimitato** (UKP) chiede di decidere quante copie di ogni oggetto inserire nello zaino, in modo che il peso totale di tutte le copie selezionate non superi la capacità dello zaino e il profitto totale di tutte le copie selezionate sia massimo.

Ipotesi

- Facciamo le seguenti ipotesi:

1. Il peso totale di tutte le copie possibili di tutti gli oggetti (fino al numero massimo che entra nello zaino) supera la capacità dello zaino (altrimenti si potrebbero selezionare tutte le copie). Formalmente:

$$\sum_{j \in \mathcal{I}} \hat{b}_j w_j > c$$

2. Ogni oggetto, da solo, entra nello zaino (altrimenti non può essere selezionato). Formalmente:

$$w_j \leq c, \quad \forall j \in \mathcal{I}$$

1.4.1 Che cosa fa il corso su questo problema

Tecnica	Dove
Definizione	Cap. 2 «Problemi di zaino»
Formulazioni	Cap. 9 «Formulazioni dei problemi di zaino»
Programmazione dinamica	Cap. 16 «Principi della programmazione dinamica»; Cap. 18 «PD per il problema dello zaino illimitato»; Cap. 27 «PD per il problema del cambio monete»; Cap. 28 «PD per il problema del taglio delle aste»
Esercizi	Esercizio 2.3; Esercizio 9.8; Cap. 18 (5 esercizi)

1.5 Problema dello zaino multidimensionale (MDKP)

Definizione 1.5: Problema dello zaino multidimensionale

Dati:

1. uno zaino k -dimensionale,
2. per ogni dimensione $d \in \underbrace{\{1, 2, \dots, k\}}_{=\mathcal{D}}$, una *capacità* $c_d \in \mathbb{Z}_{\geq 1}$,
3. un insieme $\mathcal{I} = \{1, 2, \dots, n\}$ di n oggetti,
4. per ogni oggetto $j \in \mathcal{I}$, un *profitto* $p_j \in \mathbb{Z}_{\geq 1}$,
5. per ogni dimensione $d \in \mathcal{D}$ e ogni oggetto $j \in \mathcal{I}$, un *peso* $w_{dj} \in \mathbb{Z}_{\geq 1}$,

il **problema dello zaino multidimensionale** (MDKP) chiede di decidere quali oggetti inserire nello zaino, in modo che per ogni dimensione il peso totale degli oggetti selezionati non superi la capacità corrispondente e il profitto totale degli oggetti selezionati sia massimo.

Ipotesi

- Facciamo le seguenti ipotesi:
 1. L'insieme di tutti gli oggetti *non* entra nello zaino (altrimenti selezionare tutti gli oggetti è ammissibile e ottimo). Formalmente,

$$\exists d \in \mathcal{D} \text{ tale che } \sum_{j \in \mathcal{I}} w_{dj} > c_d.$$

2. Ogni oggetto, da solo, entra nello zaino in ogni dimensione (altrimenti non può mai essere selezionato). Formalmente,

$$w_{dj} \leq c_d, \quad \forall d \in \mathcal{D}, \forall j \in \mathcal{I}.$$

1.5.1 Che cosa fa il corso su questo problema

Tecnica	Dove
Definizione	Cap. 2 «Problemi di zaino»
Formulazioni	Cap. 9 «Formulazioni dei problemi di zaino»
Programmazione dinamica	Cap. 16 «Principi della programmazione dinamica»; Cap. 22 «PD per il problema dello zaino multidimensionale»
Esercizi	Esercizio 9.9; Cap. 22 (5 esercizi)

1.6 Problema degli zaini multipli (MKP)

Definizione 1.6: Problema degli zaini multipli

Dati:

1. $m \in \mathbb{Z}_{\geq 1}$ zaini,
2. per ogni zaino $k \in \underbrace{\{1, 2, \dots, m\}}_{=\mathcal{K}}$, una *capacità* $c_k \in \mathbb{Z}_{\geq 1}$,
3. un insieme $\mathcal{I} = \{1, 2, \dots, n\}$ di n oggetti,
4. per ogni oggetto $j \in \mathcal{I}$, un *profitto* $p_j \in \mathbb{Z}_{\geq 1}$ e un *peso* $w_j \in \mathbb{Z}_{\geq 1}$,

il **problema degli zaini multipli** (MKP) chiede di decidere quali oggetti inserire in ogni zaino, in modo che in ogni zaino il peso totale degli oggetti assegnati non superi la sua capacità e il profitto totale degli oggetti selezionati sia massimo.

Ipotesi

- Facciamo le seguenti ipotesi:
 1. L'insieme di tutti gli oggetti *non* entra nel complesso degli zaini (altrimenti selezionare tutti gli oggetti è ammissibile e ottimo). Formalmente,

$$\sum_{j \in \mathcal{I}} w_j > \sum_{k \in \mathcal{K}} c_k.$$

2. Ogni oggetto, da solo, entra in (almeno) uno zaino (altrimenti non può mai essere selezionato). Formalmente,

$$\forall j \in \mathcal{I} \quad \exists k \in \mathcal{K} \text{ tale che } w_j \leq c_k.$$

1.6.1 Che cosa fa il corso su questo problema

Tecnica	Dove
Definizione	Cap. 2 «Problemi di zaino»
Formulazioni	Cap. 9 «Formulazioni dei problemi di zaino»
Programmazione dinamica	Cap. 16 «Principi della programmazione dinamica»; Cap. 24 «PD per il problema degli zaini multipli»
Esercizi	Esercizio 2.4; Esercizio 9.10; Cap. 24 (5 esercizi)

1.7 Problema dello zaino a scelta multipla (MCKP)

Definizione 1.7: Problema dello zaino a scelta multipla

Dati:

1. uno zaino di *capacità* $c \in \mathbb{Z}_{\geq 1}$,
2. un insieme $\mathcal{J} = \{1, 2, \dots, n\}$ di n oggetti,
3. una partizione dell'insieme degli oggetti in $q \in \mathbb{Z}_{\geq 2}$ classi,
4. per ogni classe $h \in \{1, 2, \dots, q\}$, il sottoinsieme $\mathcal{J}_h \subseteq \mathcal{J}$ degli oggetti di quella classe,
5. per ogni oggetto $j \in \mathcal{J}$, un *profitto* $p_j \in \mathbb{Z}_{\geq 1}$ e un *peso* $w_j \in \mathbb{Z}_{\geq 1}$,

il **problema dello zaino a scelta multipla** (MCKP) chiede di decidere quali oggetti inserire nello zaino, in modo che da ogni classe sia selezionato esattamente un oggetto, che il peso totale degli oggetti selezionati non superi la capacità dello zaino e che il profitto totale degli oggetti selezionati sia massimo.

Ipotesi

- Facciamo le seguenti ipotesi:
 1. Ogni classe contiene almeno un oggetto (altrimenti selezionare esattamente un oggetto da ogni classe è impossibile). Formalmente:

$$\mathcal{J}_h \neq \emptyset \quad \forall h \in \{1, \dots, q\}.$$

2. L'istanza è ammissibile, ossia esiste almeno una scelta di esattamente un oggetto per classe che entra nello zaino. Una condizione necessaria e sufficiente (e facile da verificare) è:

$$\sum_{h=1}^q \min_{j \in \mathcal{J}_h} w_j \leq c.$$

1.7.1 Che cosa fa il corso su questo problema

Tecnica	Dove
Definizione	Cap. 2 «Problemi di zaino»
Formulazioni	Cap. 9 «Formulazioni dei problemi di zaino»
Programmazione dinamica	Cap. 16 «Principi della programmazione dinamica»; Cap. 21 «PD per il problema dello zaino a scelta multipla»
Esercizi	Esercizio 2.5; Esercizio 9.11; Esercizio 16.5; Cap. 21 (5 esercizi)

1.8 Problema dello zaino con setup (KPS)

Definizione 1.8: Problema dello zaino con setup

Dati:

1. uno zaino di *capacità* $c \in \mathbb{Z}_{\geq 1}$,
2. un insieme $\mathcal{J} = \{1, 2, \dots, n\}$ di n oggetti,
3. una partizione dell'insieme degli oggetti in $q \in \mathbb{Z}_{\geq 2}$ classi,
4. per ogni classe $h \in \{1, 2, \dots, q\}$, il sottoinsieme $\mathcal{J}_h \subseteq \mathcal{J}$ degli oggetti di quella classe, con un *costo di setup* $f_h \in \mathbb{Z}_{\geq 1}$ e un *peso di setup* $s_h \in \mathbb{Z}_{\geq 1}$ associati,
5. per ogni oggetto $j \in \mathcal{J}$, un *ricavo* $p_j \in \mathbb{Z}_{\geq 1}$ e un *peso* $w_j \in \mathbb{Z}_{\geq 1}$,

il **problema dello zaino con setup** (KPS) chiede di decidere quali oggetti inserire nello zaino, in modo che il peso totale degli oggetti selezionati più i pesi di setup delle loro classi non superi la capacità dello zaino e il ricavo totale degli oggetti selezionati meno i costi di setup delle loro classi sia massimo.

Ipotesi

- Facciamo le seguenti ipotesi:

1. Ogni classe contiene almeno un oggetto. Formalmente:

$$\mathcal{J}_h \neq \emptyset \quad \forall h \in \{1, 2, \dots, q\}.$$

2. Ogni oggetto entra nello zaino insieme al peso di setup della sua classe (altrimenti non può mai essere selezionato). Formalmente:

$$w_j + s_h \leq c \quad \forall h \in \{1, 2, \dots, q\}, \forall j \in \mathcal{J}_h.$$

1.8.1 Che cosa fa il corso su questo problema

Tecnica	Dove
Definizione	Cap. 2 «Problemi di zaino»
Formulazioni	Cap. 9 «Formulazioni dei problemi di zaino»
Programmazione dinamica	Cap. 16 «Principi della programmazione dinamica»; Cap. 23 «PD per il problema dello zaino con setup»
Esercizi	Esercizio 9.12; Cap. 23 (5 esercizi)

Problemi di packing

- Ogni oggetto j ha un peso w_j ; una soluzione è una partizione degli oggetti in contenitori di capacità c , tale che il peso totale degli oggetti in ogni contenitore sia al più c .
- L'obiettivo è minimizzare il numero di contenitori; nel problema del cutting stock gli oggetti sono raggruppati in tipi con una domanda d_j .

Problema	Sigla	Differenza
Problema del bin packing	BPP	Il problema centrale della famiglia.
Problema del cutting stock	CSP	Il CSP è il BPP con d_j copie identiche di ogni oggetto j , e il BPP è il caso $d_j = 1$ per ogni $j \in \mathcal{J}$.

2.1 Problema del bin packing (BPP)

Definizione 2.1: Problema del bin packing

Dati:

1. un numero infinito di contenitori, ciascuno di capacità $c \in \mathbb{Z}_{\geq 1}$,
2. un insieme $\mathcal{J} = \{1, 2, \dots, n\}$ di n oggetti,
3. per ogni oggetto $j \in \mathcal{J}$, un peso $w_j \in \mathbb{Z}_{\geq 1}$,

il **problema del bin packing** (BPP) chiede di decidere in quale contenitore sistemare ogni oggetto, in modo che il peso totale degli oggetti in ogni contenitore non superi la capacità del contenitore e il numero totale di contenitori usati sia minimo.

Ipotesi

- Facciamo le seguenti ipotesi:
 1. Il peso totale di tutti gli oggetti supera la capacità di un contenitore (altrimenti tutti gli oggetti entrerebbero in un solo contenitore). Formalmente:

$$\sum_{j \in \mathcal{J}} w_j > c$$

2. Ogni oggetto, da solo, entra in un contenitore (altrimenti non può essere impaccato).
Formalmente:

$$w_j \leq c, \quad \forall j \in \mathcal{I}$$

2.1.1 Che cosa fa il corso su questo problema

Tecnica	Dove
Definizione	Cap. 3 «Problemi di packing»
Formulazioni	Cap. 10 «Formulazioni dei problemi di packing»
Euristiche	Cap. 29 «Algoritmi euristici: definizioni»; Cap. 30 «Euristiche costruttive»; Cap. 31 «Ricerca locale»; Cap. 32 «Metaeuristiche»
Euristiche MIP	Cap. 33 «Euristiche MIP per i problemi classici»
Approssimazione	Cap. 36 «Algoritmi di approssimazione per il bin packing»
Esercizi	Cap. 3 (6 esercizi); Esercizio 7.5; Cap. 10 (5 esercizi); Esercizio 29.2; Cap. 30 (3 esercizi); Cap. 31 (3 esercizi); Cap. 32 (2 esercizi); Esercizio 34.3; Cap. 36 (12 esercizi)

Proprietà

- NP-difficile in senso forte: non esiste un algoritmo pseudo-polinomiale, a meno che $P = NP$ (Cap. 1).
- Bound sui pesi $LB = \lceil \sum_j w_j / c \rceil$; non è sempre stretto: tre oggetti di peso 6 con $c = 10$ richiedono 3 contenitori, non 2 (Cap. 3).

Formulazioni

- Il modello ILP_{BPP} ha mn variabili di assegnamento x_{jb} e m variabili di attivazione y_b (Cap. 10).
- Entrambi i rilassamenti lineari valgono $\sum_j w_j / c$: arrotondato per eccesso, è di nuovo il bound sui pesi (Cap. 10).
- Il rapporto fra il bound del rilassamento lineare e l'ottimo è almeno $\frac{1}{2}$, e $\frac{1}{2}$ è raggiunto al limite (Cap. 10).

Euristiche

- First Fit Decreasing e Best Fit Decreasing trovano l'ottimo 3 sull'istanza del Cap. 29; $FFD(I) \leq \frac{11}{9} z(I) + \frac{6}{9}$, tight (Cap. 30).
- Il numero di contenitori è piatto, quindi la ricerca locale con move e swap usa l'obiettivo ausiliario $\max \sum_b L_b^2$ (Cap. 31).
- La VND con move e swap raggiunge l'ottimo in 89 esecuzioni su 100 su un'istanza con $n = 12$ (Cap. 31).
- Large neighbourhood search: su BPP24 la riparazione esatta con $q \geq 3$ raggiunge l'ottimo 9 in ogni esecuzione (Cap. 32).

Euristiche MIP

- L'arrotondamento e il diving frazionario falliscono sull'istanza del [Cap. 29](#): il rilassamento lineare distribuisce un oggetto su più contenitori ([Cap. 33](#)).
- La feasibility pump trova sempre una soluzione ammissibile, ma può lasciare aperti contenitori vuoti ([Cap. 33](#)).
- Su un'istanza con $n = 60$ tutta la difficoltà è primale: le euristiche di Gurobi decidono il tempo di soluzione ([Cap. 33](#)).

Approssimazione

- Ogni algoritmo Any Fit è half-full, quindi è un algoritmo 2-approssimato ([Cap. 36](#)).
- $r(NF) = r(WF) = 2$ e $r(FF) = r(BF) = \frac{17}{10}$ ([Cap. 36](#)).

2.2 Problema del cutting stock (CSP)

Definizione 2.2: Problema del cutting stock

Dati:

1. un numero infinito di contenitori, ciascuno di *capacità* $c \in \mathbb{Z}_{\geq 1}$,
2. un insieme $\mathcal{J} = \{1, 2, \dots, n\}$ di n oggetti,
3. per ogni oggetto $j \in \mathcal{J}$, un *peso* $w_j \in \mathbb{Z}_{\geq 1}$ e un *numero di copie richieste* $d_j \in \mathbb{Z}_{\geq 1}$,

il **problema del cutting stock** (CSP) chiede di decidere in quale contenitore sistemare ogni copia di ogni oggetto, in modo che il peso totale delle copie in ogni contenitore non superi la capacità del contenitore, che il numero richiesto di copie di ogni oggetto sia sistemato nei contenitori e che il numero totale di contenitori usati sia minimo.

Ipotesi

- Facciamo le seguenti ipotesi:
 1. Il peso totale di tutte le copie richieste supera la capacità di un contenitore (altrimenti tutte le copie entrerebbero in un solo contenitore). Formalmente:

$$\sum_{j \in \mathcal{J}} d_j w_j > c.$$

2. Ogni oggetto, da solo, entra in un contenitore (altrimenti nessuna copia di quell'oggetto può essere impaccata). Formalmente:

$$w_j \leq c, \quad \forall j \in \mathcal{J}.$$

2.2.1 Che cosa fa il corso su questo problema

Tecnica	Dove
Definizione	Cap. 3 «Problemi di packing»
Formulazioni	Cap. 10 «Formulazioni dei problemi di packing»
Esercizi	Esercizio 3.4; Esercizio 10.6

Problemi di colorazione

- Dato un grafo non orientato $G = (V, E)$, una soluzione assegna un colore a ogni vertice in modo che vertici adiacenti ricevano colori diversi.
- In modo equivalente, una soluzione è una partizione di V in insiemi stabili; il minimo numero di colori è il numero cromatico $\chi(G)$.

Problema	Sigla	Differenza
Problema della colorazione dei vertici	VCP	Il problema centrale della famiglia.

3.1 Problema della colorazione dei vertici (VCP)

Definizione 3.1: Problema della colorazione dei vertici

Dato:

1. un *grafo* semplice $G = (V, E)$, dove

$$V = \{1, 2, \dots, n\}$$

è l'insieme degli n vertici e

$$E \subseteq \{ \{u, v\} : u, v \in V, u \neq v \}$$

è l'insieme degli m lati,

il **problema della colorazione dei vertici** (VCP) chiede di decidere come assegnare un colore a ogni vertice, in modo che due vertici adiacenti non abbiano lo stesso colore e il numero totale di colori usati sia minimo.

Ipotesi

- Facciamo le seguenti ipotesi:

1. Il grafo non ha vertici isolati. Un vertice isolato può ricevere un qualunque colore usato dal resto del grafo; se il grafo non ha lati, il valore ottimo è uno. Formalmente:

$$\deg(v) \geq 1, \quad \forall v \in V.$$

2. Nessun vertice ha l'intorno contenuto nell'intorno di un altro vertice. Formalmente:

$$N(u) \not\subseteq N(v), \quad \forall u, v \in V, u \neq v.$$

- La seconda ipotesi si può fare senza perdita di generalità, perché:
 1. se $N(u) \subseteq N(v)$, il vertice u si può eliminare senza cambiare il numero cromatico: ogni colorazione di $G \setminus \{u\}$ si estende a u con il colore di v ;
 2. i vertici u e v non sono adiacenti: altrimenti $v \in N(u) \subseteq N(v)$, il che è impossibile in un grafo semplice, quindi u e v possono condividere un colore.

3.1.1 Che cosa fa il corso su questo problema

Tecnica	Dove
Definizione	Cap. 4 «Problemi di colorazione»
Formulazioni	Cap. 11 «Formulazioni dei problemi di colorazione»
Euristiche	Cap. 30 «Euristiche costruttive»
Limiti dell'approssimazione	Cap. 37 «Colorazione dei vertici: limiti dell'approssimazione»
Algoritmi ad hoc	Cap. 42 «Colorazione dei vertici: il branch-and-bound DSATUR»
Esercizi	Cap. 4 (5 esercizi); Cap. 11 (5 esercizi); Cap. 30 (3 esercizi); Cap. 37 (5 esercizi); Cap. 42 (5 esercizi)

Proprietà

- NP-difficile in senso forte: decidere se $\chi(G) \leq 3$ è NP-completo (Cap. 1).
- Bound di cricca $\omega(G) \leq \chi(G)$ e bound degli insiemi stabili $\lceil n/\alpha(G) \rceil \leq \chi(G)$ (Cap. 4).
- I bound possono non raggiungere $\chi(G)$: sul grafo di Petersen $\omega(G) = 2 < \chi(G) = 3$ (Cap. 4).
- Brooks: $\chi(G) \leq \Delta$ per un grafo connesso che non è né completo né un ciclo dispari (Cap. 4).

Formulazioni

- Il modello ILP_{VCP} ha kn variabili x_{vc} e k variabili y_c , e $k!$ copie equivalenti di ogni colorazione (Cap. 11).
- Entrambi i rilassamenti lineari valgono 2 per ogni grafo con almeno un lato (Cap. 11).
- Sul grafo completo con n vertici il gap è $n - 2$ e il rapporto $\frac{2}{n}$ tende a 0 (Cap. 11).

Euristiche

- La colorazione greedy usa al più $\Delta(G) + 1$ colori, e l'ordine dei vertici conta ([Cap. 30](#)).
- *DSATUR* colora per primo il vertice che vede più colori; è esatto sui grafi bipartiti ([Cap. 30](#)).

Approssimazione

- Non approssimabile: nessun algoritmo polinomiale garantisce un rapporto inferiore a $\frac{4}{3}$, a meno che $P = NP$, e nemmeno $n^{1-\varepsilon}$ (Zuckerman) ([Cap. 37](#)).
- Casi particolari: grafi bipartiti e grafi di intervalli in modo esatto, grafi planari entro 2 colori dall'ottimo ([Cap. 37](#)).

Algoritmi ad hoc

- Il branch-and-bound DSATUR fa branching sul vertice più saturo, con un figlio per ogni colore ammissibile e uno per un colore nuovo, e pota un nodo quando $q(k) \geq \text{UB}$ ([Cap. 42](#)).
- Sul grafo di Petersen dimostra $\chi(G) = 3$ con 6 nodi ([Cap. 42](#)).

Problemi di cricca

- Dato un grafo non orientato $G = (V, E)$, una cricca è un sottoinsieme di vertici a due a due adiacenti; l'obiettivo è massimizzarne la cardinalità $\omega(G)$, o un peso dei suoi vertici o dei suoi lati.
- Un insieme stabile di G è una cricca del grafo complementare $\bar{G}$, quindi il problema dell'insieme stabile massimo appartiene alla stessa famiglia.

Problema	Sigla	Differenza
Problema della cricca massima	MCP	Il problema centrale della famiglia.
Problema dell'insieme stabile massimo	MSSP	Un sottoinsieme è una cricca di G se e solo se è un insieme stabile di $\bar{G}$: $\omega(G) = \alpha(\bar{G})$ e $\alpha(G) = \omega(\bar{G})$.
Problema della cricca massima pesata sui vertici	VWMCP	Con tutti i pesi $w_v = 1$ è l'MCP.
Problema della cricca massima pesata sui lati	EWMCP	Con $r_{uv} = 1$ è di nuovo l'MCP, perché $r(K) = \binom{ K }{2}$ cresce con $ K $.

4.1 Problema della cricca massima (MCP)

Definizione 4.1: Problema della cricca massima

Dato:

1. un *grafo* semplice $G = (V, E)$, dove

$$V = \{1, 2, \dots, n\}$$

è l'insieme degli n *vertici* e

$$E \subseteq \{ \{u, v\} : u, v \in V, u \neq v \}$$

è l'insieme degli m *lati*,

il **problema della clique massima** (MCP) chiede di trovare una clique $K \subseteq V$ di G il cui numero di vertici sia massimo. Il suo valore ottimo è il numero di clique $\omega(G)$.

Ipotesi

- Consideriamo istanze non banali: il grafo ha almeno un lato e almeno un non-lato. Formalmente:

$$1 \leq m < \binom{n}{2}.$$

- L'ipotesi si può fare senza perdita di generalità: se il grafo non ha lati, ogni singolo vertice è una clique massima e $\omega(G) = 1$; se il grafo è completo, V stesso è una clique e $\omega(G) = n$.

4.1.1 Che cosa fa il corso su questo problema

Tecnica	Dove
Definizione	Cap. 5 «Problemi di clique»
Formulazioni	Cap. 12 «Formulazioni dei problemi di clique»
Limiti dell'approssimazione	Cap. 38 «Clique massima: limiti dell'approssimazione»
Esercizi	Cap. 4 (2 esercizi); Cap. 5 (5 esercizi); Cap. 12 (5 esercizi); Cap. 38 (4 esercizi)

Proprietà

- NP-difficile: decidere se esiste una cricca con k vertici è uno dei 21 problemi NP-completi di Karp (Cap. 5).
- Upper bound: grado $\Delta + 1$, lati, conteggio dei gradi $k^{\max}$ e colorazione $\chi(G)$ (Cap. 5).
- Una prima cricca a mano: si parte da un vertice di grado massimo e si aggiungono vertici adiacenti a tutti quelli scelti (Cap. 5).

Formulazioni

- Il modello ILP_{MCP} ha una variabile binaria per vertice e un vincolo $x_u + x_v \leq 1$ per ogni coppia di vertici non adiacenti (Cap. 12).

- $x_v = \frac{1}{2}$ per tutti i vertici è ammissibile, quindi $z(\text{LP}_{\text{MCP}}) \geq n/2$: il bound del rilassamento lineare può essere arbitrariamente debole (Cap. 12).
- Le disuguaglianze di cricca sugli insiemi stabili di G rafforzano il modello, e il duale del loro rilassamento è un rilassamento del VCP: $\omega(G) \leq z(\text{LP}_{\text{MCP}}^{\text{CL}}) \leq \chi(G)$ (Cap. 12).

Approssimazione

- Non approssimabile: per ogni $\varepsilon > 0$ nessun algoritmo polinomiale garantisce un rapporto entro $n^{1-\varepsilon}$, a meno che $P = NP$ (Zuckerman) (Cap. 38).
- Casi particolari: grafi planari e grafi di grado limitato, risolti in modo esatto per enumerazione (Cap. 38).

4.2 Problema dell'insieme stabile massimo (MSSP)

Definizione 4.2: Problema dell'insieme stabile massimo

Sia $G = (V, E)$ un grafo.

1. Un **insieme stabile** (detto anche **insieme indipendente**) è un sottoinsieme $Q \subseteq V$ di vertici a due a due non adiacenti: $\{u, v\} \notin E$ per ogni $u, v \in Q$ con $u \neq v$.
2. Il **problema dell'insieme stabile massimo** (MSSP) chiede di trovare un insieme stabile di G il cui numero di vertici sia massimo. Il suo valore ottimo è il **numero di stabilità** $\alpha(G)$.

Ipotesi

- Consideriamo istanze non banali: il grafo ha almeno un lato e almeno un non-lato. Formalmente:

$$1 \leq m < \binom{n}{2}.$$

- L'ipotesi si può fare senza perdita di generalità: se il grafo non ha lati, ogni singolo vertice è una clique massima e $\omega(G) = 1$; se il grafo è completo, V stesso è una clique e $\omega(G) = n$.

4.2.1 Che cosa fa il corso su questo problema

Tecnica	Dove
Definizione	Cap. 5 «Problemi di clique»
Formulazioni	Cap. 12 «Formulazioni dei problemi di clique»
Limiti dell'approssimazione	Cap. 38 «Clique massima: limiti dell'approssimazione»
Esercizi	Esercizio 4.3; Cap. 5 (2 esercizi); Esercizio 12.6; Cap. 38 (2 esercizi)

4.3 Problema della cricca massima pesata sui vertici (VWMCP)

Definizione 4.3: Problema della cricca massima pesata sui vertici

1. Dati un grafo G e un vettore dei pesi dei vertici w , il **problema della clique massima pesata sui vertici** (VWMCP) chiede di trovare una clique K di G con peso totale dei vertici $w(K)$ massimo.
2. Dati un grafo G e un vettore dei pesi dei lati r , il **problema della clique massima pesata sui lati** (EWMCP) chiede di trovare una clique K di G con peso totale dei lati $r(K)$ massimo.

4.3.1 Che cosa fa il corso su questo problema

Tecnica	Dove
Definizione	Cap. 5 «Problemi di clique»
Formulazioni	Cap. 12 «Formulazioni dei problemi di clique»
Esercizi	Esercizio 5.5; Esercizio 12.7

4.4 Problema della cricca massima pesata sui lati (EWMCP)

Definizione 4.4: Problema della cricca massima pesata sui lati

1. Dati un grafo G e un vettore dei pesi dei vertici w , il **problema della clique massima pesata sui vertici** (VWMCP) chiede di trovare una clique K di G con peso totale dei vertici $w(K)$ massimo.
2. Dati un grafo G e un vettore dei pesi dei lati r , il **problema della clique massima pesata sui lati** (EWMCP) chiede di trovare una clique K di G con peso totale dei lati $r(K)$ massimo.

4.4.1 Che cosa fa il corso su questo problema

Tecnica	Dove
Definizione	Cap. 5 «Problemi di clique»
Formulazioni	Cap. 12 «Formulazioni dei problemi di clique»
Esercizi	Esercizio 12.8

Problemi di routing

- Date n città e la distanza d_{ij} fra ogni coppia ordinata, una soluzione è un giro che visita ogni città esattamente una volta e torna alla prima.
- L'obiettivo è minimizzare la lunghezza totale del giro; i membri della famiglia differiscono per le proprietà delle distanze: simmetriche, metriche o asimmetriche.
- Il problema del ciclo hamiltoniano è il problema di decisione su un grafo: chiede se esiste un giro fatto di lati del grafo.

Problema	Sigla	Differenza
Problema del commesso viaggiatore	TSP	Il problema centrale della famiglia.
Problema del commesso viaggiatore metrico	TSP metrico	Il caso particolare del TSP con la disuguaglianza triangolare, in cui esistono garanzie di approssimazione.
Problema del commesso viaggiatore asimmetrico	ATSP	Il TSP è il caso particolare dell'ATSP in cui $d_{ij} = d_{ji}$ per ogni i, j ; un'istanza dell'ATSP ha $(n-1)!$ giri.
Problema del ciclo hamiltoniano	HCP	Con $d_{ij} = 1$ sui lati e una distanza grande altrove, G è hamiltoniano se e solo se $z(I_G) = n$: il TSP lo contiene.

5.1 Problema del commesso viaggiatore (TSP)

Definizione 5.1: Problema del commesso viaggiatore

Dati:

1. un insieme $\mathcal{V} = \{1, 2, \dots, n\}$ di n città,
2. per ogni coppia di città distinte $i, j \in \mathcal{V}$, una distanza $d_{ij} \in \mathbb{Z}_{\geq 1}$, con $d_{ij} = d_{ji}$,

il **problema del commesso viaggiatore** (TSP) chiede di trovare un percorso chiuso che parte da una città, visita ogni altra città esattamente una volta e ritorna alla città di partenza, in modo che la distanza totale percorsa sia minima.

Ipotesi

- Facciamo le seguenti ipotesi:

1. Ci sono almeno tre città (con una o due città c'è un solo percorso). Formalmente:

$$n \geq 3$$

2. Le distanze sono simmetriche (andare da i a j costa quanto andare da j a i).
Formalmente:

$$d_{ij} = d_{ji}, \quad \forall i, j \in \mathcal{V}, i \neq j$$

5.1.1 Che cosa fa il corso su questo problema

Tecnica	Dove
Definizione	Cap. 6 «Problemi di routing»
Formulazioni	Cap. 13 «Formulazioni dei problemi di routing»
Euristiche	Cap. 29 «Algoritmi euristici: definizioni»; Cap. 30 «Euristiche costruttive»; Cap. 31 «Ricerca locale»; Cap. 32 «Metaeuristiche»
Euristiche MIP	Cap. 33 «Euristiche MIP per i problemi classici»
Approssimazione	Cap. 39 «TSP: il caso generale e il caso metrico»
Esercizi	Cap. 6 (7 esercizi); Cap. 13 (5 esercizi); Esercizio 29.3; Cap. 30 (3 esercizi); Cap. 31 (3 esercizi); Cap. 32 (7 esercizi); Cap. 39 (3 esercizi)

Proprietà

- NP-difficile in senso forte: contiene il problema del ciclo hamiltoniano (Cap. 1).
- Lower bound $\lceil \frac{1}{2} \sum_i (a_i + b_i) \rceil$, con a_i, b_i i due lati più corti incidenti nella città i (Cap. 6).
- Il nearest neighbour dalla città 1 dà 25 sull'istanza a cinque città del Cap. 6, il cui ottimo è 18 (Cap. 6).

Formulazioni

- Il modello MTZ elimina i sottogiri con $n - 1$ variabili di posizione u_i ; il modello DFJ con $2^{n-1} - n$ vincoli di eliminazione dei sottogiri, aggiunti solo quando sono violati (Cap. 13).
- Il rilassamento MTZ è debole; sull'istanza del Cap. 6 i bound sono 16 (assegnamento), 17 (MTZ) e 18 (DFJ) (Cap. 13).

Euristiche

- Il nearest neighbour multi-start e le euristiche di inserimento NI, FI e CI raggiungono tutti 18 sull'istanza del Cap. 6 (Cap. 30).
- 2-opt valuta ciascuna delle sue $\frac{n(n-3)}{2}$ mosse in $O(1)$ e trasforma il giro di lunghezza 25 in un giro ottimo in due mosse (Cap. 31).
- Or-opt e la VND sfuggono agli ottimi locali di 2-opt (Cap. 31).

- Simulated annealing, tabu search, iterated local search, GRASP e LNS sono confrontati su TSP20 (ottimo 291) (Cap. 32).

Euristiche MIP

- Con i vincoli di eliminazione dei sottogiri l'ottimo del rilassamento lineare dell'istanza del Cap. 6 è intero: l'arrotondamento dà 18, gap 0% (Cap. 33).

Approssimazione

- TSP metrico: l'algoritmo del doppio albero è un algoritmo 2-approssimato, tight (Cap. 39).
- Christofides è un algoritmo $\frac{3}{2}$ -approssimato, tight (Cap. 39).
- TSP generale: nessun rapporto costante, a meno che $P = NP$ (Cap. 39).

5.2 Problema del commesso viaggiatore metrico (TSP metrico)

Definizione 5.2: Problema del commesso viaggiatore metrico

Un'istanza del TSP è **metrica** se le sue distanze soddisfano la **disuguaglianza triangolare**

$$d_{ij} \leq d_{ik} + d_{kj} \quad \text{per ogni terna di città distinte } i, j, k \in \mathcal{V}.$$

Il **problema del commesso viaggiatore metrico** (TSP metrico) è il TSP ristretto alle istanze metriche.

5.2.1 Che cosa fa il corso su questo problema

Tecnica	Dove
Definizione	Cap. 6 «Problemi di routing»
Formulazioni	Cap. 13 «Formulazioni dei problemi di routing»
Euristiche	Cap. 30 «Euristiche costruttive»
Approssimazione	Cap. 39 «TSP: il caso generale e il caso metrico»
Esercizi	Esercizio 6.4; Cap. 39 (2 esercizi)

5.3 Problema del commesso viaggiatore asimmetrico (ATSP)

Definizione 5.3: Problema del commesso viaggiatore asimmetrico

Nel **problema del commesso viaggiatore asimmetrico** (ATSP) le distanze non devono essere simmetriche: d_{ij} può essere diversa da d_{ji} . La definizione di giro e della sua lunghezza $L(\sigma)$ non cambia, ma un giro e lo stesso giro percorso al contrario sono soluzioni diverse.

5.3.1 Che cosa fa il corso su questo problema

Tecnica	Dove
Definizione	Cap. 6 «Problemi di routing»
Formulazioni	Cap. 13 «Formulazioni dei problemi di routing»
Esercizi	Esercizio 6.5; Esercizio 13.6

5.4 Problema del ciclo hamiltoniano (HCP)

Definizione 5.4: Problema del ciclo hamiltoniano

Il **problema del ciclo hamiltoniano** chiede se un grafo $G = (V, E)$ ha un ciclo che passa per tutti i suoi vertici.

5.4.1 Che cosa fa il corso su questo problema

Tecnica	Dove
Definizione	Questa pagina
Approssimazione	Cap. 39 «TSP: il caso generale e il caso metrico»
Esercizi	Esercizio 39.3

Problemi di scheduling

- Dati n lavori con i loro tempi di processamento e m macchine parallele, una soluzione assegna ogni lavoro a una macchina, che lo esegue senza interruzioni.
- L'obiettivo è minimizzare il makespan $C_{\max}$, l'istante di completamento dell'ultimo lavoro; nella notazione a tre campi $\alpha || C_{\max}$ il campo α descrive le macchine: identiche (P), un numero fissato di macchine identiche (Pm), con velocità diverse (Q) o scorrelate (R).

Problema	Sigla	Differenza
Scheduling su macchine parallele identiche	$P C_{\max}$	Il problema centrale della famiglia.
Scheduling su un numero fissato di macchine identiche	$Pm C_{\max}$	Caso particolare di $P C_{\max}$ con m fissato; con $m = 2$ chiede di dividere i lavori in due insiemi con carichi il più possibile uguali.
Scheduling su macchine parallele con velocità diverse	$Q C_{\max}$	Generalizza $P C_{\max}$, che è il caso di macchine con la stessa velocità.
Scheduling su macchine parallele scorrelate	$R C_{\max}$	Generalizza $Q C_{\max}$ e $P C_{\max}$.

6.1 Scheduling su macchine parallele identiche ($P || C_{\max}$)

Definizione 6.1: Scheduling su macchine parallele identiche

Dati:

1. un insieme di m macchine identiche,
2. un insieme $\mathcal{J} = \{1, 2, \dots, n\}$ di n lavori,
3. per ogni lavoro $j \in \mathcal{J}$, un tempo di processamento $t_j \in \mathbb{Z}_{\geq 1}$,

il problema $P || C_{\max}$ chiede di assegnare ogni lavoro a una macchina, in modo da minimizzare il tempo necessario per completare tutti i lavori, cioè il massimo tempo di processamento totale assegnato a una macchina.

Ipotesi

- Facciamo la seguente ipotesi:

1. Ci sono più lavori che macchine (altrimenti ogni lavoro ha una macchina tutta per sé, e il makespan ottimo è $\max_j t_j$). Formalmente:

$$n > m$$

6.1.1 Che cosa fa il corso su questo problema

Tecnica	Dove
Definizione	Cap. 7 «Problemi di scheduling»
Formulazioni	Cap. 14 «Formulazioni dei problemi di scheduling»
Euristiche	Cap. 29 «Algoritmi euristici: definizioni»; Cap. 30 «Euristiche costruttive»; Cap. 31 «Ricerca locale»; Cap. 32 «Metaeuristiche»
Euristiche MIP	Cap. 33 «Euristiche MIP per i problemi classici»
Approssimazione	Cap. 34 «Principi degli algoritmi di approssimazione»; Cap. 40 «Approssimazione per $P C_{\max}$ »
Esercizi	Cap. 7 (5 esercizi); Cap. 14 (5 esercizi); Cap. 29 (2 esercizi); Cap. 30 (3 esercizi); Cap. 31 (3 esercizi); Cap. 32 (3 esercizi); Esercizio 33.5; Esercizio 34.2; Cap. 40 (3 esercizi)

Proprietà

- NP-difficile in senso forte quando m fa parte dell'input; NP-difficile in senso debole per un numero fissato di macchine (Cap. 1).
- Lower bound $\max\{\max_j t_j, \lceil \sum_j t_j / m \rceil\}$; non è sempre raggiunto: (4, 3, 3) su due macchine dà 5, l'ottimo è 6 (Cap. 7).
- Esiste un makespan al più T se e solo se i lavori entrano in m contenitori di capacità T (Cap. 7).

Formulazioni

- Il modello ILP $_{P||C_{\max}}$ ha nm variabili binarie x_{jh} e la variabile continua $C_{\max}$ (Cap. 14).
- Il bound del rilassamento lineare è $\sum_j t_j / m$; il suo rapporto con l'ottimo è almeno $1/m$, e $1/m$ è raggiunto al limite (Cap. 14).
- Le disuguaglianze valide rafforzano il modello; l'ordinamento dei carichi rompe la simmetria solo quando i carichi sono distinti (Cap. 14).

Euristiche

- Sull'istanza del Cap. 7 (tre macchine, ottimo 9) LIST con i lavori più brevi per primi dà 12 e LPT dà 11 (Cap. 30).

- La soluzione di LPT è un ottimo locale per la mossa move, mentre due swap raggiungono 9; l'obiettivo ausiliario $\sum_h L_h^2$ dà una pendenza ai plateau (Cap. 31).
- Tabu search e VNS raggiungono 9 sulla stessa istanza (Cap. 32).

Euristiche MIP

- Sull'istanza del Cap. 7 il diving frazionario si ferma a 10, il diving guidato da LPT trova 9, la feasibility pump restituisce 11 (Cap. 33).
- RINS, local branching e proximity search, partendo da LPT, raggiungono 9; proximity search ne dimostra anche l'ottimalità (Cap. 33).

Approssimazione

- LIST è un algoritmo $(2 - \frac{1}{m})$ -approssimato per ogni lista, tight (Cap. 40).
- LPT è un algoritmo $(\frac{4}{3} - \frac{1}{3m})$ -approssimato, tight sulla famiglia di Graham; per $m = 3$ è l'istanza del Cap. 7 (Cap. 40).

6.2 Scheduling su un numero fissato di macchine identiche ($Pm||C_{\max}$)

Definizione 6.2: Scheduling su un numero fissato di macchine identiche

Il problema $P||C_{\max}$ in cui il numero m di macchine è una costante, per esempio $m = 2$, e solo i lavori $\mathcal{J}$ e i tempi di processamento $t_j \in \mathbb{Z}_{\geq 1}$ formano l'input.

6.2.1 Che cosa fa il corso su questo problema

Tecnica	Dove
Definizione	Questa pagina
Programmazione dinamica	Cap. 16 «Principi della programmazione dinamica»
Esercizi	Esercizio 16.3

6.3 Scheduling su macchine parallele con velocità diverse ($Q||C_{\max}$)

Definizione 6.3: Scheduling su macchine parallele con velocità diverse

Nella notazione a tre campi $\alpha|\beta|\gamma$, l'ambiente delle macchine $\alpha = Q$ indica macchine parallele con velocità diverse; $Q||C_{\max}$ assegna lavori senza caratteristiche aggiuntive a tali macchine, minimizzando il makespan $C_{\max}$.

6.3.1 Che cosa fa il corso su questo problema

Tecnica	Dove
Definizione	Cap. 7 «Problemi di scheduling»

6.4 Scheduling su macchine parallele scorrelate ($R||C_{\max}$)

Definizione 6.4: Scheduling su macchine parallele scorrelate

Nella notazione a tre campi $\alpha|\beta|\gamma$, l'ambiente delle macchine $\alpha = R$ indica macchine parallele scorrelate; $R||C_{\max}$ assegna lavori senza caratteristiche aggiuntive a tali macchine, minimizzando il makespan $C_{\max}$.

6.4.1 Che cosa fa il corso su questo problema

Tecnica	Dove
Definizione	Cap. 7 «Problemi di scheduling»

Problemi di lot sizing

- Dato un insieme $T = \{1, 2, \dots, m\}$ di periodi con una domanda d_t in ogni periodo $t \in T$, una soluzione fissa i periodi con un setup e la quantità prodotta in ciascuno di essi, in modo che ogni domanda sia soddisfatta.
- L'obiettivo è minimizzare il costo totale di setup, produzione e mantenimento a scorta; i membri della famiglia differiscono per le capacità, per il numero di prodotti e per il trattamento della scorta iniziale, dello straordinario e della domanda non soddisfatta.

Problema	Sigla	Differenza
Lot sizing capacitato	CLS	Il problema centrale della famiglia.
Problema del lot sizing senza capacità	ULS	Caso particolare del CLS in cui le capacità non sono vincolanti, per esempio $cap_t \geq \sum_u d_u$.
Lot sizing capacitato con capacità costante	CLS-C	Caso particolare del CLS; la proprietà zero-inventory ordering è sostituita da una più debole: fra due periodi con scorta vuota al più un periodo produce una quantità strettamente compresa fra 0 e C .
Lot sizing con più prodotti e tempi di setup	MLS	Generalizza il CLS, che è il caso di un solo prodotto con $vt_t = 1$ e nessun tempo di setup.
Lot sizing con scorta iniziale	LS-SI	Toglie l'ipotesi di scorta iniziale vuota del CLS.
Lot sizing con straordinario	LS-OT	Sostituisce la capacità fissa cap_t del CLS con due livelli di capacità con costi diversi.
Lot sizing con domanda non soddisfatta	LS-UD	Rilassa i vincoli di domanda del CLS, con una penalità nell'obiettivo.

7.1 Lot sizing capacitato (CLS)

Definizione 7.1: Lot sizing capacitato

Dati:

1. un insieme $T = \{1, 2, \dots, m\}$ di m periodi di tempo,
2. per ogni periodo $t \in T$, una domanda $d_t \in \mathbb{Z}_{\geq 1}$ e una capacità $cap_t \in \mathbb{Z}_{\geq 1}$, entrambe misurate in unità di un prodotto,
3. per ogni periodo $t \in T$, tre costi interi non negativi: il costo di setup sc_t , il costo variabile di produzione vc_t per unità prodotta e il costo di mantenimento a scorta hc_t per unità tenuta in scorta alla fine del periodo,

il **problema del lot sizing capacitato (CLS)** chiede di decidere quanto produrre in ogni periodo, in modo che ogni domanda sia soddisfatta con ciò che si produce e ciò che è in scorta, le capacità siano rispettate, e il costo totale di setup, produzione e mantenimento a scorta sia minimo.

Ipotesi

- Facciamo le seguenti ipotesi:
 1. La scorta è vuota all'inizio e alla fine dell'orizzonte. Lasciare scorta alla fine aggiunge solo costo, quindi nessun piano ottimo ne ha bisogno; una scorta iniziale è una variante del problema (Sezione).
 2. Le capacità permettono di soddisfare le domande, periodo dopo periodo (altrimenti non esiste alcun piano ammissibile). Formalmente:

$$\sum_{u=1}^t cap_u \geq \sum_{u=1}^t d_u, \quad \forall t \in T$$

3. Le quantità sono intere. Con dati interi, una volta fissati i periodi con un setup, ciò che resta è un problema di flusso di costo minimo, che ha una soluzione ottima intera: restringere le quantità agli interi non cambia il costo ottimo.

7.1.1 Che cosa fa il corso su questo problema

Tecnica	Dove
Definizione	Cap. 8 «Problemi di lot sizing»
Formulazioni	Cap. 15 «Formulazioni dei problemi di lot sizing»
Programmazione dinamica	Cap. 16 «Principi della programmazione dinamica»; Cap. 26 «PD per il problema del lot sizing capacitato»
Euristiche MIP	Cap. 33 «Euristiche MIP per i problemi classici»
Esercizi	Cap. 8 (5 esercizi); Cap. 15 (5 esercizi); Cap. 26 (2 esercizi)

Proprietà

- NP-difficile con capacità generali; senza capacità (ULS) è polinomiale (Cap. 1).
- Lower bound: la somma dei k costi di setup più piccoli, con k il minimo numero di periodi le cui capacità coprono la domanda totale (Cap. 8).
- Lot-for-lot e accorpamenti a mano: $250 \rightarrow 210 \rightarrow 180$ sull'istanza del Cap. 8, dove 180 è ottimo (Cap. 8).

Formulazioni

- Formulazione big-M: il valore forte $M_t = \min\{cap_t, D_{t,m}\}$ alza il bound del rilassamento lineare sull'istanza del Cap. 8 da 99.65 a 157.5 (Cap. 15).
- La formulazione a cammino minimo dà il bound 170, il più forte del capitolo (Cap. 15).

Programmazione dinamica

- Senza capacità un piano ottimo produce solo con scorta vuota (Wagner e Whitin): una PD sull'ultimo lotto di produzione risolve l'ULS in $\Theta(m^2)$ passi (Cap. 25).
- Con capacità lo stato è (periodo, scorta): $O(m D C)$ passi, un tempo pseudo-polinomiale (Cap. 26).

Euristiche MIP

- Sull'istanza senza capacità del corso di modellazione MIP relax-and-fix trova l'ottimo 170, mentre l euristica del minimo costo unitario dà 200 (Cap. 33).
- Su un'istanza capacitata relax-and-fix si ferma a 170, e fix-and-optimize raggiunge l'ottimo 160 (Cap. 33).

7.2 Problema del lot sizing senza capacità (ULS)

Definizione 7.2: Problema del lot sizing senza capacità

Dati:

1. un insieme $T = \{1, 2, \dots, m\}$ di $m \in \mathbb{Z}_{\geq 1}$ periodi,
2. per ogni periodo $t \in T$, una *domanda* $d_t \in \mathbb{Z}_{\geq 1}$,
3. per ogni periodo $t \in T$, un *costo di setup* $sc_t \in \mathbb{Z}_{\geq 0}$, un *costo variabile di produzione* $vc_t \in \mathbb{Z}_{\geq 0}$ per unità, e un *costo di mantenimento a scorta* $hc_t \in \mathbb{Z}_{\geq 0}$ per unità in scorta alla fine del periodo t ,

il **problema del lot sizing senza capacità** (ULS) chiede di decidere quante unità produrre in ogni periodo, in modo che, partendo con la scorta vuota e terminando con la scorta vuota, la domanda di ogni periodo sia soddisfatta, e il costo totale di setup, produzione e mantenimento a scorta sia minimo.

Ipotesi

- Facciamo le seguenti ipotesi:
 1. C'è un solo prodotto, e la scorta è vuota all'inizio e alla fine dell'orizzonte: l'ULS è il CLS di un solo prodotto senza capacità e senza scorta iniziale.
 2. Le quantità sono intere. Con dati interi, una volta fissati i periodi con un setup, resta un problema di flusso di costo minimo, che ha una soluzione ottima intera: restringere le quantità agli interi non cambia il costo ottimo.

7.2.1 Che cosa fa il corso su questo problema

Tecnica	Dove
Definizione	Questa pagina
Formulazioni	Cap. 15 «Formulazioni dei problemi di lot sizing»
Programmazione dinamica	Cap. 16 «Principi della programmazione dinamica»; Cap. 25 «PD per il problema del lot sizing senza capacità»
Euristiche MIP	Cap. 33 «Euristiche MIP per i problemi classici»
Esercizi	Esercizio 8.4; Esercizio 15.6; Cap. 25 (3 esercizi)

7.3 Lot sizing capacitato con capacità costante (CLS-C)

Definizione 7.3: Lot sizing capacitato con capacità costante

Il CLS in cui la capacità è la stessa in tutti i periodi, $cap_t = C$ per ogni $t \in T$.

7.3.1 Che cosa fa il corso su questo problema

Tecnica	Dove
Definizione	Questa pagina
Programmazione dinamica	Cap. 26 «PD per il problema del lot sizing capacitato»

7.4 Lot sizing con più prodotti e tempi di setup (MLS)

Definizione 7.4: Lot sizing con più prodotti e tempi di setup

Un insieme P di prodotti condivide la risorsa di ogni periodo: d_{pt} , sc_{pt} , vc_{pt} e hc_{pt} sono i dati del prodotto $p \in P$ nel periodo $t \in T$; produrre un'unità di p richiede il tempo $vt_{pt} \in \mathbb{Z}_{\geq 1}$, un setup richiede il tempo $st_{pt} \in \mathbb{Z}_{\geq 0}$, e la capacità cap_t è un tempo.

7.4.1 Che cosa fa il corso su questo problema

Tecnica	Dove
Definizione	Questa pagina
Formulazioni	Cap. 15 «Formulazioni dei problemi di lot sizing»
Esercizi	Esercizio 15.7

7.5 Lot sizing con scorta iniziale (LS-SI)

Definizione 7.5: Lot sizing con scorta iniziale

La quantità si_p del prodotto p disponibile all'inizio dell'orizzonte è una decisione, acquistata al costo fc_p per unità; il bilancio del primo periodo diventa $si_p + x_{p1} = d_{p1} + s_{p1}$.

7.5.1 Che cosa fa il corso su questo problema

Tecnica	Dove
Definizione	Cap. 8 «Problemi di lot sizing»
Formulazioni	Cap. 15 «Formulazioni dei problemi di lot sizing»

7.6 Lot sizing con straordinario (LS-OT)

Definizione 7.6: Lot sizing con straordinario

La capacità non è fissa: il tempo di lavoro in orario normale costa cw_t per unità fino a cap_t^N , e lo straordinario costa co_t per unità fino a cap_t^O .

7.6.1 Che cosa fa il corso su questo problema

Tecnica	Dove
Definizione	Cap. 8 «Problemi di lot sizing»
Formulazioni	Cap. 15 «Formulazioni dei problemi di lot sizing»
Esercizi	Esercizio 15.8

7.7 Lot sizing con domanda non soddisfatta (LS-UD)

Definizione 7.7: Lot sizing con domanda non soddisfatta

Una parte della domanda può restare non soddisfatta, con una penalità cu_{pt} per unità.

7.7.1 Che cosa fa il corso su questo problema

Tecnica	Dove
Definizione	Cap. 8 «Problemi di lot sizing»
Formulazioni	Cap. 15 «Formulazioni dei problemi di lot sizing»
Esercizi	Esercizio 15.9

Problemi misti

- Nel problema del taglio delle aste un'asta di lunghezza n è tagliata in pezzi di lunghezza intera ℓ con profitti p_ℓ ; la lunghezza totale dei pezzi è uguale a n e il loro profitto totale è massimo.
- Nel problema del cambio monete si restituisce un resto c con monete dei tagli $w_1, \dots, w_n$, ciascuno disponibile in quantità illimitata; il valore totale è uguale a c e il numero di monete è minimo.

Problema	Sigla	Differenza
Problema del taglio delle aste	RCP	Uno zaino illimitato con un tipo di oggetto per ogni lunghezza ℓ dei pezzi, peso ℓ , profitto p_ℓ e capacità n , in cui la lunghezza totale deve essere esattamente n .
Problema del cambio monete	CCP	Come l'UKP ha copie illimitate di ogni tipo di oggetto, ma chiede esattamente c e minimizza il numero di monete, mentre l'UKP chiede al più c e massimizza il profitto.

8.1 Problema del taglio delle aste (RCP)

Definizione 8.1: Problema del taglio delle aste

Dati:

1. un'asta di *lunghezza* $n \in \mathbb{Z}_{\geq 1}$,
2. un insieme $\{1, 2, \dots, n\}$ di n possibili lunghezze dei pezzi,
3. per ogni lunghezza $\ell \in \{1, 2, \dots, n\}$, un *profitto* $p_\ell \in \mathbb{Z}_{\geq 1}$,

il **problema del taglio delle aste** (RCP) chiede di decidere quanti pezzi di ogni lunghezza tagliare, in modo che la lunghezza totale dei pezzi ottenuti sia uguale alla lunghezza dell'asta, e il profitto totale dei pezzi ottenuti sia massimo.

8.1.1 Che cosa fa il corso su questo problema

Tecnica	Dove
Definizione	Questa pagina
Programmazione dinamica	Cap. 16 «Principi della programmazione dinamica»; Cap. 28 «PD per il problema del taglio delle aste»
Esercizi	Cap. 28 (5 esercizi)

8.2 Problema del cambio monete (CCP)

Definizione 8.2: Problema del cambio monete

Dati:

1. un vettore $w \in \mathbb{Z}_{\geq 1}^n$ di n tagli, ciascuno disponibile in quantità illimitata,
2. un resto intero $c \in \mathbb{Z}_{\geq 0}$ da restituire,

il **problema del cambio monete** (CCP) chiede di decidere quante monete di ogni taglio usare in modo che il loro valore totale sia esattamente uguale a c , e il numero totale di monete usate sia minimo. Se nessuna combinazione di monete restituisce esattamente c , l'istanza è *non ammissibile*.

8.2.1 Che cosa fa il corso su questo problema

Tecnica	Dove
Definizione	Questa pagina
Programmazione dinamica	Cap. 16 «Principi della programmazione dinamica»; Cap. 27 «PD per il problema del cambio monete»
Esercizi	Cap. 27 (5 esercizi)