

SAPIENZA
UNIVERSITÀ DI ROMA

Algoritmi euristici

Esercizi svolti

Esercizi svolti sui capitoli della dispensa, algoritmi euristici: ogni esercizio è seguito dalla sua soluzione completa.

Fabio Furini

fabiofurini.github.io

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

Versione online e materiale scaricabile:
fabiofurini.github.io/algoritmi-euristici/
github.com/fabiofurini/algoritmi-euristici

Anno accademico 2026–2027

Indice

1 Algoritmi euristici: definizioni **2**

2 Euristiche costruttive **10**

2.1 Zaino: ordinamenti degli oggetti 10

2.2 Bin packing: euristiche decrescenti 16

2.3 $P||C_{\max}$: LIST e LPT 23

2.4 TSP: euristiche costruttive 27

2.5 Colorazione: greedy e DSATUR 32

3 Ricerca locale **36**

3.1 Zaino 36

3.2 Bin packing 45

3.3 TSP: 2-opt 53

3.4 $P||C_{\max}$ 61

4 Metaeuristiche **70**

4.1 Simulated annealing 70

4.2 Tabu search 76

4.3 Iterated local search e VNS 79

4.4 GRASP 82

4.5 Large neighbourhood search 86

4.6 Algoritmi genetici 89

5 Euristiche MIP per i problemi classici **92**

5.1 Arrotondamento e diving 92

5.2 RINS, local branching e proximity search 95

Capitolo 1

Algoritmi euristici: definizioni

Esercizio 1.1 — Gap ed errore relativo (1)

Consideriamo uno zaino di capacità $c = 12$ e $n = 5$ oggetti con i seguenti profitti e pesi:

	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$
p_j	20	18	14	9	6
w_j	6	5	4	3	2

Un algoritmo euristico restituisce gli oggetti $\{1, 2\}$.

Domande

1. Verificare che la soluzione euristica sia ammissibile e darne il valore $\mathscr{A}(I)$. È un bound primale o duale, un upper bound o un lower bound?
2. Calcolare il bound duale $\mathrm{UB}(I)$ dato dal rilassamento lineare, arrotondato per difetto.
3. Calcolare il gap della soluzione euristica.
4. Trovare il valore ottimo $z(I)$ per enumerazione, calcolare l'errore relativo e verificare che sia al più il gap.

Soluzione

Per un valore euristico $\mathscr{A}(I)$ e un bound duale, il gap è la differenza dei due bound divisa per il minore:

$$gap(I) = \frac{\mathscr{A}(I) - \mathrm{LB}(I)}{\mathrm{LB}(I)} \quad \text{(minimo)},$$
$$gap(I) = \frac{\mathrm{UB}(I) - \mathscr{A}(I)}{\mathscr{A}(I)} \quad \text{(massimo)},$$

e l'errore relativo è $\frac{\mathscr{A}(I)-z(I)}{z(I)}$ (minimo) oppure $\frac{z(I)-\mathscr{A}(I)}{z(I)}$ (massimo). L'errore relativo è al più il gap.

1. Il peso è $6+5=11 \leq 12$, quindi la soluzione è ammissibile, con $\mathscr{A}(I) = 20+18 = 38$. Il KP è un problema di massimo: $\mathscr{A}(I)$ è un bound primale, e un lower bound su $z(I)$.

2. Le efficienze $\frac{p_j}{w_j}$ sono $\frac{10}{3}$, $\frac{18}{5}$, $\frac{7}{2}$, 3, 3, quindi gli oggetti ordinati per efficienza non crescente sono $(2,3,1,4,5)$. Il rilassamento inserisce gli oggetti 2, 3 (peso 9) e una frazione $\frac{1}{2}$ dell'oggetto 1:

$$z(I) \leq 18 + 14 + 20 \cdot \frac{1}{2} = 42.$$

   I profitti sono interi, quindi $\mathrm{UB}(I) = \lfloor 42 \rfloor = 42$.

3. $$gap(I) = \frac{\mathrm{UB}(I) - \mathscr{A}(I)}{\mathscr{A}(I)} = \frac{42-38}{38} = \frac{2}{19} \approx 10.5\%.$$

4. Enumerando i $2^5 = 32$ sottoinsiemi si ottiene $z(I) = 41$, raggiunto da $\{2,3,4\}$ (peso 12). L'errore relativo è

$$\frac{z(I) - \mathscr{A}(I)}{z(I)} = \frac{41-38}{41} = \frac{3}{41} \approx 7.3\% \;\leq\; 10.5\%.$$

   Il gap sovrastima l'errore perché $\mathrm{UB}(I) > z(I)$, e perché divide per $\mathscr{A}(I) < z(I)$.

Esercizio 1.2 — Gap per problemi di minimo (1)

Consideriamo le tre istanze e le soluzioni euristiche seguenti.

(a) BPP con $c = 10$ e pesi $(6,5,4,4,3,3,2,2)$; schemi di riempimento euristici $\{1\}$, $\{2,3\}$, $\{4,5\}$, $\{6,7,8\}$.

(b) BPP con $c = 10$ e pesi $(6,6,6,5,2)$; schemi di riempimento euristici $\{1\}$, $\{2\}$, $\{3\}$, $\{4\}$, $\{5\}$.

(c) $P\|C_{\max}$ con $m = 3$ e tempi di lavorazione $(7,6,5,4,3,3,2)$; schedulazione euristica $J_1 = \{1,4\}$, $J_2 = \{2,5,6\}$, $J_3 = \{3,7\}$.

Domande

1. Verificare che le tre soluzioni euristiche siano ammissibili e darne i valori $\mathscr{A}(I)$.

2. Calcolare i bound duali $\mathrm{LB}(I) = \left\lceil \frac{\sum_j w_j}{c} \right\rceil$ per il BPP e $\mathrm{LB}(I) = \max\left\{ \max_j t_j, \left\lceil \frac{\sum_j t_j}{m} \right\rceil \right\}$ per $P\|C_{\max}$, e i gap.

3. Calcolare i valori ottimi (Gurobi per il BPP, enumerazione per $P\|C_{\max}$) e gli errori relativi. Quando il gap e l'errore sono uguali?

Soluzione

Per un valore euristico $\mathscr{A}(I)$ e un bound duale, il gap è la differenza dei due bound divisa per il minore:

$$gap(I) = \frac{\mathscr{A}(I) - \mathrm{LB}(I)}{\mathrm{LB}(I)} \quad \text{(minimo)},$$
$$gap(I) = \frac{\mathrm{UB}(I) - \mathscr{A}(I)}{\mathscr{A}(I)} \quad \text{(massimo)},$$

e l'errore relativo è $\frac{\mathscr{A}(I)-z(I)}{z(I)}$ (minimo) oppure $\frac{z(I)-\mathscr{A}(I)}{z(I)}$ (massimo). L'errore relativo è al più il gap.

1. (a) I pesi degli schemi sono 6, 9, 7, 7, tutti al più 10: $\mathscr{A}(I) = 4$.
   (b) Ogni oggetto è da solo nel suo contenitore: $\mathscr{A}(I) = 5$.
   (c) I carichi sono 11, 12, 7: $\mathscr{A}(I) = 12$.

   I tre problemi sono tutti problemi di minimo, quindi questi valori sono upper bound su $z(I)$.

2. (a) $\mathrm{LB}(I) = \lceil \frac{29}{10} \rceil = 3$ e $gap(I) = \frac{4-3}{3} \approx 33.3\%$.
   (b) $\mathrm{LB}(I) = \lceil \frac{5}{2} \rceil = 3$ e $gap(I) = \frac{5-3}{3} \approx 66.7\%$.
   (c) $\mathrm{LB}(I) = \max\{7, \lceil 30/3 \rceil\} = 10$ e $gap(I) = \frac{12-10}{10} = \frac{1}{5} \approx 20.0\%$.

3. (a) $z(I) = 3$, per esempio con $\{1,3\}$, $\{2,5,7\}$, $\{4,6,8\}$; l'errore è $\frac{4-3}{3} \approx 33.3\%$.
   (b) $z(I) = 4$: i tre oggetti di peso 6 richiedono tre contenitori, e l'oggetto di peso 5 non entra con nessuno di essi; l'errore è $\frac{5-4}{4} = \frac{1}{4} = 25.0\%$, molto minore del gap.
   (c) $z(I) = 10$, per esempio con $\{1,5\}$, $\{2,4\}$, $\{3,6,7\}$; l'errore è $\frac{1}{5} \approx 20.0\%$.

   Il gap è uguale all'errore relativo esattamente quando il bound duale è uguale a $z(I)$, come in (a) e (c). In (b) il bound è debole, e il gap sovrastima l'errore.

Esercizio 1.3 — Un bound duale per il TSP (1)

Consideriamo un'istanza del TSP con $n = 5$ città e le distanze simmetriche

d_{ij}	1	2	3	4	5
1	–	4	10	2	8
2	4	–	11	2	5
3	10	11	–	4	2
4	2	2	4	–	2
5	8	5	2	2	–

Due algoritmi euristici restituiscono i giri $\sigma_1 = (1,2,3,4,5)$ e $\sigma_2 = (1,4,2,5,3)$.

Domande

1. Calcolare le lunghezze $L(\sigma_1)$ e $L(\sigma_2)$.
2. Calcolare il bound duale del capitolo basato sulle due distanze minime da ogni città.

3. Calcolare i gap dei due giri.
4. Trovare il giro ottimo enumerando i giri distinti e calcolare gli errori relativi. Confrontarli con i gap.

Soluzione

Per un valore euristico $\mathscr{A}(I)$ e un bound duale, il gap è la differenza dei due bound divisa per il minore:

$$gap(I) = \frac{\mathscr{A}(I) - \mathrm{LB}(I)}{\mathrm{LB}(I)} \quad \text{(minimo)},$$
$$gap(I) = \frac{\mathrm{UB}(I) - \mathscr{A}(I)}{\mathscr{A}(I)} \quad \text{(massimo)},$$

e l'errore relativo è $\frac{\mathscr{A}(I)-z(I)}{z(I)}$ (minimo) oppure $\frac{z(I)-\mathscr{A}(I)}{z(I)}$ (massimo). L'errore relativo è al più il gap.

1. $L(\sigma_1) = 4+11+4+2+8 = 29$ e $L(\sigma_2) = 2+2+5+2+10 = 21$.
2. Le due distanze minime da ogni città sono $(2,4)$ per la città 1, $(2,4)$ per la città 2, $(2,4)$ per la città 3, $(2,2)$ per la città 4, $(2,2)$ per la città 5. Quindi
$$\mathrm{LB}(I) = \left\lceil \frac{1}{2}(6+6+6+4+4) \right\rceil = \lceil 13 \rceil = 13.$$
3. $gap_1 = \frac{29-13}{13} = \frac{16}{13} \approx 123.1\%$ e $gap_2 = \frac{21-13}{13} = \frac{8}{13} \approx 61.5\%$.
4. Ci sono $\frac{(n-1)!}{2} = 12$ giri distinti, e il più corto è $(1,2,5,3,4)$, di lunghezza $z(I) = 17$. Gli errori relativi sono $\frac{29-17}{17} \approx 70.6\%$ e $\frac{21-17}{17} \approx 23.5\%$, minori dei gap, perché $\mathrm{LB}(I) = 13 < z(I) = 17$. Entrambe le misure classificano σ_2 meglio di σ_1.

Esercizio 1.4 — Intorni e ottimi locali (1)

Consideriamo uno zaino di capacità $c = 11$ e $n = 5$ oggetti:

	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$
p_j	12	10	9	7	4
w_j	6	5	4	3	2

Le soluzioni sono rappresentate da vettori binari $\boldsymbol{x} \in \{0,1\}^n$. L'intorno **flip** aggiunge o toglie un oggetto; l'intorno **swap** toglie un oggetto scelto e aggiunge un oggetto non scelto. Contano solo i vicini ammissibili.

Domande

1. Dare la dimensione dello spazio di ricerca e il numero dei suoi elementi che sono soluzioni ammissibili.

2. Considerare la soluzione $\{1, 3\}$. Elencarne i vicini flip e i vicini swap, e stabilire se è un ottimo locale per ciascun intorno.

3. Fare lo stesso per la soluzione $\{1, 2\}$.

4. Trovare l'ottimo globale per enumerazione. Uno dei due intorni è esatto su questa istanza?

Soluzione

1. Lo spazio di ricerca è $\{0, 1\}^5$, con $2^5 = 32$ vettori; 20 di essi rispettano la capacità. Gli altri la superano: la rappresentazione può descrivere oggetti inammissibili.

2. La soluzione ha peso 10 e profitto 21.

   Vicini flip:

vicino	mossa	peso	profitto
$\{3\}$	esce 1	4	9
$\{1, 2, 3\}$	entra 2	15	— (inammissibile)
$\{1\}$	esce 3	6	12
$\{1, 3, 4\}$	entra 4	13	— (inammissibile)
$\{1, 3, 5\}$	entra 5	12	— (inammissibile)

   Nessun vicino flip ammissibile ha profitto maggiore di 21: la soluzione è un ottimo locale per l'intorno flip.

   Vicini swap:

vicino	mossa	peso	profitto
$\{2, 3\}$	esce 1, entra 2	9	19
$\{3, 4\}$	esce 1, entra 4	7	16
$\{3, 5\}$	esce 1, entra 5	6	13
$\{1, 2\}$	esce 3, entra 2	11	22
$\{1, 4\}$	esce 3, entra 4	9	19
$\{1, 5\}$	esce 3, entra 5	8	16

   Il vicino $\{1, 2\}$ ha profitto $22 > 21$: la soluzione non è un ottimo locale per l'intorno swap.

3. La soluzione ha peso 11 e profitto 22.

   Vicini flip:

vicino	mossa	peso	profitto
$\{2\}$	esce 1	5	10
$\{1\}$	esce 2	6	12
$\{1, 2, 3\}$	entra 3	15	— (inammissibile)
$\{1, 2, 4\}$	entra 4	14	— (inammissibile)
$\{1, 2, 5\}$	entra 5	13	— (inammissibile)

Vicini swap:

vicino	mossa	peso	profitto
$\{2,3\}$	esce 1, entra 3	9	19
$\{2,4\}$	esce 1, entra 4	8	17
$\{2,5\}$	esce 1, entra 5	7	14
$\{1,3\}$	esce 2, entra 3	10	21
$\{1,4\}$	esce 2, entra 4	9	19
$\{1,5\}$	esce 2, entra 5	8	16

Nessun vicino ammissibile è migliore: $\{1,2\}$ è un ottimo locale per entrambi gli intorni.

4. L'ottimo globale è $\{1,4,5\}$, di peso 11 e profitto $z(I) = 23$. La soluzione $\{1,2\}$ è un ottimo locale per entrambi gli intorni con profitto $22 < 23$, quindi nessuno dei due intorni è esatto: per raggiungere l'ottimo da $\{1,2\}$ bisogna togliere un oggetto e aggiungerne due.

Esercizio 1.5 — Euristiche costruttive, ricerca locale e metaeuristiche (1)

Consideriamo $P\|C_{\max}$ con $m = 2$ macchine e $n = 6$ lavori:

	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$
t_j	8	4	4	6	3	7

Una soluzione è rappresentata dal vettore $\boldsymbol{mach}$, dove $mach_j$ è la macchina del lavoro j. Si propongono tre procedure:

(P1) considerare i lavori nell'ordine $1,\dots,n$ e assegnare ogni lavoro alla macchina con il carico corrente minimo (pareggi: indice di macchina minimo); non cambiare mai un assegnamento;

(P2) partendo da una soluzione, spostare un lavoro sull'altra macchina, scegliendo la mossa con il makespan minimo (pareggi: indice di lavoro minimo); ripetere finché il makespan diminuisce strettamente;

(P3) ripetere (P2) molte volte, ogni volta da una soluzione ottenuta assegnando ogni lavoro a una macchina casuale, e tenere la soluzione migliore trovata.

Domande

1. Dare la dimensione dello spazio di ricerca di questa rappresentazione. I suoi elementi sono tutti ammissibili?
2. Classificare le tre procedure usando le classi del capitolo.
3. Applicare (P1), e poi applicare (P2) partendo dalla sua soluzione. Mostrare i vicini esaminati a ogni passo.

4. Trovare il valore ottimo per enumerazione. La soluzione restituita da (P2) è un ottimo globale? Sarebbe un ottimo locale per l'intorno che scambia due lavori su macchine diverse?

5. Spiegare i ruoli dell'intensificazione e della diversificazione in (P2) e (P3).

Soluzione

1. Lo spazio di ricerca è $\{1,2\}^6$, con $m^n = 2^6 = 64$ vettori. Ogni vettore è una schedulazione ammissibile, perché le macchine non hanno capacità.

2. (P1) è un'euristica costruttiva: estende una soluzione parziale un lavoro alla volta e non rivede mai una scelta. (P2) è una ricerca locale con l'intorno che sposta un lavoro. (P3) è una metaeuristica: guida la ricerca locale oltre il primo ottimo locale facendola ripartire da altre soluzioni.

3. (P1) dà $\boldsymbol{mach} = (1,2,2,1,2,2)$, con carichi $(14,18)$ e makespan 18.

   Iterazione 1

   I vicini della soluzione di (P1) sono:

mossa	carichi	makespan
lavoro 1 sulla macchina 2	$(6,26)$	26
lavoro 2 sulla macchina 1	$(18,14)$	18
lavoro 3 sulla macchina 1	$(18,14)$	18
lavoro 4 sulla macchina 2	$(8,24)$	24
lavoro 5 sulla macchina 1	$(17,15)$	17
lavoro 6 sulla macchina 1	$(21,11)$	21

   Il migliore ha makespan $17 < 18$: la nuova soluzione è $\boldsymbol{mach} = (1,2,2,1,1,2)$.

   Iterazione 2

   I suoi vicini sono:

mossa	carichi	makespan
lavoro 1 sulla macchina 2	$(9,23)$	23
lavoro 2 sulla macchina 1	$(21,11)$	21
lavoro 3 sulla macchina 1	$(21,11)$	21
lavoro 4 sulla macchina 2	$(11,21)$	21
lavoro 5 sulla macchina 2	$(14,18)$	18
lavoro 6 sulla macchina 1	$(24,8)$	24

   Nessun vicino ha makespan minore di 17: (P2) si ferma in un ottimo locale.

4. L'enumerazione dà $z(I) = 16$, per esempio con $J_1 = \{1,2,3\}$ e $J_2 = \{4,5,6\}$. L'ottimo locale di (P2), di makespan 17, non è un ottimo globale. Scambiando i lavori 1 e 6 si ottengono carichi $(16,16)$ e makespan 16: per l'intorno di scambio la soluzione non è un ottimo locale. Essere un ottimo locale dipende dall'intorno.

5. (P2) è pura intensificazione: scende e si ferma sul fondo della prima valle che incontra. Le soluzioni di partenza casuali di (P3) sono diversificazione: portano la ricerca in altre regioni dello spazio di ricerca, dove un'altra ricerca locale può trovare una valle migliore. Tenere la soluzione migliore rende utili le esecuzioni ripetute.

Capitolo 2

Euristiche costruttive

2.1 Zaino: ordinamenti degli oggetti

Esercizio 2.1 — Ordinamenti greedy per il KP 1

Consideriamo uno zaino di capacità $c = 9$ e un insieme $\mathscr{I} = \{1, \dots, 5\}$ di 5 oggetti ($n = 5$), con i seguenti profitti e pesi:

	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$
p_j	4	15	11	8	2
w_j	3	8	7	2	5

Domande

1. Ordinare gli oggetti per profitto non crescente, per peso non decrescente e per efficienza non crescente $\frac{p_j}{w_j}$ (pareggi per indice).
2. Applicare l'algoritmo greedy con ciascuno dei tre ordinamenti (GR_p, GR_w, GR), mostrando la capacità residua rc dopo ogni passo, e fornire le tre soluzioni e i loro profitti.
3. Calcolare l'upper bound UB dato dal rilassamento lineare (oggetti per efficienza non crescente, l'oggetto critico preso in modo frazionario) e il gap $\frac{\lfloor \mathrm{UB} \rfloor - \mathscr{A}(I)}{\mathscr{A}(I)}$ della migliore delle tre soluzioni.
4. Sapendo che il valore ottimo è $z(I) = 19$, dire quali ordinamenti trovano una soluzione ottima, e calcolare l'errore relativo degli altri. Spiegare che cosa non funziona.

Soluzione

1. Le efficienze sono

$$\frac{p_1}{w_1} = \frac{4}{3} \approx 1.33, \quad \frac{p_2}{w_2} = \frac{15}{8} \approx 1.88, \quad \frac{p_3}{w_3} = \frac{11}{7} \approx 1.57,$$
$$\frac{p_4}{w_4} = \frac{8}{2} \approx 4.00, \quad \frac{p_5}{w_5} = \frac{2}{5} \approx 0.40$$

e i tre ordinamenti sono

criterio	ordinamento
profitto	$(2, 3, 4, 1, 5)$
peso	$(4, 1, 5, 3, 2)$
efficienza	$(4, 2, 3, 1, 5)$

2. GR_p, ordinamento $(2, 3, 4, 1, 5)$, partendo da $rc = 9$:

- oggetto 2 ($w_2 = 8 \leq rc = 9$): selezionato, $rc \leftarrow 1$;
- oggetto 3 ($w_3 = 7 > rc = 1$): non selezionato;
- oggetto 4 ($w_4 = 2 > rc = 1$): non selezionato;
- oggetto 1 ($w_1 = 3 > rc = 1$): non selezionato;
- oggetto 5 ($w_5 = 5 > rc = 1$): non selezionato;

Soluzione $\{2\}$, $GR_p(I) = 15$.

GR_w, ordinamento $(4, 1, 5, 3, 2)$, partendo da $rc = 9$:

- oggetto 4 ($w_4 = 2 \leq rc = 9$): selezionato, $rc \leftarrow 7$;
- oggetto 1 ($w_1 = 3 \leq rc = 7$): selezionato, $rc \leftarrow 4$;
- oggetto 5 ($w_5 = 5 > rc = 4$): non selezionato;
- oggetto 3 ($w_3 = 7 > rc = 4$): non selezionato;
- oggetto 2 ($w_2 = 8 > rc = 4$): non selezionato;

Soluzione $\{1, 4\}$, $GR_w(I) = 4 + 8 = 12$.

GR, ordinamento $(4, 2, 3, 1, 5)$, partendo da $rc = 9$:

- oggetto 4 ($w_4 = 2 \leq rc = 9$): selezionato, $rc \leftarrow 7$;
- oggetto 2 ($w_2 = 8 > rc = 7$): non selezionato;
- oggetto 3 ($w_3 = 7 \leq rc = 7$): selezionato, $rc \leftarrow 0$;
- oggetto 1 ($w_1 = 3 > rc = 0$): non selezionato;
- oggetto 5 ($w_5 = 5 > rc = 0$): non selezionato;

Soluzione $\{3, 4\}$, $GR(I) = 11 + 8 = 19$.

3. Nell'ordinamento per efficienza $(4, 2, 3, 1, 5)$ entrano gli oggetti $\{4\}$, con peso totale 2, e l'oggetto critico è 2, con capacità residua $rc = 7$. Quindi

$$\mathrm{UB} = 8 + 15 \cdot \frac{7}{8} = \frac{169}{8} \approx 21.12, \qquad \lfloor \mathrm{UB} \rfloor = 21.$$

   La migliore delle tre soluzioni ha profitto 19, e

$$gap(I) = \frac{21 - 19}{19} \approx 10.5\%.$$

4. La soluzione ottima è $\{3, 4\}$, con $z(I) = 19$ (il bound $\lfloor \mathrm{UB} \rfloor = 21$ non è raggiunto: qui il gap sovrastima l'errore). Solo GR la trova. GR_p prende l'oggetto 2, il più prezioso ma anche il più pesante, che non lascia spazio a nient'altro di valore: $GR_p(I) = 15$, errore relativo 21.1%. GR_w riempie lo zaino con i due oggetti più leggeri, 4 e 1, che valgono poco: $GR_w(I) = 12$, errore relativo 36.8%. Su questa istanza l'efficienza è il criterio giusto.

Esercizio 2.2 — Ordinamenti greedy per il KP 2

Consideriamo uno zaino di capacità $c = 13$ e un insieme $\mathscr{I} = \{1, \ldots, 5\}$ di 5 oggetti ($n = 5$), con i seguenti profitti e pesi:

	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$
p_j	9	4	8	6	7
w_j	7	2	9	5	6

Domande

1. Ordinare gli oggetti per profitto non crescente, per peso non decrescente e per efficienza non crescente $\frac{p_j}{w_j}$ (pareggi per indice).
2. Applicare l'algoritmo greedy con ciascuno dei tre ordinamenti (GR_p, GR_w, GR), mostrando la capacità residua rc dopo ogni passo, e fornire le tre soluzioni e i loro profitti.
3. Calcolare l'upper bound UB dato dal rilassamento lineare (oggetti per efficienza non crescente, l'oggetto critico preso in modo frazionario) e il gap $\frac{\lfloor \mathrm{UB} \rfloor - \mathscr{A}(I)}{\mathscr{A}(I)}$ della migliore delle tre soluzioni.
4. Stabilire se qualcuna delle tre soluzioni è ottima. Calcolare l'errore relativo delle altre, e spiegare perché l'ordinamento per efficienza fallisce su questa istanza.

Soluzione

1. Le efficienze sono

$$\frac{p_1}{w_1} = \frac{9}{7} \approx 1.29, \quad \frac{p_2}{w_2} = \frac{4}{2} \approx 2.00, \quad \frac{p_3}{w_3} = \frac{8}{9} \approx 0.89,$$
$$\frac{p_4}{w_4} = \frac{6}{5} \approx 1.20, \quad \frac{p_5}{w_5} = \frac{7}{6} \approx 1.17$$

e i tre ordinamenti sono

criterio	ordinamento
profitto	$(1, 3, 5, 4, 2)$
peso	$(2, 4, 5, 1, 3)$
efficienza	$(2, 1, 4, 5, 3)$

2. GR_p, ordinamento $(1, 3, 5, 4, 2)$, partendo da $rc = 13$:

- oggetto 1 ($w_1 = 7 \leq rc = 13$): selezionato, $rc \leftarrow 6$;
- oggetto 3 ($w_3 = 9 > rc = 6$): non selezionato;
- oggetto 5 ($w_5 = 6 \leq rc = 6$): selezionato, $rc \leftarrow 0$;
- oggetto 4 ($w_4 = 5 > rc = 0$): non selezionato;
- oggetto 2 ($w_2 = 2 > rc = 0$): non selezionato;

Soluzione $\{1, 5\}$, $GR_p(I) = 9 + 7 = 16$.

GR_w, ordinamento $(2, 4, 5, 1, 3)$, partendo da $rc = 13$:

- oggetto 2 ($w_2 = 2 \leq rc = 13$): selezionato, $rc \leftarrow 11$;
- oggetto 4 ($w_4 = 5 \leq rc = 11$): selezionato, $rc \leftarrow 6$;
- oggetto 5 ($w_5 = 6 \leq rc = 6$): selezionato, $rc \leftarrow 0$;
- oggetto 1 ($w_1 = 7 > rc = 0$): non selezionato;
- oggetto 3 ($w_3 = 9 > rc = 0$): non selezionato;

Soluzione $\{2, 4, 5\}$, $GR_w(I) = 4 + 6 + 7 = 17$.

GR, ordinamento $(2, 1, 4, 5, 3)$, partendo da $rc = 13$:

- oggetto 2 ($w_2 = 2 \leq rc = 13$): selezionato, $rc \leftarrow 11$;
- oggetto 1 ($w_1 = 7 \leq rc = 11$): selezionato, $rc \leftarrow 4$;
- oggetto 4 ($w_4 = 5 > rc = 4$): non selezionato;
- oggetto 5 ($w_5 = 6 > rc = 4$): non selezionato;
- oggetto 3 ($w_3 = 9 > rc = 4$): non selezionato;

Soluzione $\{1, 2\}$, $GR(I) = 9 + 4 = 13$.

3. Nell'ordinamento per efficienza $(2, 1, 4, 5, 3)$ entrano gli oggetti $\{2, 1\}$, con peso totale 9, e l'oggetto critico è 4, con capacità residua $rc = 4$. Quindi

$$\mathrm{UB} = 4 + 9 + 6 \cdot \frac{4}{5} = \frac{89}{5} \approx 17.80, \qquad \lfloor \mathrm{UB} \rfloor = 17.$$

La migliore delle tre soluzioni ha profitto 17, e

$$gap(I) = \frac{17 - 17}{17} = 0\%.$$

4. $GR_w(I) = 17 = \lfloor \mathrm{UB} \rfloor$, quindi la soluzione $\{2, 4, 5\}$ di GR_w è ottima e $z(I) = 17$. GR fallisce: dopo i due oggetti più efficienti 2 e 1 la capacità residua è $13 - 9 = 4$, troppo piccola per qualunque altro oggetto, e $GR(I) = 13$, errore relativo 23.5%. Anche $GR_p(I) = 16$ è peggiore di GR_w, con errore relativo $\frac{17-16}{17} \approx 5.9\%$. Tre oggetti leggeri riempiono esattamente lo zaino $(2 + 5 + 6 = 13)$: qui conta quanto bene gli oggetti riempiono la capacità, aspetto che l'efficienza ignora.

Esercizio 2.3 — Ordinamenti greedy per il KP 3

Consideriamo uno zaino di capacità $c = 10$ e un insieme $\mathscr{I} = \{1, \ldots, 5\}$ di 5 oggetti $(n = 5)$, con i seguenti profitti e pesi:

	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$
p_j	8	13	5	12	7
w_j	5	9	4	8	2

Domande

1. Ordinare gli oggetti per profitto non crescente, per peso non decrescente e per efficienza non crescente $\frac{p_j}{w_j}$ (pareggi per indice).
2. Applicare l'algoritmo greedy con ciascuno dei tre ordinamenti (GR_p, GR_w, GR), mostrando la capacità residua rc dopo ogni passo, e fornire le tre soluzioni e i loro profitti.
3. Calcolare l'upper bound UB dato dal rilassamento lineare (oggetti per efficienza non crescente, l'oggetto critico preso in modo frazionario) e il gap $\frac{\lfloor \mathrm{UB} \rfloor - \mathscr{A}(I)}{\mathscr{A}(I)}$ della migliore delle tre soluzioni.
4. Trovare una soluzione ammissibile di profitto $\lfloor \mathrm{UB} \rfloor$, e calcolare l'errore relativo di ciascuna delle tre soluzioni greedy. Individuare la scelta che impedisce al greedy per efficienza di trovarla.

Soluzione

1. Le efficienze sono

$$\frac{p_1}{w_1} = \frac{8}{5} \approx 1.60, \quad \frac{p_2}{w_2} = \frac{13}{9} \approx 1.44, \quad \frac{p_3}{w_3} = \frac{5}{4} \approx 1.25,$$
$$\frac{p_4}{w_4} = \frac{12}{8} \approx 1.50, \quad \frac{p_5}{w_5} = \frac{7}{2} \approx 3.50$$

e i tre ordinamenti sono

criterio	ordinamento
profitto	$(2, 4, 1, 5, 3)$
peso	$(5, 3, 1, 4, 2)$
efficienza	$(5, 1, 4, 2, 3)$

2. GR_p, ordinamento $(2, 4, 1, 5, 3)$, partendo da $rc = 10$:

- oggetto 2 ($w_2 = 9 \leq rc = 10$): selezionato, $rc \leftarrow 1$;
- oggetto 4 ($w_4 = 8 > rc = 1$): non selezionato;
- oggetto 1 ($w_1 = 5 > rc = 1$): non selezionato;
- oggetto 5 ($w_5 = 2 > rc = 1$): non selezionato;
- oggetto 3 ($w_3 = 4 > rc = 1$): non selezionato;

Soluzione $\{2\}$, $GR_p(I) = 13$.

GR_w, ordinamento $(5, 3, 1, 4, 2)$, partendo da $rc = 10$:

- oggetto 5 ($w_5 = 2 \leq rc = 10$): selezionato, $rc \leftarrow 8$;
- oggetto 3 ($w_3 = 4 \leq rc = 8$): selezionato, $rc \leftarrow 4$;
- oggetto 1 ($w_1 = 5 > rc = 4$): non selezionato;
- oggetto 4 ($w_4 = 8 > rc = 4$): non selezionato;
- oggetto 2 ($w_2 = 9 > rc = 4$): non selezionato;

Soluzione $\{3, 5\}$, $GR_w(I) = 5 + 7 = 12$.

GR, ordinamento $(5, 1, 4, 2, 3)$, partendo da $rc = 10$:

- oggetto 5 ($w_5 = 2 \leq rc = 10$): selezionato, $rc \leftarrow 8$;
- oggetto 1 ($w_1 = 5 \leq rc = 8$): selezionato, $rc \leftarrow 3$;
- oggetto 4 ($w_4 = 8 > rc = 3$): non selezionato;
- oggetto 2 ($w_2 = 9 > rc = 3$): non selezionato;
- oggetto 3 ($w_3 = 4 > rc = 3$): non selezionato;

Soluzione $\{1, 5\}$, $GR(I) = 8 + 7 = 15$.

3. Nell'ordinamento per efficienza $(5, 1, 4, 2, 3)$ entrano gli oggetti $\{5, 1\}$, con peso totale 7, e l'oggetto critico è 4, con capacità residua $rc = 3$. Quindi

$$\mathrm{UB} = 7 + 8 + 12 \cdot \frac{3}{8} = \frac{39}{2} \approx 19.50, \qquad \lfloor \mathrm{UB} \rfloor = 19.$$

La migliore delle tre soluzioni ha profitto 15, e

$$gap(I) = \frac{19 - 15}{15} \approx 26.7\%.$$

4. Poiché $GR(I) = 15 < \lfloor \mathrm{UB} \rfloor$, il bound non certifica l'ottimalità. Infatti la soluzione $\{4, 5\}$ ha peso $8 + 2 = 10 \leq 10$ e profitto $19 = \lfloor \mathrm{UB} \rfloor$, quindi è ottima, $z(I) = 19$. Gli errori relativi sono 31.6% per GR_p, 36.8% per GR_w e 21.1% per GR: nessun ordinamento trova l'ottimo. La soluzione ottima combina l'oggetto più efficiente 5 con l'oggetto 4, che è terzo nell'ordinamento per efficienza: dopo che l'oggetto 1 è stato selezionato, l'oggetto 4 non entra più, e la scelta dell'oggetto 1 non si rivede mai.

2.2 Bin packing: euristiche decrescenti

Esercizio 2.4 — Bin packing con ordinamento decrescente 1

Consideriamo contenitori di capacità $c = 10$ e un insieme $\mathscr{I} = \{1, \ldots, 7\}$ di 7 oggetti $(n = 7)$, con i seguenti pesi:

	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$	$j=7$
w_j	3	3	5	2	6	2	6

Domande

1. Applicare First Fit (FF) agli oggetti nell'ordine di input, passo per passo, e fornire i pattern di riempimento.
2. Ordinare gli oggetti per peso non crescente (pareggi per indice) e applicare First Fit Decreasing (FFD).
3. Applicare Best Fit Decreasing (BFD), e confrontarne i pattern di riempimento con quelli di FFD.
4. Calcolare il lower bound $\left\lceil \frac{\sum_j w_j}{c} \right\rceil$. Stabilire quali soluzioni sono ottime. Spiegare perché qui l'ordinamento aiuta.

Soluzione

1. (a) Oggetto 1 (peso 3): aprire il contenitore 1, $rc[1] = 7$.
   (b) Oggetto 2 (peso 3): primo contenitore in cui entra ($rc[1] = 7$): contenitore 1, $rc[1] = 4$.
   (c) Oggetto 3 (peso 5): non entra in nessun contenitore ($rc[1] = 4$): aprire il contenitore 2, $rc[2] = 5$.
   (d) Oggetto 4 (peso 2): primo contenitore in cui entra ($rc[1] = 4$, $rc[2] = 5$): contenitore 1, $rc[1] = 2$.
   (e) Oggetto 5 (peso 6): non entra in nessun contenitore ($rc[1] = 2$, $rc[2] = 5$): aprire il contenitore 3, $rc[3] = 4$.
   (f) Oggetto 6 (peso 2): primo contenitore in cui entra ($rc[1] = 2$, $rc[2] = 5$, $rc[3] = 4$): contenitore 1, $rc[1] = 0$.
   (g) Oggetto 7 (peso 6): non entra in nessun contenitore ($rc[1] = 0$, $rc[2] = 5$, $rc[3] = 4$): aprire il contenitore 4, $rc[4] = 4$.

   I pattern di riempimento sono

   $$P_1 = \{1, 2, 4, 6\}, \quad P_2 = \{3\}, \quad P_3 = \{5\}, \quad P_4 = \{7\}$$

   e $FF(I) = 4$.

2. L'ordinamento è $\sigma = (5, 7, 3, 1, 2, 4, 6)$ (pesi $6, 6, 5, 3, 3, 2, 2$).
   (a) Oggetto 5 (peso 6): aprire il contenitore 1, $rc[1] = 4$.
   (b) Oggetto 7 (peso 6): non entra in nessun contenitore ($rc[1] = 4$): aprire il contenitore 2, $rc[2] = 4$.
   (c) Oggetto 3 (peso 5): non entra in nessun contenitore ($rc[1] = 4$, $rc[2] = 4$): aprire il contenitore 3, $rc[3] = 5$.
   (d) Oggetto 1 (peso 3): primo contenitore in cui entra ($rc[1] = 4$, $rc[2] = 4$, $rc[3] = 5$): contenitore 1, $rc[1] = 1$.
   (e) Oggetto 2 (peso 3): primo contenitore in cui entra ($rc[1] = 1$, $rc[2] = 4$, $rc[3] = 5$): contenitore 2, $rc[2] = 1$.
   (f) Oggetto 4 (peso 2): primo contenitore in cui entra ($rc[1] = 1$, $rc[2] = 1$, $rc[3] = 5$): contenitore 3, $rc[3] = 3$.
   (g) Oggetto 6 (peso 2): primo contenitore in cui entra ($rc[1] = 1$, $rc[2] = 1$, $rc[3] = 3$): contenitore 3, $rc[3] = 1$.

   I pattern di riempimento sono

   $$P_1 = \{1, 5\}, \quad P_2 = \{2, 7\}, \quad P_3 = \{3, 4, 6\}$$

   e $FFD(I) = 3$.

3. L'ordinamento è $\sigma = (5, 7, 3, 1, 2, 4, 6)$ (pesi $6, 6, 5, 3, 3, 2, 2$).
   (a) Oggetto 5 (peso 6): aprire il contenitore 1, $rc[1] = 4$.

(b) Oggetto 7 (peso 6): non entra in nessun contenitore ($rc[1] = 4$): aprire il contenitore 2, $rc[2] = 4$.

(c) Oggetto 3 (peso 5): non entra in nessun contenitore ($rc[1] = 4$, $rc[2] = 4$): aprire il contenitore 3, $rc[3] = 5$.

(d) Oggetto 1 (peso 3): entra nei contenitori $1, 2, 3$; il più pieno è il contenitore 1 ($rc[1] = 4$, $rc[2] = 4$, $rc[3] = 5$): contenitore 1, $rc[1] = 1$.

(e) Oggetto 2 (peso 3): entra nei contenitori $2, 3$; il più pieno è il contenitore 2 ($rc[1] = 1$, $rc[2] = 4$, $rc[3] = 5$): contenitore 2, $rc[2] = 1$.

(f) Oggetto 4 (peso 2): entra solo nel contenitore 3 ($rc[1] = 1$, $rc[2] = 1$, $rc[3] = 5$): contenitore 3, $rc[3] = 3$.

(g) Oggetto 6 (peso 2): entra solo nel contenitore 3 ($rc[1] = 1$, $rc[2] = 1$, $rc[3] = 3$): contenitore 3, $rc[3] = 1$.

I pattern di riempimento sono

$$P_1 = \{1, 5\}, \quad P_2 = \{2, 7\}, \quad P_3 = \{3, 4, 6\}$$

e $BFD(I) = 3$.

4. Il peso totale è 27, quindi $LB = \lceil \frac{27}{10} \rceil = 3$. FFD e BFD usano 3 contenitori e sono quindi ottimi, $z(I) = 3$, mentre FF ne usa 4. Nell'ordine di input i quattro oggetti piccoli (1, 2, 4, 6, di pesi $3, 3, 2, 2$) finiscono tutti insieme nel contenitore 1, e gli oggetti grandi 3, 5, 7 ricevono un contenitore ciascuno, con vuoti 5, 4, 4 che nessun oggetto rimasto può riempire. L'ordinamento inserisce prima gli oggetti grandi, e gli oggetti piccoli ne riempiono i vuoti: vince l'ordine giusto.

Esercizio 2.5 — Bin packing con ordinamento decrescente 2

Consideriamo contenitori di capacità $c = 10$ e un insieme $\mathscr{I} = \{1, \ldots, 8\}$ di 8 oggetti ($n = 8$), con i seguenti pesi:

	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$	$j = 7$	$j = 8$
w_j	2	6	3	5	4	2	6	2

Domande

1. Applicare First Fit (FF) agli oggetti nell'ordine di input, passo per passo, e fornire i pattern di riempimento.
2. Ordinare gli oggetti per peso non crescente (pareggi per indice) e applicare First Fit Decreasing (FFD).
3. Applicare Best Fit Decreasing (BFD), e confrontarne i pattern di riempimento con quelli di FFD.

4. Calcolare il lower bound $\left\lceil \frac{\sum_j w_j}{c} \right\rceil$. Stabilire quali soluzioni sono ottime. Spiegare perché qui l'ordinamento non aiuta.

Soluzione

1. (a) Oggetto 1 (peso 2): aprire il contenitore 1, $rc[1] = 8$.
   (b) Oggetto 2 (peso 6): primo contenitore in cui entra ($rc[1] = 8$): contenitore 1, $rc[1] = 2$.
   (c) Oggetto 3 (peso 3): non entra in nessun contenitore ($rc[1] = 2$): aprire il contenitore 2, $rc[2] = 7$.
   (d) Oggetto 4 (peso 5): primo contenitore in cui entra ($rc[1] = 2$, $rc[2] = 7$): contenitore 2, $rc[2] = 2$.
   (e) Oggetto 5 (peso 4): non entra in nessun contenitore ($rc[1] = 2$, $rc[2] = 2$): aprire il contenitore 3, $rc[3] = 6$.
   (f) Oggetto 6 (peso 2): primo contenitore in cui entra ($rc[1] = 2$, $rc[2] = 2$, $rc[3] = 6$): contenitore 1, $rc[1] = 0$.
   (g) Oggetto 7 (peso 6): primo contenitore in cui entra ($rc[1] = 0$, $rc[2] = 2$, $rc[3] = 6$): contenitore 3, $rc[3] = 0$.
   (h) Oggetto 8 (peso 2): primo contenitore in cui entra ($rc[1] = 0$, $rc[2] = 2$, $rc[3] = 0$): contenitore 2, $rc[2] = 0$.

   I pattern di riempimento sono

   $$P_1 = \{1, 2, 6\}, \quad P_2 = \{3, 4, 8\}, \quad P_3 = \{5, 7\}$$

   e $FF(I) = 3$.

2. L'ordinamento è $\sigma = (2, 7, 4, 5, 3, 1, 6, 8)$ (pesi $6, 6, 5, 4, 3, 2, 2, 2$).
   (a) Oggetto 2 (peso 6): aprire il contenitore 1, $rc[1] = 4$.
   (b) Oggetto 7 (peso 6): non entra in nessun contenitore ($rc[1] = 4$): aprire il contenitore 2, $rc[2] = 4$.
   (c) Oggetto 4 (peso 5): non entra in nessun contenitore ($rc[1] = 4$, $rc[2] = 4$): aprire il contenitore 3, $rc[3] = 5$.
   (d) Oggetto 5 (peso 4): primo contenitore in cui entra ($rc[1] = 4$, $rc[2] = 4$, $rc[3] = 5$): contenitore 1, $rc[1] = 0$.
   (e) Oggetto 3 (peso 3): primo contenitore in cui entra ($rc[1] = 0$, $rc[2] = 4$, $rc[3] = 5$): contenitore 2, $rc[2] = 1$.
   (f) Oggetto 1 (peso 2): primo contenitore in cui entra ($rc[1] = 0$, $rc[2] = 1$, $rc[3] = 5$): contenitore 3, $rc[3] = 3$.
   (g) Oggetto 6 (peso 2): primo contenitore in cui entra ($rc[1] = 0$, $rc[2] = 1$, $rc[3] = 3$): contenitore 3, $rc[3] = 1$.
   (h) Oggetto 8 (peso 2): non entra in nessun contenitore ($rc[1] = 0$, $rc[2] = 1$, $rc[3] = 1$): aprire il contenitore 4, $rc[4] = 8$.

I pattern di riempimento sono

$$P_1 = \{2, 5\}, \quad P_2 = \{3, 7\}, \quad P_3 = \{1, 4, 6\}, \quad P_4 = \{8\}$$

e $FFD(I) = 4$.

3. L'ordinamento è $\sigma = (2, 7, 4, 5, 3, 1, 6, 8)$ (pesi $6, 6, 5, 4, 3, 2, 2, 2$).
   (a) Oggetto 2 (peso 6): aprire il contenitore 1, $rc[1] = 4$.
   (b) Oggetto 7 (peso 6): non entra in nessun contenitore ($rc[1] = 4$): aprire il contenitore 2, $rc[2] = 4$.
   (c) Oggetto 4 (peso 5): non entra in nessun contenitore ($rc[1] = 4$, $rc[2] = 4$): aprire il contenitore 3, $rc[3] = 5$.
   (d) Oggetto 5 (peso 4): entra nei contenitori $1, 2, 3$; il più pieno è il contenitore 1 ($rc[1] = 4$, $rc[2] = 4$, $rc[3] = 5$): contenitore 1, $rc[1] = 0$.
   (e) Oggetto 3 (peso 3): entra nei contenitori $2, 3$; il più pieno è il contenitore 2 ($rc[1] = 0$, $rc[2] = 4$, $rc[3] = 5$): contenitore 2, $rc[2] = 1$.
   (f) Oggetto 1 (peso 2): entra solo nel contenitore 3 ($rc[1] = 0$, $rc[2] = 1$, $rc[3] = 5$): contenitore 3, $rc[3] = 3$.
   (g) Oggetto 6 (peso 2): entra solo nel contenitore 3 ($rc[1] = 0$, $rc[2] = 1$, $rc[3] = 3$): contenitore 3, $rc[3] = 1$.
   (h) Oggetto 8 (peso 2): non entra in nessun contenitore ($rc[1] = 0$, $rc[2] = 1$, $rc[3] = 1$): aprire il contenitore 4, $rc[4] = 8$.

   I pattern di riempimento sono

$$P_1 = \{2, 5\}, \quad P_2 = \{3, 7\}, \quad P_3 = \{1, 4, 6\}, \quad P_4 = \{8\}$$

   e $BFD(I) = 4$.

4. Il peso totale è 30, quindi $LB = \lceil \frac{30}{10} \rceil = 3$: FF, con 3 contenitori, è ottimo, $z(I) = 3$, mentre FFD e BFD usano 4 contenitori. Nell'ordinamento per peso non crescente i due oggetti di peso 6 vengono accoppiati con gli oggetti di peso 4 e 3, il che lascia un vuoto di 1 nel contenitore 2; anche il contenitore 3 termina con un vuoto di 1, e l'ultimo oggetto di peso 2 non entra da nessuna parte. L'ordine di input produce per caso tre contenitori pieni ($2 + 6 + 2$, $3 + 5 + 2$, $4 + 6$). Una buona garanzia nel caso peggiore ($\frac{11}{9}$ per FFD) non significa una soluzione migliore su ogni istanza: qui l'ordinamento peggiora.

Esercizio 2.6 — Bin packing con ordinamento decrescente 3

Consideriamo contenitori di capacità $c = 10$ e un insieme $\mathscr{I} = \{1, \ldots, 7\}$ di 7 oggetti ($n = 7$), con i seguenti pesi:

	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$	$j=7$
w_j	4	4	5	7	2	2	4

Domande

1. Applicare First Fit (FF) agli oggetti nell'ordine di input, passo per passo, e fornire i pattern di riempimento.
2. Ordinare gli oggetti per peso non crescente (pareggi per indice) e applicare First Fit Decreasing (FFD).
3. Applicare Best Fit Decreasing (BFD), e confrontarne i pattern di riempimento con quelli di FFD.
4. Calcolare il lower bound $\left\lceil \frac{\sum_j w_j}{c} \right\rceil$ e confrontare le quattro regole FF, BF (entrambe nell'ordine di input), FFD e BFD.

Soluzione

1. (a) Oggetto 1 (peso 4): aprire il contenitore 1, $rc[1] = 6$.
   (b) Oggetto 2 (peso 4): primo contenitore in cui entra ($rc[1] = 6$): contenitore 1, $rc[1] = 2$.
   (c) Oggetto 3 (peso 5): non entra in nessun contenitore ($rc[1] = 2$): aprire il contenitore 2, $rc[2] = 5$.
   (d) Oggetto 4 (peso 7): non entra in nessun contenitore ($rc[1] = 2$, $rc[2] = 5$): aprire il contenitore 3, $rc[3] = 3$.
   (e) Oggetto 5 (peso 2): primo contenitore in cui entra ($rc[1] = 2$, $rc[2] = 5$, $rc[3] = 3$): contenitore 1, $rc[1] = 0$.
   (f) Oggetto 6 (peso 2): primo contenitore in cui entra ($rc[1] = 0$, $rc[2] = 5$, $rc[3] = 3$): contenitore 2, $rc[2] = 3$.
   (g) Oggetto 7 (peso 4): non entra in nessun contenitore ($rc[1] = 0$, $rc[2] = 3$, $rc[3] = 3$): aprire il contenitore 4, $rc[4] = 6$.

   I pattern di riempimento sono

   $$P_1 = \{1, 2, 5\}, \quad P_2 = \{3, 6\}, \quad P_3 = \{4\}, \quad P_4 = \{7\}$$

   e $FF(I) = 4$.
2. L'ordinamento è $\sigma = (4, 3, 1, 2, 7, 5, 6)$ (pesi $7, 5, 4, 4, 4, 2, 2$).
   (a) Oggetto 4 (peso 7): aprire il contenitore 1, $rc[1] = 3$.
   (b) Oggetto 3 (peso 5): non entra in nessun contenitore ($rc[1] = 3$): aprire il contenitore 2, $rc[2] = 5$.
   (c) Oggetto 1 (peso 4): primo contenitore in cui entra ($rc[1] = 3$, $rc[2] = 5$): contenitore 2, $rc[2] = 1$.

(d) Oggetto 2 (peso 4): non entra in nessun contenitore ($rc[1] = 3$, $rc[2] = 1$): aprire il contenitore 3, $rc[3] = 6$.

(e) Oggetto 7 (peso 4): primo contenitore in cui entra ($rc[1] = 3$, $rc[2] = 1$, $rc[3] = 6$): contenitore 3, $rc[3] = 2$.

(f) Oggetto 5 (peso 2): primo contenitore in cui entra ($rc[1] = 3$, $rc[2] = 1$, $rc[3] = 2$): contenitore 1, $rc[1] = 1$.

(g) Oggetto 6 (peso 2): primo contenitore in cui entra ($rc[1] = 1$, $rc[2] = 1$, $rc[3] = 2$): contenitore 3, $rc[3] = 0$.

I pattern di riempimento sono

$$P_1 = \{4, 5\}, \quad P_2 = \{1, 3\}, \quad P_3 = \{2, 6, 7\}$$

e $FFD(I) = 3$.

3. L'ordinamento è $\sigma = (4, 3, 1, 2, 7, 5, 6)$ (pesi $7, 5, 4, 4, 4, 2, 2$).

(a) Oggetto 4 (peso 7): aprire il contenitore 1, $rc[1] = 3$.

(b) Oggetto 3 (peso 5): non entra in nessun contenitore ($rc[1] = 3$): aprire il contenitore 2, $rc[2] = 5$.

(c) Oggetto 1 (peso 4): entra solo nel contenitore 2 ($rc[1] = 3$, $rc[2] = 5$): contenitore 2, $rc[2] = 1$.

(d) Oggetto 2 (peso 4): non entra in nessun contenitore ($rc[1] = 3$, $rc[2] = 1$): aprire il contenitore 3, $rc[3] = 6$.

(e) Oggetto 7 (peso 4): entra solo nel contenitore 3 ($rc[1] = 3$, $rc[2] = 1$, $rc[3] = 6$): contenitore 3, $rc[3] = 2$.

(f) Oggetto 5 (peso 2): entra nei contenitori $1, 3$; il più pieno è il contenitore 3 ($rc[1] = 3$, $rc[2] = 1$, $rc[3] = 2$): contenitore 3, $rc[3] = 0$.

(g) Oggetto 6 (peso 2): entra solo nel contenitore 1 ($rc[1] = 3$, $rc[2] = 1$, $rc[3] = 0$): contenitore 1, $rc[1] = 1$.

I pattern di riempimento sono

$$P_1 = \{4, 6\}, \quad P_2 = \{1, 3\}, \quad P_3 = \{2, 5, 7\}$$

e $BFD(I) = 3$.

4. Il peso totale è 28, quindi $LB = \lceil \frac{28}{10} \rceil = 3$, e FFD e BFD sono entrambi ottimi, $z(I) = 3$, con pattern di riempimento diversi: l'oggetto 5, di peso 2, esaminato subito dopo l'oggetto 7, va nel contenitore 1 in FFD (il primo contenitore in cui entra) e nel contenitore 3 in BFD (il più pieno), e le due regole terminano con soluzioni diverse, ugualmente buone. Nell'ordine di input, FF richiede 4 contenitori mentre BF (Best Fit senza ordinamento) ne trova 3, con pattern $P_1 = \{1, 2, 5\}$, $P_2 = \{3, 7\}$, $P_3 = \{4, 6\}$. Su questa istanza la scelta del contenitore conta per le regole senza ordinamento, mentre dopo l'ordinamento entrambe le scelte danno il numero ottimo di contenitori.

2.3 $P\|C_{\max}$: LIST e LPT

Esercizio 2.7 — List scheduling e LPT 1

Consideriamo $m = 3$ macchine identiche e un insieme $\mathscr{J} = \{1, \ldots, 8\}$ di 8 lavori ($n = 8$), con i seguenti tempi di esecuzione:

	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$	$j=7$	$j=8$
t_j	2	8	7	6	8	3	4	9

Domande

1. Applicare il list scheduling ($LIST$) con i lavori nell'ordine di input, mostrando i carichi prima di ogni assegnamento.
2. Applicare LPT: ordinare i lavori per tempo di esecuzione non crescente (pareggi per indice) e applicare il list scheduling.
3. Calcolare il lower bound $\max\left\{\max_j t_j, \left\lceil \frac{\sum_j t_j}{m} \right\rceil\right\}$ e i gap delle due soluzioni.
4. Stabilire se la soluzione di LPT è ottima. Spiegare perché l'ordine di input dà un risultato peggiore.

Soluzione

1.

lavoro j	t_j	carichi prima	macchina
1	2	$(0,0,0)$	1
2	8	$(2,0,0)$	2
3	7	$(2,8,0)$	3
4	6	$(2,8,7)$	1
5	8	$(8,8,7)$	3
6	3	$(8,8,15)$	1
7	4	$(11,8,15)$	2
8	9	$(11,12,15)$	1

Gli insiemi sono $J_1 = \{1,4,6,8\}$, $J_2 = \{2,7\}$, $J_3 = \{3,5\}$, con carichi $(20,12,15)$. Quindi $LIST(I) = 20$.

2. La lista è $\sigma = (8,2,5,3,4,7,6,1)$.

lavoro j	t_j	carichi prima	macchina
8	9	$(0,0,0)$	1
2	8	$(9,0,0)$	2
5	8	$(9,8,0)$	3
3	7	$(9,8,8)$	2
4	6	$(9,15,8)$	3
7	4	$(9,15,14)$	1
6	3	$(13,15,14)$	1
1	2	$(16,15,14)$	3

Gli insiemi sono $J_1 = \{6,7,8\}$, $J_2 = \{2,3\}$, $J_3 = \{1,4,5\}$, con carichi $(16,15,16)$. Quindi $LPT(I) = 16$.

3. $\sum_j t_j = 47$ e $\max_j t_j = 9$, quindi

$$LB = \max\left\{9, \left\lceil \frac{47}{3} \right\rceil\right\} = 16,$$

e i gap sono $\frac{20-16}{16} = 25\%$ per $LIST$ e $\frac{16-16}{16} = 0\%$ per LPT.

4. $LPT(I) = 16 = LB$, quindi la soluzione di LPT è ottima e $z(I) = 16$; $LIST$ nell'ordine di input è il 25.0% sopra l'ottimo. Nell'ordine di input il lavoro più lungo, il lavoro 8 di durata 9, è l'ultimo: arriva quando la macchina meno carica ha già carico 11, e il makespan salta a 20. Con prima i lavori più lunghi, gli ultimi lavori sono brevi e livellano i carichi.

Esercizio 2.8 — List scheduling e LPT 2

Consideriamo $m = 3$ macchine identiche e un insieme $\mathcal{J} = \{1, \dots, 7\}$ di 7 lavori ($n = 7$), con i seguenti tempi di esecuzione:

	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$	$j=7$
t_j	3	6	5	4	4	9	7

Domande

1. Applicare il list scheduling ($LIST$) con i lavori nell'ordine di input, mostrando i carichi prima di ogni assegnamento.
2. Applicare LPT: ordinare i lavori per tempo di esecuzione non crescente (pareggi per indice) e applicare il list scheduling.
3. Calcolare il lower bound $\max\left\{\max_j t_j, \left\lceil \frac{\sum_j t_j}{m} \right\rceil\right\}$ e i gap delle due soluzioni.
4. Trovare un assegnamento con makespan uguale al lower bound. Verificare i bound di Graham per $LIST$ e LPT su questa istanza.

Soluzione

1.

lavoro j	t_j	carichi prima	macchina
1	3	$(0,0,0)$	1
2	6	$(3,0,0)$	2
3	5	$(3,6,0)$	3
4	4	$(3,6,5)$	1
5	4	$(7,6,5)$	3
6	9	$(7,6,9)$	2
7	7	$(7,15,9)$	1

Gli insiemi sono $J_1 = \{1,4,7\}$, $J_2 = \{2,6\}$, $J_3 = \{3,5\}$, con carichi $(14,15,9)$. Quindi $LIST(I) = 15$.

2. La lista è $\sigma = (6,7,2,3,4,5,1)$.

lavoro j	t_j	carichi prima	macchina
6	9	$(0,0,0)$	1
7	7	$(9,0,0)$	2
2	6	$(9,7,0)$	3
3	5	$(9,7,6)$	3
4	4	$(9,7,11)$	2
5	4	$(9,11,11)$	1
1	3	$(13,11,11)$	2

Gli insiemi sono $J_1 = \{5,6\}$, $J_2 = \{1,4,7\}$, $J_3 = \{2,3\}$, con carichi $(13,14,11)$. Quindi $LPT(I) = 14$.

3. $\sum_j t_j = 38$ e $\max_j t_j = 9$, quindi

$$LB = \max\left\{9, \left\lceil \frac{38}{3} \right\rceil\right\} = 13,$$

e i gap sono $\frac{15-13}{13} \approx 15.4\%$ per $LIST$ e $\frac{14-13}{13} \approx 7.7\%$ per LPT.

4. L'assegnamento $J_1 = \{1,2,4\}$, $J_2 = \{3,7\}$, $J_3 = \{5,6\}$ ha carichi $(13,12,13)$ e makespan $13 = LB$: è ottimo, $z(I) = 13$, e LPT non lo è. Il bound di Graham dà $LPT(I) \leq \left(\frac{4}{3} - \frac{1}{9}\right) z(I) = \frac{11}{9} \cdot 13 \approx 15.89$, e infatti $14 \leq 15.89$; per $LIST$, $\left(2 - \frac{1}{3}\right) z(I) = \frac{5}{3} \cdot 13 \approx 21.67 \geq 15$. L'ordinamento migliora il makespan da 15 a 14, ma non raggiunge l'ottimo.

Esercizio 2.9 — List scheduling e LPT 3

Consideriamo $m = 3$ macchine identiche e un insieme $\mathcal{J} = \{1, \ldots, 7\}$ di 7 lavori ($n = 7$), con i seguenti tempi di esecuzione:

	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$	$j=7$
t_j	5	8	8	4	5	7	4

Domande

1. Applicare il list scheduling ($LIST$) con i lavori nell'ordine di input, mostrando i carichi prima di ogni assegnamento.
2. Applicare LPT: ordinare i lavori per tempo di esecuzione non crescente (pareggi per indice) e applicare il list scheduling.
3. Calcolare il lower bound $\max\left\{\max_j t_j, \left\lceil \frac{\sum_j t_j}{m} \right\rceil\right\}$ e i gap delle due soluzioni.
4. Confrontare le due soluzioni. Sapendo che il makespan ottimo è 15, stabilire quale delle due è ottima e che cosa se ne deduce sull'ordinamento.

Soluzione

1.

lavoro j	t_j	carichi prima	macchina
1	5	$(0,0,0)$	1
2	8	$(5,0,0)$	2
3	8	$(5,8,0)$	3
4	4	$(5,8,8)$	1
5	5	$(9,8,8)$	2
6	7	$(9,13,8)$	3
7	4	$(9,13,15)$	1

Gli insiemi sono $J_1=\{1,4,7\}$, $J_2=\{2,5\}$, $J_3=\{3,6\}$, con carichi $(13,13,15)$. Quindi $LIST(I)=15$.

2. La lista è $\sigma=(2,3,6,1,5,4,7)$.

lavoro j	t_j	carichi prima	macchina
2	8	$(0,0,0)$	1
3	8	$(8,0,0)$	2
6	7	$(8,8,0)$	3
1	5	$(8,8,7)$	3
5	5	$(8,8,12)$	1
4	4	$(13,8,12)$	2
7	4	$(13,12,12)$	2

Gli insiemi sono $J_1=\{2,5\}$, $J_2=\{3,4,7\}$, $J_3=\{1,6\}$, con carichi $(13,16,12)$. Quindi $LPT(I)=16$.

3. $\sum_j t_j = 41$ e $\max_j t_j = 8$, quindi

$$LB = \max\left\{8, \left\lceil \frac{41}{3} \right\rceil\right\} = 14,$$

e i gap sono $\frac{15-14}{14} \approx 7.1\%$ per $LIST$ e $\frac{16-14}{14} \approx 14.3\%$ per LPT.

4. Qui $LIST(I) = 15 < LPT(I) = 16$: su questa istanza l'ordine di input è migliore di quello ordinato. Il lower bound 14 non è raggiunto; un'enumerazione completa dei 3^7 assegnamenti mostra che $z(I) = 15$ (per esempio $J_1 = \{1, 2\}$, $J_2 = \{3, 5\}$, $J_3 = \{4, 6, 7\}$, carichi $(13, 13, 15)$), quindi $LIST$ nell'ordine di input è ottimo, mentre LPT è il 6.7% sopra l'ottimo. L'ordinamento è una regola che funziona bene in generale, e il bound di Graham per LPT è migliore di quello per $LIST$; ma una garanzia è un'affermazione sul caso peggiore, e su una singola istanza un ordine arbitrario può essere fortunato.

2.4 TSP: euristiche costruttive

Esercizio 2.10 — Costruzione di giri del TSP 1

Consideriamo un insieme $\mathcal{V} = \{1, \ldots, 6\}$ di 6 città con coordinate 1: $(8, 0)$, 2: $(1, 0)$, 3: $(1, 2)$, 4: $(1, 5)$, 5: $(6, 3)$, 6: $(0, 4)$ e distanze uguali alle distanze euclidee arrotondate all'intero più vicino (l'istanza è metrica):

d_{ij}	1	2	3	4	5	6
1	—	7	7	9	4	9
2	7	—	2	5	6	4
3	7	2	—	3	5	2
4	9	5	3	—	5	1
5	4	6	5	5	—	6
6	9	4	2	1	6	—

Domande

1. Applicare l'euristica nearest neighbour (NN) dalla città 1, passo per passo (pareggi per indice).
2. Applicare NN da ciascuna delle altre città, e fornire il risultato del NN multi-start.
3. Calcolare il lower bound $\left\lceil \frac{1}{2} \sum_i (a_i + b_i) \right\rceil$, dove a_i e b_i sono le due distanze minime dalla città i, e il gap della soluzione multi-start.
4. Sapendo che il valore ottimo è $z(I) = 21$, calcolare l'errore relativo di NN dalla città 1, e spiegare perché il suo giro è lungo.

Soluzione

1. (a) dalla città 1 le distanze verso $2, 3, 4, 5, 6$ sono $7, 7, 9, 4, 9$: città 5;
   (b) dalla città 5 le distanze verso $2, 3, 4, 6$ sono $6, 5, 5, 6$: città 3 (pareggio con 4, risolto per indice);
   (c) dalla città 3 le distanze verso $2, 4, 6$ sono $2, 3, 2$: città 2 (pareggio con 6, risolto per indice);
   (d) dalla città 2 le distanze verso $4, 6$ sono $5, 4$: città 6;
   (e) dalla città 6 resta solo la città 4, a distanza 1;
   (f) il giro si chiude con l'arco $(4, 1)$, di lunghezza 9;

   Quindi $\pi = (1, 5, 3, 2, 6, 4)$ e $NN(I) = 4 + 5 + 2 + 4 + 1 + 9 = 25$.

2. Procedendo allo stesso modo da ogni città:

partenza	giro	L
1	$(1, 5, 3, 2, 6, 4)$	25
2	$(2, 3, 6, 4, 5, 1)$	21
3	$(3, 2, 6, 4, 5, 1)$	23
4	$(4, 6, 3, 2, 5, 1)$	24
5	$(5, 1, 2, 3, 6, 4)$	21
6	$(6, 4, 3, 2, 5, 1)$	25

   Il NN multi-start restituisce il primo giro più corto, $(2, 3, 6, 4, 5, 1)$, di lunghezza 21.

3. Le due distanze minime sono $(4, 7)$ per la città 1, $(2, 4)$ per la città 2, $(2, 2)$ per la città 3, $(1, 3)$ per la città 4, $(4, 5)$ per la città 5, $(1, 2)$ per la città 6, quindi

$$LB = \left\lceil \frac{11 + 6 + 4 + 4 + 9 + 3}{2} \right\rceil = \lceil 18.5 \rceil = 19,$$

   e il gap della soluzione multi-start è $\frac{21-19}{19} \approx 10.5\%$.

4. L'errore relativo di NN dalla città 1 è $\frac{25-21}{21} \approx 19.0\%$, mentre il multi-start trova il valore ottimo 21 (dalle città 2 e 5). Dalla città 1 la città più vicina è la 5, e il cammino si sposta poi sul gruppo di città $2, 3, 4, 6$; la città 1 è lontana da tutte (le sue distanze verso $2, 3, 4, 6$ sono $7, 7, 9, 9$), e il cammino deve tornare dall'ultima città del gruppo: l'arco di chiusura $(4, 1)$ ha lunghezza 9. Cambiare la città di partenza cambia l'arco lungo forzato alla fine.

Esercizio 2.11 — Costruzione di giri del TSP 2

Consideriamo un insieme $\mathscr{V} = \{1, \ldots, 6\}$ di 6 città con coordinate 1: $(4, 3)$, 2: $(6, 1)$, 3: $(0, 5)$, 4: $(0, 1)$, 5: $(4, 5)$, 6: $(4, 0)$ e distanze uguali alle distanze euclidee arrotondate all'intero più vicino (l'istanza è metrica):

d_{ij}	1	2	3	4	5	6
1	—	3	4	4	2	3
2	3	—	7	6	4	2
3	4	7	—	4	4	6
4	4	6	4	—	6	4
5	2	4	4	6	—	5
6	3	2	6	4	5	—

Domande

1. Applicare il nearest insertion (NI) dalla città 1: a ogni passo fornire $d(q, T)$ per le città q non nel sottogiro T, la città selezionata e il suo arco più economico.
2. Applicare il farthest insertion (FI) dalla città 1.
3. Applicare il cheapest insertion (CI) dalla città 1.
4. Sapendo che il valore ottimo è $z(I) = 19$, confrontare i tre giri fra loro e con il giro di NN dalla città 1.

Soluzione

1. Partendo da $T = (1)$ (inserire la città q nel sottogiro (1) costa $2d_{1q}$):

sottogiro T	$d(q, T)$	costi di inserimento della città selezionata	inserita
(1)	2: 3, 3: 4, 4: 4, 5: 2, 6: 3	(4)	5 su $(1, 1)$, costo 4
$(1, 5)$	2: 3, 3: 4, 4: 4, 6: 3	$(5, 5)$	2 su $(1, 5)$, costo 5
$(1, 2, 5)$	3: 4, 4: 4, 6: 2	$(2, 3, 6)$	6 su $(1, 2)$, costo 2
$(1, 6, 2, 5)$	3: 4, 4: 4	$(7, 11, 7, 6)$	3 su $(5, 1)$, costo 6
$(1, 6, 2, 5, 3)$	4: 4	$(5, 8, 8, 6, 4)$	4 su $(3, 1)$, costo 4

   Quindi $\pi = (1, 6, 2, 5, 3, 4)$ e $NI(I) = 3 + 2 + 4 + 4 + 4 + 4 = 21$.

2. Partendo da $T = (1)$ (inserire la città q nel sottogiro (1) costa $2d_{1q}$):

sottogiro T	$d(q, T)$	costi di inserimento della città selezionata	inserita
(1)	2: 3, 3: 4, 4: 4, 5: 2, 6: 3	(8)	3 su $(1, 1)$, costo 8
$(1, 3)$	2: 3, 4: 4, 5: 2, 6: 3	$(4, 4)$	4 su $(1, 3)$, costo 4
$(1, 4, 3)$	2: 3, 5: 2, 6: 3	$(5, 9, 6)$	2 su $(1, 4)$, costo 5
$(1, 2, 4, 3)$	5: 2, 6: 2	$(3, 4, 6, 2)$	5 su $(3, 1)$, costo 2
$(1, 2, 4, 3, 5)$	6: 2	$(2, 0, 6, 7, 6)$	6 su $(2, 4)$, costo 0

Quindi $\pi = (1, 2, 6, 4, 3, 5)$ e $FI(I) = 3 + 2 + 4 + 4 + 4 + 2 = 19$.

3. Partendo da $T = (1)$ (inserire la città q nel sottogiro (1) costa $2d_{1q}$):

sottogiro T	costo di inserimento minimo	costi di inserimento della città selezionata	inserita
(1)	2: 6, 3: 8, 4: 8, 5: 4, 6: 6	(4)	5 su (1, 1), costo 4
(1, 5)	2: 5, 3: 6, 4: 8, 6: 6	(5, 5)	2 su (1, 5), costo 5
(1, 2, 5)	3: 6, 4: 7, 6: 2	(2, 3, 6)	6 su (1, 2), costo 2
(1, 6, 2, 5)	3: 6, 4: 5	(5, 8, 8, 8)	4 su (1, 6), costo 5
(1, 4, 6, 2, 5)	3: 4	(4, 6, 11, 7, 6)	3 su (1, 4), costo 4

Quindi $\pi = (1, 3, 4, 6, 2, 5)$ e $CI(I) = 4 + 4 + 4 + 2 + 4 + 2 = 20$.

4. Le lunghezze sono $NI(I) = 21$, $FI(I) = 19$, $CI(I) = 20$ e $NN(I) = 20$ dalla città 1. FI trova un giro ottimo, $z(I) = 19$; gli errori relativi sono 10.5% per NI, 5.3% per CI e 5.3% per NN. FI costruisce prima il contorno del giro attraverso le città 3 e 4, le più lontane dalla città 1, e poi inserisce le città 2, 5 e 6 dove costano poco (la città 6 a costo 0); NI e CI partono dalle città 5, 2 e 6, vicine alla città 1, e devono inserire le città lontane 3 e 4 alla fine, quando il sottogiro non ha più una posizione economica per loro.

Esercizio 2.12 — Costruzione di giri del TSP 3

Consideriamo un insieme $\mathscr{V} = \{1, \ldots, 5\}$ di 5 città con coordinate 1: (8, 1), 2: (5, 0), 3: (6, 4), 4: (6, 3), 5: (3, 5) e distanze uguali alle distanze euclidee arrotondate all'intero più vicino (l'istanza è metrica):

d_{ij}	1	2	3	4	5
1	—	3	4	3	6
2	3	—	4	3	5
3	4	4	—	1	3
4	3	3	1	—	4
5	6	5	3	4	—

Domande

1. Applicare l'euristica nearest neighbour (NN) dalla città 1.
2. Applicare il nearest neighbour randomizzato RNN_α dalla città 1 con $\alpha = \frac{1}{2}$, usando i numeri casuali $u_1 = 0.0159$, $u_2 = 0.4956$, $u_3 = 0.8564$ in quest'ordine. La RCL elenca le sue città per indice crescente, si sceglie la città in posizione $\lfloor u \cdot |RCL| \rfloor + 1$, e si usa un numero casuale solo quando $|RCL| \geq 2$. Fornire il giro della prima esecuzione.

3. Proseguire con i numeri casuali rimanenti per una seconda esecuzione di RNN_α dalla città 1.

4. Sapendo che il valore ottimo è $z(I) = 15$, confrontare i tre giri. Descrivere la RCL con $\alpha = 0$ e con $\alpha = 1$.

Soluzione

1. (a) dalla città 1 le distanze verso $2, 3, 4, 5$ sono $3, 4, 3, 6$: città 2 (pareggio con 4, risolto per indice);
   (b) dalla città 2 le distanze verso $3, 4, 5$ sono $4, 3, 5$: città 4;
   (c) dalla città 4 le distanze verso $3, 5$ sono $1, 4$: città 3;
   (d) dalla città 3 resta solo la città 5, a distanza 3;
   (e) il giro si chiude con l'arco $(5, 1)$, di lunghezza 6;

   Quindi $\pi = (1, 2, 4, 3, 5)$ e $NN(I) = 3 + 3 + 1 + 3 + 6 = 16$.

2. (a) dalla città 1 le distanze verso $2, 3, 4, 5$ sono $3, 4, 3, 6$; soglia $3 + \frac{1}{2} \cdot (6 - 3) = 4.5$, $RCL = \{2, 3, 4\}$; $u_1 = 0.0159$, posizione $\lfloor 0.0159 \cdot 3 \rfloor + 1 = 1$: città 2;
   (b) dalla città 2 le distanze verso $3, 4, 5$ sono $4, 3, 5$; soglia $3 + \frac{1}{2} \cdot (5 - 3) = 4$, $RCL = \{3, 4\}$; $u_2 = 0.4956$, posizione $\lfloor 0.4956 \cdot 2 \rfloor + 1 = 1$: città 3;
   (c) dalla città 3 le distanze verso $4, 5$ sono $1, 3$; soglia $1 + \frac{1}{2} \cdot (3 - 1) = 2$, $RCL = \{4\}$: città 4;
   (d) dalla città 4 resta solo la città 5;

   Quindi $\pi = (1, 2, 3, 4, 5)$ e $L = 3 + 4 + 1 + 4 + 6 = 18$.

3. (a) dalla città 1 le distanze verso $2, 3, 4, 5$ sono $3, 4, 3, 6$; soglia $3 + \frac{1}{2} \cdot (6 - 3) = 4.5$, $RCL = \{2, 3, 4\}$; $u_3 = 0.8564$, posizione $\lfloor 0.8564 \cdot 3 \rfloor + 1 = 3$: città 4;
   (b) dalla città 4 le distanze verso $2, 3, 5$ sono $3, 1, 4$; soglia $1 + \frac{1}{2} \cdot (4 - 1) = 2.5$, $RCL = \{3\}$: città 3;
   (c) dalla città 3 le distanze verso $2, 5$ sono $4, 3$; soglia $3 + \frac{1}{2} \cdot (4 - 3) = 3.5$, $RCL = \{5\}$: città 5;
   (d) dalla città 5 resta solo la città 2;

   Quindi $\pi = (1, 4, 3, 5, 2)$ e $L = 3 + 1 + 3 + 5 + 3 = 15$.

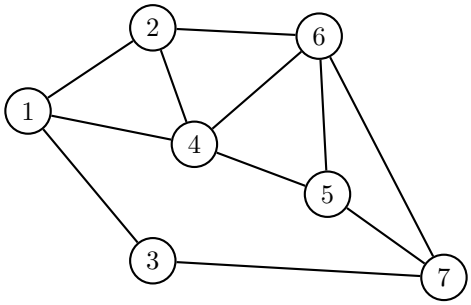
4. NN dalla città 1 dà 16, il 6.7% sopra l'ottimo; la prima esecuzione randomizzata dà 18, peggio di NN, e la seconda $15 = z(I)$, un giro ottimo. La randomizzazione può dare giri peggiori o migliori della scelta greedy, e ciò che la rende utile è ripetere la costruzione e tenere il giro migliore. Con $\alpha = 0$ la RCL contiene solo le città più vicine, e RNN_0 è NN con pareggi risolti a caso; con $\alpha = 1$ la RCL contiene tutte le città non visitate, e il giro è completamente casuale.

2.5 Colorazione: greedy e DSATUR

Esercizio 2.13 — Colorazione greedy e DSATUR 1

Consideriamo il grafo $G = (V, E)$ con $V = \{1, \dots, 7\}$ ($n = 7$) e $|E| = 11$ archi:

$$E = \big\{\{1,2\},\{1,3\},\{1,4\},\{2,4\},\{2,6\},\{3,7\},\{4,5\},\{4,6\},\{5,6\},\{5,7\},\{6,7\}\big\}.$$



Domande

1. Calcolare i gradi e $\Delta(G)$, e il bound $\Delta(G)+1$ sul numero di colori di GC.
2. Applicare GC con la lista $\lambda = (1, 2, \dots, 7)$: per ogni vertice fornire l'insieme U dei colori dei suoi vicini colorati e il colore assegnato.
3. Applicare $DSATUR$: per ogni passo fornire il vertice scelto, il suo grado di saturazione sat, il suo grado fra i vertici non colorati du, e il suo colore.
4. Trovare una clique di cardinalità massima e determinare $\chi(G)$. Stabilire quale delle due colorazioni è ottima.

Soluzione

1. Gli insiemi dei vicini sono $N(1) = \{2,3,4\}$, $N(2) = \{1,4,6\}$, $N(3) = \{1,7\}$, $N(4) = \{1,2,5,6\}$, $N(5) = \{4,6,7\}$, $N(6) = \{2,4,5,7\}$, $N(7) = \{3,5,6\}$. I gradi sono $(3,3,2,4,3,4,3)$, quindi $\Delta(G) = 4$ e GC usa al più $\Delta(G)+1 = 5$ colori, con qualunque lista.

2.

v	1	2	3	4	5	6	7
U	$\emptyset$	$\{1\}$	$\{1\}$	$\{1,2\}$	$\{3\}$	$\{1,2,3\}$	$\{1,2,4\}$
$col[v]$	1	2	2	3	1	4	3

   GC usa 4 colori: $Q_1 = \{1,5\}$, $Q_2 = \{2,3\}$, $Q_3 = \{4,7\}$, $Q_4 = \{6\}$. Il problema nasce al vertice 5: prende il colore 1, e il vertice 6, adiacente a 2, 4 e 5, vede allora i colori $1, 2, 3$.

3. Al passo 1 tutti i gradi di saturazione sono 0, e i vertici di grado massimo sono 4 e 6: si sceglie il vertice 4.

passo	1	2	3	4	5	6	7
v	4	6	2	1	5	7	3
$sat[v]$	0	1	2	2	2	2	2
$du[v]$	4	3	1	1	1	1	0
$col[v]$	1	2	3	2	3	1	3

Al passo 2 i vicini di 4 hanno grado di saturazione 1, e il vertice 6 ha più vicini non colorati (5, 7 e 2). Al passo 3 i vertici 2 e 5 (entrambi adiacenti a 4 e 6) hanno grado di saturazione 2 e un vicino non colorato ciascuno, e si sceglie 2. $DSATUR$ usa 3 colori: $Q_1 = \{4, 7\}$, $Q_2 = \{1, 6\}$, $Q_3 = \{2, 3, 5\}$.

4. $\{1, 2, 4\}$ è una clique di cardinalità 3 ($\{1, 2\}, \{1, 4\}, \{2, 4\} \in E$), quindi $\chi(G) \geq \omega(G) \geq 3$; non ci sono clique di cardinalità 4, ma non serve: la colorazione di $DSATUR$ usa 3 colori, quindi $\chi(G) = 3$. La colorazione di $DSATUR$ è ottima, quella di GC con la lista $(1, \ldots, 7)$ usa un colore in più:

$$\frac{GC(I)}{\chi(G)} = \frac{4}{3}, \qquad \frac{DSATUR(I)}{\chi(G)} = 1.$$

Esercizio 2.14 — Colorazione greedy e DSATUR 2

Consideriamo il grafo corona G_5 con $n = 10$ vertici: i vertici $1, \ldots, 5$ da un lato, i vertici $6, \ldots, 10$ dall'altro, e il vertice $i \in \{1, \ldots, 5\}$ adiacente al vertice $j + 5$, $j \in \{1, \ldots, 5\}$, se e solo se $i \neq j$:

$$E = \big\{\{i, j + 5\} : i, j \in \{1, \ldots, 5\},\ i \neq j\big\}, \qquad |E| = 20.$$

Domande

1. Calcolare $\Delta(G_5)$ e $\chi(G_5)$.
2. Applicare GC con la lista $\lambda = (1, 6, 2, 7, 3, 8, 4, 9, 5, 10)$ e fornire $GC(I)$. Confrontarlo con $\Delta(G_5) + 1$ e con $\chi(G_5)$.
3. Fornire una lista per cui GC è ottimo.
4. Applicare $DSATUR$ e fornire $DSATUR(I)$.

Soluzione

1. Ogni vertice $i \leq 5$ è adiacente ai 4 vertici $j + 5$ con $j \neq i$, e simmetricamente: ogni grado è 4 e $\Delta(G_5) = 4$. Il grafo è bipartito (tutti gli archi collegano $\{1, \ldots, 5\}$ e $\{6, \ldots, 10\}$) e ha archi, quindi $\chi(G_5) = 2$.
2. Con $u_i = i$ e $v_i = i + 5$ la lista è $(u_1, v_1, u_2, v_2, \ldots, u_5, v_5)$, la lista cattiva delle note:

v	1	6	2	7	3	8	4	9	5	10
U	$\emptyset$	$\emptyset$	$\{1\}$	$\{1\}$	$\{1,2\}$	$\{1,2\}$	$\{1,2,3\}$	$\{1,2,3\}$	$\{1,2,3,4\}$	$\{1,2,3,4\}$
$col[v]$	1	1	2	2	3	3	4	4	5	5

I vertici i e $i+5$ non sono adiacenti e ricevono entrambi il colore i, quindi $GC(I) = 5 = \frac{n}{2}$. È esattamente il bound $\Delta(G_5)+1 = 5$, e

$$\frac{GC(I)}{\chi(G_5)} = \frac{5}{2}.$$

3. La lista $(1,2,3,4,5,6,7,8,9,10)$, prima un lato: i vertici $1,\dots,5$ sono a due a due non adiacenti e ricevono tutti il colore 1; ogni vertice $6,\dots,10$ ha solo vicini di colore 1 e riceve il colore 2. Quindi $GC(I) = 2 = \chi(G_5)$.

4. Al passo 1 tutti i vertici hanno grado 4 e si sceglie il vertice 1 (colore 1). Poi:

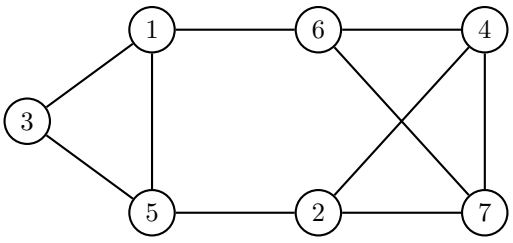
passo	1	2	3	4	5	6	7	8	9	10
v	1	7	3	4	5	6	2	8	9	10
$sat[v]$	0	1	1	1	1	1	1	1	1	1
$du[v]$	4	3	3	3	3	1	3	0	0	0
$col[v]$	1	2	1	1	1	2	1	2	2	2

Al passo 2 i vicini $7,8,9,10$ del vertice 1 hanno grado di saturazione 1 e 3 vicini non colorati ciascuno: si sceglie il vertice 7, che riceve il colore 2. Da quel momento, ogni vertice adiacente a un vertice colorato ha grado di saturazione 1 (i suoi vicini colorati stanno tutti dall'altro lato, con lo stesso colore); i vertici del lato $\{1,\dots,5\}$ ricevono il colore 1 e quelli del lato $\{6,\dots,10\}$ il colore 2. Quindi $DSATUR(I) = 2 = \chi(G_5)$, come garantito sui grafi bipartiti.

Esercizio 2.15 — Colorazione greedy e DSATUR 3

Consideriamo il grafo $G = (V, E)$ con $V = \{1,\dots,7\}$ ($n = 7$) e $|E| = 10$ archi:

$$E = \big\{\{1,3\},\{1,5\},\{1,6\},\{2,4\},\{2,5\},\{2,7\},\{3,5\},\{4,6\},\{4,7\},\{6,7\}\big\}.$$



Domande

1. Applicare $DSATUR$ (pareggi: grado massimo fra i vertici non colorati, poi indice minimo) e fornire $DSATUR(I)$.

2. Mostrare che $\chi(G) = 3$.
3. Stabilire se il risultato della Domanda 1 è in contraddizione con il bound $\Delta(G) + 1$.

Soluzione

1. Gli insiemi dei vicini sono $N(1) = \{3, 5, 6\}$, $N(2) = \{4, 5, 7\}$, $N(3) = \{1, 5\}$, $N(4) = \{2, 6, 7\}$, $N(5) = \{1, 2, 3\}$, $N(6) = \{1, 4, 7\}$, $N(7) = \{2, 4, 6\}$.

passo	1	2	3	4	5	6	7
v	1	5	3	2	4	6	7
$sat[v]$	0	1	2	1	1	2	3
$du[v]$	3	2	0	2	2	1	0
$col[v]$	1	2	3	1	2	3	4

   - Passo 1: vertice 1 (grado 3, indice minimo fra i vertici di grado 3), colore 1.
   - Passo 2: i vicini $3, 5, 6$ del vertice 1 hanno grado di saturazione 1; i loro vicini non colorati sono rispettivamente 1, 2, 2, e si sceglie il vertice 5: colore 2.
   - Passo 3: il vertice 3 vede i colori 1 e 2 (grado di saturazione 2): colore 3.
   - Passo 4: i vertici 2 (vicino 5) e 6 (vicino 1) hanno grado di saturazione 1 e due vicini non colorati ciascuno; si sceglie il vertice 2, che riceve il colore 1.
   - Passo 5: i vertici 4, 6, 7 hanno grado di saturazione 1 e due vicini non colorati ciascuno; il vertice 4 riceve il colore 2.
   - Passo 6: il vertice 6 vede i colori 1 (vertice 1) e 2 (vertice 4), il vertice 7 vede 1 (vertice 2) e 2 (vertice 4); entrambi hanno grado di saturazione 2 e un vicino non colorato, e il vertice 6 riceve il colore 3.
   - Passo 7: il vertice 7 vede i colori $1, 2, 3$ e riceve il colore 4.

   Quindi $DSATUR(I) = 4$.

2. $\{1, 3, 5\}$ è un triangolo, quindi $\chi(G) \geq 3$. La colorazione $Q_1 = \{1, 4\}$, $Q_2 = \{2, 3, 6\}$, $Q_3 = \{5, 7\}$ è ammissibile (nessun arco dentro una classe: $\{1, 4\}$, $\{2, 3\}$, $\{2, 6\}$, $\{3, 6\}$, $\{5, 7\} \notin E$), quindi $\chi(G) = 3$ e

$$\frac{DSATUR(I)}{\chi(G)} = \frac{4}{3}.$$

   Il danno lo fa il passo 5: i vertici 6 e 7 vedono già il colore 1 (dai vertici 1 e 2), e il vertice 4, loro vicino comune, prende il colore 2. Ai due vertici adiacenti 6 e 7 resta solo il colore 3, e uno dei due richiede un quarto colore. Nella colorazione ottima il vertice 4 ha lo stesso colore del vertice 1.

3. No: i gradi sono $(3, 3, 2, 3, 3, 3, 3)$, quindi $\Delta(G) + 1 = 4$ e $DSATUR(I) = 4$ raggiunge il bound. Il bound $\Delta(G) + 1$ limita il numero di colori, non lo confronta con $\chi(G)$.

Capitolo 3

Ricerca locale

3.1 Zaino

Esercizio 3.1 — Ricerca locale KP 1

Consideriamo uno zaino di capacità $c = 18$ e un insieme $\mathscr{I} = \{1, 2, 3, 4, 5\}$ di 5 oggetti ($n = 5$), con i seguenti profitti e pesi:

	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$
p_j	10	7	13	14	8
w_j	2	3	8	9	7

Gli oggetti sono ordinati per efficienza non crescente. L'intorno flip+swap contiene i vettori ammissibili ottenuti con un flip (aggiungere o togliere un oggetto) o con uno swap (togliere un oggetto selezionato j, aggiungere un oggetto non selezionato k); si esaminano prima i flip, per $j = 1, \ldots, n$, poi gli swap, per j crescente e, per ogni j, per k crescente. In best improvement i pareggi si risolvono a favore della mossa esaminata per prima.

Domande

1. Calcolare la soluzione dell'algoritmo greedy GR (per efficienza) e il suo profitto.
2. Applicare la ricerca locale con l'intorno flip+swap e la regola best improvement, partendo dalla soluzione di GR. A ogni iterazione elencare i vicini, con il loro peso e il loro profitto, e indicare la mossa accettata.
3. Calcolare la soluzione del greedy per profitto GR_p, e applicare la stessa ricerca locale partendo da essa.
4. Il valore ottimo è $z(I) = 32$. Quale delle due ricerche locali trova una soluzione ottima? Spiegare perché l'altra si ferma, esaminando le mosse necessarie per raggiungere la soluzione ottima.

Soluzione

1. Le efficienze sono $\frac{10}{2}$, $\frac{7}{3}$, $\frac{13}{8}$, $\frac{14}{9}$, $\frac{8}{7}$, non crescenti. Con $rc = 18$:
   - l'oggetto 1 ($w_1 = 2$) è selezionato, $rc = 16$;
   - l'oggetto 2 ($w_2 = 3$) è selezionato, $rc = 13$;
   - l'oggetto 3 ($w_3 = 8$) è selezionato, $rc = 5$;
   - l'oggetto 4 ($w_4 = 9 > rc = 5$) non è selezionato;
   - l'oggetto 5 ($w_5 = 7 > rc = 5$) non è selezionato;

   GR restituisce $x = (1, 1, 1, 0, 0)$, gli oggetti $\{1, 2, 3\}$, di peso 13 e profitto $GR(I) = 30$.

2. Iterazione 1: soluzione corrente $x = (1, 1, 1, 0, 0)$, peso 13, profitto 30.

mossa	vicino	peso	profitto	
flip 1	$(0, 1, 1, 0, 0)$	11	20	
flip 2	$(1, 0, 1, 0, 0)$	10	23	
flip 3	$(1, 1, 0, 0, 0)$	5	17	
flip 4	$(1, 1, 1, 1, 0)$	22	—	non ammissibile
flip 5	$(1, 1, 1, 0, 1)$	20	—	non ammissibile
1 fuori, 4 dentro	$(0, 1, 1, 1, 0)$	20	—	non ammissibile
1 fuori, 5 dentro	$(0, 1, 1, 0, 1)$	18	28	
2 fuori, 4 dentro	$(1, 0, 1, 1, 0)$	19	—	non ammissibile
2 fuori, 5 dentro	$(1, 0, 1, 0, 1)$	17	31	← accettata
3 fuori, 4 dentro	$(1, 1, 0, 1, 0)$	14	31	
3 fuori, 5 dentro	$(1, 1, 0, 0, 1)$	12	25	

   La mossa è accettata dopo aver esaminato 7 vicini ammissibili.

   Iterazione 2: soluzione corrente $x = (1, 0, 1, 0, 1)$, peso 17, profitto 31.

mossa	vicino	peso	profitto	
flip 1	$(0, 0, 1, 0, 1)$	15	21	
flip 2	$(1, 1, 1, 0, 1)$	20	—	non ammissibile
flip 3	$(1, 0, 0, 0, 1)$	9	18	
flip 4	$(1, 0, 1, 1, 1)$	26	—	non ammissibile
flip 5	$(1, 0, 1, 0, 0)$	10	23	
1 fuori, 2 dentro	$(0, 1, 1, 0, 1)$	18	28	
1 fuori, 4 dentro	$(0, 0, 1, 1, 1)$	24	—	non ammissibile
3 fuori, 2 dentro	$(1, 1, 0, 0, 1)$	12	25	
3 fuori, 4 dentro	$(1, 0, 0, 1, 1)$	18	32	← accettata
5 fuori, 2 dentro	$(1, 1, 1, 0, 0)$	13	30	
5 fuori, 4 dentro	$(1, 0, 1, 1, 0)$	19	—	non ammissibile

   La mossa è accettata dopo aver esaminato 7 vicini ammissibili.

   Iterazione 3: soluzione corrente $x = (1, 0, 0, 1, 1)$, peso 18, profitto 32.

mossa	vicino	peso	profitto	
flip 1	$(0,0,0,1,1)$	16	22	
flip 2	$(1,1,0,1,1)$	21	—	non ammissibile
flip 3	$(1,0,1,1,1)$	26	—	non ammissibile
flip 4	$(1,0,0,0,1)$	9	18	
flip 5	$(1,0,0,1,0)$	11	24	
1 fuori, 2 dentro	$(0,1,0,1,1)$	19	—	non ammissibile
1 fuori, 3 dentro	$(0,0,1,1,1)$	24	—	non ammissibile
4 fuori, 2 dentro	$(1,1,0,0,1)$	12	25	
4 fuori, 3 dentro	$(1,0,1,0,1)$	17	31	
5 fuori, 2 dentro	$(1,1,0,1,0)$	14	31	
5 fuori, 3 dentro	$(1,0,1,1,0)$	19	—	non ammissibile

Nessun vicino ammissibile ha un profitto maggiore: $x = (1,0,0,1,1)$ è un **ottimo locale**, di profitto 32 (6 vicini ammissibili esaminati).

La ricerca locale esegue 2 mosse, $30 \to 31 \to 32$, e si ferma in $x = (1,0,0,1,1)$, di profitto 32.

3. Gli oggetti in ordine di profitto non crescente sono $(4,3,1,5,2)$. Con $rc = 18$:

   - l'oggetto 4 ($w_4 = 9$) è selezionato, $rc = 9$;
   - l'oggetto 3 ($w_3 = 8$) è selezionato, $rc = 1$;
   - l'oggetto 1 ($w_1 = 2 > rc = 1$) non è selezionato;
   - l'oggetto 5 ($w_5 = 7 > rc = 1$) non è selezionato;
   - l'oggetto 2 ($w_2 = 3 > rc = 1$) non è selezionato;

   L'algoritmo greedy per profitto restituisce $x = (0,0,1,1,0)$, di peso 17 e profitto 27.

   Iterazione 1: soluzione corrente $x = (0,0,1,1,0)$, peso 17, profitto 27.

mossa	vicino	peso	profitto	
flip 1	$(1,0,1,1,0)$	19	—	non ammissibile
flip 2	$(0,1,1,1,0)$	20	—	non ammissibile
flip 3	$(0,0,0,1,0)$	9	14	
flip 4	$(0,0,1,0,0)$	8	13	
flip 5	$(0,0,1,1,1)$	24	—	non ammissibile
3 fuori, 1 dentro	$(1,0,0,1,0)$	11	24	
3 fuori, 2 dentro	$(0,1,0,1,0)$	12	21	
3 fuori, 5 dentro	$(0,0,0,1,1)$	16	22	
4 fuori, 1 dentro	$(1,0,1,0,0)$	10	23	
4 fuori, 2 dentro	$(0,1,1,0,0)$	11	20	
4 fuori, 5 dentro	$(0,0,1,0,1)$	15	21	

Nessun vicino ammissibile ha un profitto maggiore: $x = (0,0,1,1,0)$ è un **ottimo locale**, di profitto 27 (8 vicini ammissibili esaminati).

4. La ricerca locale da GR raggiunge $x = (1, 0, 0, 1, 1)$, di profitto $32 = z(I)$: una soluzione **ottima**. La ricerca locale dalla soluzione greedy per profitto si ferma subito in $x = (0, 0, 1, 1, 0)$, di profitto 27, un ottimo locale che non è ottimo. Per passare da $\{3, 4\}$ all'insieme ottimo $\{1, 4, 5\}$ occorre togliere l'oggetto 3 e aggiungere gli oggetti 1 e 5: tre flip, oppure uno swap e un flip, e ogni vettore intermedio è non ammissibile o peggiore. L'ottimo locale raggiunto dipende dalla **soluzione di partenza**.

Esercizio 3.2 — Ricerca locale KP 2

Consideriamo uno zaino di capacità $c = 13$ e un insieme $\mathscr{I} = \{1, 2, 3, 4, 5\}$ di 5 oggetti ($n = 5$), con i seguenti profitti e pesi:

	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$
p_j	8	9	14	11	9
w_j	3	5	9	9	8

Gli oggetti sono ordinati per efficienza non crescente. L'intorno flip+swap contiene i vettori ammissibili ottenuti con un flip (aggiungere o togliere un oggetto) o con uno swap (togliere un oggetto selezionato j, aggiungere un oggetto non selezionato k); si esaminano prima i flip, per $j = 1, \ldots, n$, poi gli swap, per j crescente e, per ogni j, per k crescente. In best improvement i pareggi si risolvono a favore della mossa esaminata per prima.

Domande

1. Calcolare la soluzione dell'algoritmo greedy GR (per efficienza) e il suo profitto.
2. Applicare la ricerca locale con l'intorno flip+swap e la regola **first improvement**, partendo dalla soluzione di GR.
3. Applicare la stessa ricerca locale con la regola **best improvement**, partendo dalla soluzione di GR.
4. Confrontare i due ottimi locali, e il numero di vicini ammissibili esaminati dalle due regole. Uno dei due ottimi locali è ottimo? (Suggerimento: qui il bound del rilassamento lineare non basta; dividere le soluzioni a seconda che l'oggetto 3 sia selezionato o no.)

Soluzione

1. Le efficienze sono $\frac{8}{3}$, $\frac{9}{5}$, $\frac{14}{9}$, $\frac{11}{9}$, $\frac{9}{8}$, non crescenti. Con $rc = 13$:
   - l'oggetto 1 ($w_1 = 3$) è selezionato, $rc = 10$;
   - l'oggetto 2 ($w_2 = 5$) è selezionato, $rc = 5$;
   - l'oggetto 3 ($w_3 = 9 > rc = 5$) non è selezionato;
   - l'oggetto 4 ($w_4 = 9 > rc = 5$) non è selezionato;

- l'oggetto 5 ($w_5 = 8 > rc = 5$) non è selezionato;

GR restituisce $x = (1, 1, 0, 0, 0)$, di peso 8 e profitto $GR(I) = 17$.

2. Iterazione 1: soluzione corrente $x = (1, 1, 0, 0, 0)$, peso 8, profitto 17.

mossa	vicino	peso	profitto	
flip 1	$(0, 1, 0, 0, 0)$	5	9	
flip 2	$(1, 0, 0, 0, 0)$	3	8	
flip 3	$(1, 1, 1, 0, 0)$	17	—	non ammissibile
flip 4	$(1, 1, 0, 1, 0)$	17	—	non ammissibile
flip 5	$(1, 1, 0, 0, 1)$	16	—	non ammissibile
1 fuori, 3 dentro	$(0, 1, 1, 0, 0)$	14	—	non ammissibile
1 fuori, 4 dentro	$(0, 1, 0, 1, 0)$	14	—	non ammissibile
1 fuori, 5 dentro	$(0, 1, 0, 0, 1)$	13	18	← accettata
2 fuori, 3 dentro	$(1, 0, 1, 0, 0)$	12	22	
2 fuori, 4 dentro	$(1, 0, 0, 1, 0)$	12	19	
2 fuori, 5 dentro	$(1, 0, 0, 0, 1)$	11	17	

La mossa è accettata dopo aver esaminato 3 vicini ammissibili.

Iterazione 2: soluzione corrente $x = (0, 1, 0, 0, 1)$, peso 13, profitto 18.

mossa	vicino	peso	profitto	
flip 1	$(1, 1, 0, 0, 1)$	16	—	non ammissibile
flip 2	$(0, 0, 0, 0, 1)$	8	9	
flip 3	$(0, 1, 1, 0, 1)$	22	—	non ammissibile
flip 4	$(0, 1, 0, 1, 1)$	22	—	non ammissibile
flip 5	$(0, 1, 0, 0, 0)$	5	9	
2 fuori, 1 dentro	$(1, 0, 0, 0, 1)$	11	17	
2 fuori, 3 dentro	$(0, 0, 1, 0, 1)$	17	—	non ammissibile
2 fuori, 4 dentro	$(0, 0, 0, 1, 1)$	17	—	non ammissibile
5 fuori, 1 dentro	$(1, 1, 0, 0, 0)$	8	17	
5 fuori, 3 dentro	$(0, 1, 1, 0, 0)$	14	—	non ammissibile
5 fuori, 4 dentro	$(0, 1, 0, 1, 0)$	14	—	non ammissibile

Nessun vicino ammissibile ha un profitto maggiore: $x = (0, 1, 0, 0, 1)$ è un **ottimo locale**, di profitto 18 (4 vicini ammissibili esaminati).

First improvement si ferma in $x = (0, 1, 0, 0, 1)$, di profitto 18, dopo aver esaminato 7 vicini ammissibili in totale.

3. Iterazione 1: soluzione corrente $x = (1, 1, 0, 0, 0)$, peso 8, profitto 17.

mossa	vicino	peso	profitto	
flip 1	$(0,1,0,0,0)$	5	9	
flip 2	$(1,0,0,0,0)$	3	8	
flip 3	$(1,1,1,0,0)$	17	—	non ammissibile
flip 4	$(1,1,0,1,0)$	17	—	non ammissibile
flip 5	$(1,1,0,0,1)$	16	—	non ammissibile
1 fuori, 3 dentro	$(0,1,1,0,0)$	14	—	non ammissibile
1 fuori, 4 dentro	$(0,1,0,1,0)$	14	—	non ammissibile
1 fuori, 5 dentro	$(0,1,0,0,1)$	13	18	
2 fuori, 3 dentro	$(1,0,1,0,0)$	12	22	← accettata
2 fuori, 4 dentro	$(1,0,0,1,0)$	12	19	
2 fuori, 5 dentro	$(1,0,0,0,1)$	11	17	

La mossa è accettata dopo aver esaminato 6 vicini ammissibili.

Iterazione 2: soluzione corrente $x = (1,0,1,0,0)$, peso 12, profitto 22.

mossa	vicino	peso	profitto	
flip 1	$(0,0,1,0,0)$	9	14	
flip 2	$(1,1,1,0,0)$	17	—	non ammissibile
flip 3	$(1,0,0,0,0)$	3	8	
flip 4	$(1,0,1,1,0)$	21	—	non ammissibile
flip 5	$(1,0,1,0,1)$	20	—	non ammissibile
1 fuori, 2 dentro	$(0,1,1,0,0)$	14	—	non ammissibile
1 fuori, 4 dentro	$(0,0,1,1,0)$	18	—	non ammissibile
1 fuori, 5 dentro	$(0,0,1,0,1)$	17	—	non ammissibile
3 fuori, 2 dentro	$(1,1,0,0,0)$	8	17	
3 fuori, 4 dentro	$(1,0,0,1,0)$	12	19	
3 fuori, 5 dentro	$(1,0,0,0,1)$	11	17	

Nessun vicino ammissibile ha un profitto maggiore: $x = (1,0,1,0,0)$ è un **ottimo locale**, di profitto 22 (5 vicini ammissibili esaminati).

Best improvement si ferma in $x = (1,0,1,0,0)$, di profitto 22, dopo aver esaminato 11 vicini ammissibili in totale.

4. First improvement accetta il primo swap migliorante che trova, che guadagna solo 1, e si ferma in un ottimo locale di profitto 18; best improvement prende il guadagno massimo e raggiunge profitto 22. Esaminano 7 e 11 vicini ammissibili. Il rilassamento lineare dà $z(I) \leq 8 + 9 + 14 \cdot \frac{5}{9} = \frac{223}{9}$, quindi $z(I) \leq 24$, che non dimostra che 22 sia ottimo. Ragioniamo direttamente. Se l'oggetto 3 è selezionato, la capacità residua è 4 ed entra solo l'oggetto 1, per un profitto di 22. Se l'oggetto 3 non è selezionato, la soluzione migliore è $\{1,4\}$, di profitto 19. Quindi $z(I) = 22$ e la soluzione di best improvement, $x = (1,0,1,0,0)$, è **ottima**. Dalla stessa soluzione di partenza le due regole raggiungono **ottimi locali diversi**.

Esercizio 3.3 — Ricerca locale KP 3

Consideriamo uno zaino di capacità $c = 11$ e un insieme $\mathscr{I} = \{1, 2, 3, 4, 5\}$ di 5 oggetti ($n = 5$), con i seguenti profitti e pesi:

	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$
p_j	14	7	10	8	6
w_j	4	4	7	7	9

Gli oggetti sono ordinati per efficienza non crescente. Con le penalità, la ricerca locale massimizza $g_\alpha(x) = \sum_j p_j x_j - \alpha \max\{0, \sum_j w_j x_j - c\}$ su tutti i vettori binari, con l'intorno flip (flip esaminati per $j = 1, \ldots, n$) e la regola best improvement.

Domande

1. Calcolare la soluzione di GR, e mostrare che è un ottimo locale per l'intorno flip ristretto ai vettori ammissibili.
2. Applicare la ricerca locale penalty flip con $\alpha = 1$, partendo dalla soluzione di GR. L'ottimo locale è ammissibile? Quale soluzione deve restituire la ricerca?
3. Applicare la ricerca locale penalty flip con $\alpha = 2$, partendo dalla soluzione di GR.
4. Applicarla con $\alpha = 3$. Il valore ottimo è $z(I) = 24$: discutere il ruolo di α, e indicare per quali valori di α gli ottimi locali sono sicuramente ammissibili.

Soluzione

1. Con $rc = 11$:
   - l'oggetto 1 ($w_1 = 4$) è selezionato, $rc = 7$;
   - l'oggetto 2 ($w_2 = 4$) è selezionato, $rc = 3$;
   - l'oggetto 3 ($w_3 = 7 > rc = 3$) non è selezionato;
   - l'oggetto 4 ($w_4 = 7 > rc = 3$) non è selezionato;
   - l'oggetto 5 ($w_5 = 9 > rc = 3$) non è selezionato;

   GR restituisce $x = (1, 1, 0, 0, 0)$, di peso 8 e profitto 21. I suoi flip sono

mossa	vicino	peso	profitto	
flip 1	$(0, 1, 0, 0, 0)$	4	7	
flip 2	$(1, 0, 0, 0, 0)$	4	14	
flip 3	$(1, 1, 1, 0, 0)$	15	—	non ammissibile
flip 4	$(1, 1, 0, 1, 0)$	15	—	non ammissibile
flip 5	$(1, 1, 0, 0, 1)$	17	—	non ammissibile

I vicini ammissibili tolgono un oggetto e hanno un profitto minore: x è un ottimo locale per l'intorno flip ammissibile.

2. Iterazione 1: $x = (1, 1, 0, 0, 0)$, peso 8, $g_1(x) = 21$.

mossa	vicino	peso	eccesso	g_1	
flip 1	$(0, 1, 0, 0, 0)$	4	0	$7 - 1 \cdot 0 = 7$	
flip 2	$(1, 0, 0, 0, 0)$	4	0	$14 - 1 \cdot 0 = 14$	
flip 3	$(1, 1, 1, 0, 0)$	15	4	$31 - 1 \cdot 4 = 27$	$\leftarrow$
flip 4	$(1, 1, 0, 1, 0)$	15	4	$29 - 1 \cdot 4 = 25$	
flip 5	$(1, 1, 0, 0, 1)$	17	6	$27 - 1 \cdot 6 = 21$	

Iterazione 2: $x = (1, 1, 1, 0, 0)$, peso 15, $g_1(x) = 27$.

mossa	vicino	peso	eccesso	g_1	
flip 1	$(0, 1, 1, 0, 0)$	11	0	$17 - 1 \cdot 0 = 17$	
flip 2	$(1, 0, 1, 0, 0)$	11	0	$24 - 1 \cdot 0 = 24$	
flip 3	$(1, 1, 0, 0, 0)$	8	0	$21 - 1 \cdot 0 = 21$	
flip 4	$(1, 1, 1, 1, 0)$	22	11	$39 - 1 \cdot 11 = 28$	$\leftarrow$
flip 5	$(1, 1, 1, 0, 1)$	24	13	$37 - 1 \cdot 13 = 24$	

Iterazione 3: $x = (1, 1, 1, 1, 0)$, peso 22, $g_1(x) = 28$.

mossa	vicino	peso	eccesso	g_1
flip 1	$(0, 1, 1, 1, 0)$	18	7	$25 - 1 \cdot 7 = 18$
flip 2	$(1, 0, 1, 1, 0)$	18	7	$32 - 1 \cdot 7 = 25$
flip 3	$(1, 1, 0, 1, 0)$	15	4	$29 - 1 \cdot 4 = 25$
flip 4	$(1, 1, 1, 0, 0)$	15	4	$31 - 1 \cdot 4 = 27$
flip 5	$(1, 1, 1, 1, 1)$	31	20	$45 - 1 \cdot 20 = 25$

Nessun flip aumenta g_1: la ricerca si ferma in $x = (1, 1, 1, 1, 0)$, di peso 22 e profitto 39.

L'ottimo locale è **non ammissibile** (peso $22 > 11$): la penalità è troppo piccola. La ricerca deve restituire il miglior vettore ammissibile visitato, $(1, 1, 0, 0, 0)$, di profitto 21, cioè la soluzione di partenza.

3. Iterazione 1: $x = (1, 1, 0, 0, 0)$, peso 8, $g_2(x) = 21$.

mossa	vicino	peso	eccesso	g_2	
flip 1	$(0, 1, 0, 0, 0)$	4	0	$7 - 2 \cdot 0 = 7$	
flip 2	$(1, 0, 0, 0, 0)$	4	0	$14 - 2 \cdot 0 = 14$	
flip 3	$(1, 1, 1, 0, 0)$	15	4	$31 - 2 \cdot 4 = 23$	$\leftarrow$
flip 4	$(1, 1, 0, 1, 0)$	15	4	$29 - 2 \cdot 4 = 21$	
flip 5	$(1, 1, 0, 0, 1)$	17	6	$27 - 2 \cdot 6 = 15$	

Iterazione 2: $x = (1, 1, 1, 0, 0)$, peso 15, $g_2(x) = 23$.

mossa	vicino	peso	eccesso	g_2	
flip 1	$(0, 1, 1, 0, 0)$	11	0	$17 - 2 \cdot 0 = 17$	
flip 2	$(1, 0, 1, 0, 0)$	11	0	$24 - 2 \cdot 0 = 24$	$\leftarrow$
flip 3	$(1, 1, 0, 0, 0)$	8	0	$21 - 2 \cdot 0 = 21$	
flip 4	$(1, 1, 1, 1, 0)$	22	11	$39 - 2 \cdot 11 = 17$	
flip 5	$(1, 1, 1, 0, 1)$	24	13	$37 - 2 \cdot 13 = 11$	

Iterazione 3: $x = (1, 0, 1, 0, 0)$, peso 11, $g_2(x) = 24$.

mossa	vicino	peso	eccesso	g_2
flip 1	$(0, 0, 1, 0, 0)$	7	0	$10 - 2 \cdot 0 = 10$
flip 2	$(1, 1, 1, 0, 0)$	15	4	$31 - 2 \cdot 4 = 23$
flip 3	$(1, 0, 0, 0, 0)$	4	0	$14 - 2 \cdot 0 = 14$
flip 4	$(1, 0, 1, 1, 0)$	18	7	$32 - 2 \cdot 7 = 18$
flip 5	$(1, 0, 1, 0, 1)$	20	9	$30 - 2 \cdot 9 = 12$

Nessun flip aumenta g_2: la ricerca si ferma in $x = (1, 0, 1, 0, 0)$, di peso 11 e profitto 24.

L'ottimo locale $(1, 0, 1, 0, 0)$ è ammissibile, di profitto $24 = z(I)$: è **ottimo**. La ricerca lo ha raggiunto attraverso il vettore non ammissibile $(1, 1, 1, 0, 0)$.

4. Iterazione 1: $x = (1, 1, 0, 0, 0)$, peso 8, $g_3(x) = 21$.

mossa	vicino	peso	eccesso	g_3
flip 1	$(0, 1, 0, 0, 0)$	4	0	$7 - 3 \cdot 0 = 7$
flip 2	$(1, 0, 0, 0, 0)$	4	0	$14 - 3 \cdot 0 = 14$
flip 3	$(1, 1, 1, 0, 0)$	15	4	$31 - 3 \cdot 4 = 19$
flip 4	$(1, 1, 0, 1, 0)$	15	4	$29 - 3 \cdot 4 = 17$
flip 5	$(1, 1, 0, 0, 1)$	17	6	$27 - 3 \cdot 6 = 9$

Nessun flip aumenta g_3: la ricerca si ferma in $x = (1, 1, 0, 0, 0)$, di peso 8 e profitto 21.

Con $\alpha = 3$ la ricerca non si muove: si comporta come la ricerca ammissibile e si ferma nella soluzione di GR, di profitto 21. Un α piccolo (1) lascia che la ricerca si fermi fra vettori non ammissibili; un α grande (3) le impedisce di attraversare la regione non ammissibile; qui solo $\alpha = 2$ raggiunge l'ottimo. Se $\alpha > \max_j p_j = 14$, ogni vettore non ammissibile ha un flip migliorante (togliere un qualsiasi oggetto selezionato diminuisce l'eccesso di almeno 1 e guadagna almeno $\alpha - p_j > 0$), quindi tutti gli ottimi locali sono ammissibili; ma allora, da un vettore ammissibile, nessun flip che crea un eccesso è migliorante, e la ricerca non lascia mai la regione ammissibile.

3.2 Bin packing

Esercizio 3.4 — Ricerca locale BPP 1

Contenitori di capacità $c = 10$ e un insieme $\mathscr{I} = \{1, 2, 3, 4, 5, 6, 7\}$ di 7 oggetti con pesi

	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$	$j=7$
w_j	2	3	2	7	4	7	4

I contenitori sono etichettati $1, 2, \ldots$ nell'ordine in cui FF li apre, e mantengono la loro etichetta. Una *move* mette un oggetto in un altro contenitore non vuoto in cui entra (esaminata per $j = 1, \ldots, n$ e, per ogni j, per etichetta di contenitore crescente); uno *swap* scambia due oggetti $j < k$ di peso diverso in contenitori diversi, se entrambi i contenitori restano ammissibili. L'obiettivo ausiliario $SQ = \sum_b L_b^2$ si massimizza; il guadagno di un trasferimento di δ dal contenitore a al contenitore b è $2\delta(L_b - L_a + \delta)$ (una move dell'oggetto j trasferisce $\delta = w_j$, uno swap dell'oggetto j nel contenitore a e dell'oggetto k nel contenitore b trasferisce $\delta = w_j - w_k$). Best improvement risolve i pareggi a favore della mossa esaminata per prima.

Domande

1. Calcolare la soluzione di First Fit (FF), il suo numero di contenitori e il lower bound $\lceil \sum_j w_j / c \rceil$.
2. Applicare la ricerca locale move con l'obiettivo nb (numero di contenitori), partendo dalla soluzione di FF. Perché si ferma?
3. Applicare la ricerca locale move con l'obiettivo ausiliario $SQ = \sum_b L_b^2$ e la regola best improvement, partendo dalla soluzione di FF.
4. La soluzione finale è ottima? Spiegare il diverso comportamento dei due obiettivi.

Soluzione

1.
   - l'oggetto 1 ($w_1 = 2$) va nel contenitore 1, aperto per lui, carico 2;
   - l'oggetto 2 ($w_2 = 3$) va nel contenitore 1, carico 5;
   - l'oggetto 3 ($w_3 = 2$) va nel contenitore 1, carico 7;
   - l'oggetto 4 ($w_4 = 7$) non entra in nessun contenitore aperto: si apre il contenitore 2, carico 7;
   - l'oggetto 5 ($w_5 = 4$) non entra in nessun contenitore aperto: si apre il contenitore 3, carico 4;
   - l'oggetto 6 ($w_6 = 7$) non entra in nessun contenitore aperto: si apre il contenitore 4, carico 7;
   - l'oggetto 7 ($w_7 = 4$) va nel contenitore 3, carico 8;

FF usa $nb = 4$ contenitori, con schemi di carico $\{1,2,3\},\{4\},\{5,7\},\{6\}$ e carichi $(7,7,8,7)$. Il lower bound è $\lceil 29/10 \rceil = 3$.

2. Iterazione 1 (move): schemi di carico $\{1,2,3\},\{4\},\{5,7\},\{6\}$, carichi $(7,7,8,7)$, $nb = 4$, $SQ = 211$.

mossa	carichi	contenitore svuotato
oggetto 1: contenitore $1 \to 2$	$(5,9,8,7)$	no
oggetto 1: contenitore $1 \to 3$	$(5,7,10,7)$	no
oggetto 1: contenitore $1 \to 4$	$(5,7,8,9)$	no
oggetto 2: contenitore $1 \to 2$	$(4,10,8,7)$	no
oggetto 2: contenitore $1 \to 4$	$(4,7,8,10)$	no
oggetto 3: contenitore $1 \to 2$	$(5,9,8,7)$	no
oggetto 3: contenitore $1 \to 3$	$(5,7,10,7)$	no
oggetto 3: contenitore $1 \to 4$	$(5,7,8,9)$	no

Nessuna mossa è migliorante.

Nessuna mossa svuota un contenitore, quindi nessuna mossa diminuisce nb: la soluzione di FF è un ottimo locale con 4 contenitori. Tutte le mosse sono neutre per nb: la ricerca è su un **plateau**.

3. Iterazione 1 (move): schemi di carico $\{1,2,3\},\{4\},\{5,7\},\{6\}$, carichi $(7,7,8,7)$, $nb = 4$, $SQ = 211$.

mossa	carichi	guadagno	
oggetto 1: contenitore $1 \to 2$	$(5,9,8,7)$	$2 \cdot 2 \cdot (7-7+2) = 8$	
oggetto 1: contenitore $1 \to 3$	$(5,7,10,7)$	$2 \cdot 2 \cdot (8-7+2) = 12$	
oggetto 1: contenitore $1 \to 4$	$(5,7,8,9)$	$2 \cdot 2 \cdot (7-7+2) = 8$	
oggetto 2: contenitore $1 \to 2$	$(4,10,8,7)$	$2 \cdot 3 \cdot (7-7+3) = 18$	$\leftarrow$
oggetto 2: contenitore $1 \to 4$	$(4,7,8,10)$	$2 \cdot 3 \cdot (7-7+3) = 18$	
oggetto 3: contenitore $1 \to 2$	$(5,9,8,7)$	$2 \cdot 2 \cdot (7-7+2) = 8$	
oggetto 3: contenitore $1 \to 3$	$(5,7,10,7)$	$2 \cdot 2 \cdot (8-7+2) = 12$	
oggetto 3: contenitore $1 \to 4$	$(5,7,8,9)$	$2 \cdot 2 \cdot (7-7+2) = 8$	

Si accetta la mossa indicata.

Iterazione 2 (move): schemi di carico $\{1,3\},\{2,4\},\{5,7\},\{6\}$, carichi $(4,10,8,7)$, $nb = 4$, $SQ = 229$.

mossa	carichi	guadagno	
oggetto 1: contenitore $1 \to 3$	$(2, 10, 10, 7)$	$2 \cdot 2 \cdot (8 - 4 + 2) = 24$	$\leftarrow$
oggetto 1: contenitore $1 \to 4$	$(2, 10, 8, 9)$	$2 \cdot 2 \cdot (7 - 4 + 2) = 20$	
oggetto 2: contenitore $2 \to 1$	$(7, 7, 8, 7)$	$2 \cdot 3 \cdot (4 - 10 + 3) = -18$	
oggetto 2: contenitore $2 \to 4$	$(4, 7, 8, 10)$	$2 \cdot 3 \cdot (7 - 10 + 3) = 0$	
oggetto 3: contenitore $1 \to 3$	$(2, 10, 10, 7)$	$2 \cdot 2 \cdot (8 - 4 + 2) = 24$	
oggetto 3: contenitore $1 \to 4$	$(2, 10, 8, 9)$	$2 \cdot 2 \cdot (7 - 4 + 2) = 20$	
oggetto 5: contenitore $3 \to 1$	$(8, 10, 4, 7)$	$2 \cdot 4 \cdot (4 - 8 + 4) = 0$	
oggetto 7: contenitore $3 \to 1$	$(8, 10, 4, 7)$	$2 \cdot 4 \cdot (4 - 8 + 4) = 0$	

Si accetta la mossa indicata.

Iterazione 3 (move): schemi di carico $\{3\}, \{2, 4\}, \{1, 5, 7\}, \{6\}$, carichi $(2, 10, 10, 7)$, $nb = 4$, $SQ = 253$.

mossa	carichi	guadagno	
oggetto 1: contenitore $3 \to 1$	$(4, 10, 8, 7)$	$2 \cdot 2 \cdot (2 - 10 + 2) = -24$	
oggetto 1: contenitore $3 \to 4$	$(2, 10, 8, 9)$	$2 \cdot 2 \cdot (7 - 10 + 2) = -4$	
oggetto 2: contenitore $2 \to 1$	$(5, 7, 10, 7)$	$2 \cdot 3 \cdot (2 - 10 + 3) = -30$	
oggetto 2: contenitore $2 \to 4$	$(2, 7, 10, 10)$	$2 \cdot 3 \cdot (7 - 10 + 3) = 0$	
oggetto 3: contenitore $1 \to 4$	$(10, 10, 9)$	$2 \cdot 2 \cdot (7 - 2 + 2) = 28$	$\leftarrow$
oggetto 4: contenitore $2 \to 1$	$(9, 3, 10, 7)$	$2 \cdot 7 \cdot (2 - 10 + 7) = -14$	
oggetto 5: contenitore $3 \to 1$	$(6, 10, 6, 7)$	$2 \cdot 4 \cdot (2 - 10 + 4) = -32$	
oggetto 6: contenitore $4 \to 1$	$(9, 10, 10)$	$2 \cdot 7 \cdot (2 - 7 + 7) = 28$	
oggetto 7: contenitore $3 \to 1$	$(6, 10, 6, 7)$	$2 \cdot 4 \cdot (2 - 10 + 4) = -32$	

Si accetta la mossa indicata.

Iterazione 4 (move): schemi di carico $\{2, 4\}, \{1, 5, 7\}, \{3, 6\}$, carichi $(10, 10, 9)$, $nb = 3$, $SQ = 281$. Non c'è nessuna mossa ammissibile.

La ricerca esegue 3 mosse, con SQ: $211 \to 229 \to 253 \to 281$, e si ferma con schemi di carico $\{2, 4\}, \{1, 5, 7\}, \{3, 6\}$ e $nb = 3$.

4. La soluzione finale usa $3 = \lceil 29/10 \rceil$ contenitori: è **ottima**. Con nb la ricerca non vede differenze fra i vicini, perché solo una mossa che svuota un contenitore cambia nb; con SQ sono miglioranti le mosse che rendono più pesante un contenitore pesante e più leggero un contenitore leggero, e preparano la mossa che svuota un contenitore (che aumenta sempre SQ).

Esercizio 3.5 — Ricerca locale BPP 2

Contenitori di capacità $c = 10$ e un insieme $\mathscr{I} = \{1, 2, 3, 4, 5, 6\}$ di 6 oggetti con pesi

	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$
w_j	2	3	4	5	4	2

I contenitori sono etichettati $1, 2, \ldots$ nell'ordine in cui FF li apre, e mantengono la loro etichetta. Una *move* mette un oggetto in un altro contenitore non vuoto in cui entra (esaminata per $j = 1, \ldots, n$ e, per ogni j, per etichetta di contenitore crescente); uno *swap* scambia due oggetti $j < k$ di peso diverso in contenitori diversi, se entrambi i contenitori restano ammissibili. L'obiettivo ausiliario $SQ = \sum_b L_b^2$ si massimizza; il guadagno di un trasferimento di δ dal contenitore a al contenitore b è $2\delta(L_b - L_a + \delta)$ (una move dell'oggetto j trasferisce $\delta = w_j$, uno swap dell'oggetto j nel contenitore a e dell'oggetto k nel contenitore b trasferisce $\delta = w_j - w_k$). Best improvement risolve i pareggi a favore della mossa esaminata per prima.

Domande

1. Calcolare la soluzione di First Fit (FF) e il lower bound $\lceil \sum_j w_j / c \rceil$.
2. Applicare la ricerca locale move con l'obiettivo ausiliario SQ e la regola best improvement, partendo dalla soluzione di FF.
3. Applicare la ricerca locale swap con l'obiettivo SQ e la regola best improvement, partendo dalla soluzione di FF.
4. Applicare la variable neighbourhood descent con $N_1 =$ move e $N_2 =$ swap (obiettivo SQ, best improvement), partendo dalla soluzione di FF. Il risultato è ottimo? Che cosa mostrano le tre ricerche locali?

Soluzione

1. - l'oggetto 1 ($w_1 = 2$) va nel contenitore 1, aperto per lui, carico 2;
   - l'oggetto 2 ($w_2 = 3$) va nel contenitore 1, carico 5;
   - l'oggetto 3 ($w_3 = 4$) va nel contenitore 1, carico 9;
   - l'oggetto 4 ($w_4 = 5$) non entra in nessun contenitore aperto: si apre il contenitore 2, carico 5;
   - l'oggetto 5 ($w_5 = 4$) va nel contenitore 2, carico 9;
   - l'oggetto 6 ($w_6 = 2$) non entra in nessun contenitore aperto: si apre il contenitore 3, carico 2;

   FF usa $nb = 3$ contenitori, con schemi di carico $\{1, 2, 3\}, \{4, 5\}, \{6\}$ e carichi $(9, 9, 2)$. Il lower bound è $\lceil 20/10 \rceil = 2$.
2. Iterazione 1 (move): schemi di carico $\{1, 2, 3\}, \{4, 5\}, \{6\}$, carichi $(9, 9, 2)$, $nb = 3$, $SQ = 166$.

mossa	carichi	guadagno
oggetto 1: contenitore $1 \to 3$	$(7, 9, 4)$	$2 \cdot 2 \cdot (2 - 9 + 2) = -20$
oggetto 2: contenitore $1 \to 3$	$(6, 9, 5)$	$2 \cdot 3 \cdot (2 - 9 + 3) = -24$
oggetto 3: contenitore $1 \to 3$	$(5, 9, 6)$	$2 \cdot 4 \cdot (2 - 9 + 4) = -24$
oggetto 4: contenitore $2 \to 3$	$(9, 4, 7)$	$2 \cdot 5 \cdot (2 - 9 + 5) = -20$
oggetto 5: contenitore $2 \to 3$	$(9, 5, 6)$	$2 \cdot 4 \cdot (2 - 9 + 4) = -24$

Nessuna mossa è migliorante.

La soluzione di FF è un ottimo locale per move, con 3 contenitori.

3. Iterazione 1 (swap): schemi di carico $\{1, 2, 3\}, \{4, 5\}, \{6\}$, carichi $(9, 9, 2)$, $nb = 3$, $SQ = 166$.

swap	carichi	guadagno	
oggetti $2, 5$ (contenitori $1, 2$)	$(10, 8, 2)$	$2 \cdot (-1) \cdot (9 - 9 - 1) = 2$	$\leftarrow$
oggetti $2, 6$ (contenitori $1, 3$)	$(8, 9, 3)$	$2 \cdot (1) \cdot (2 - 9 + 1) = -12$	
oggetti $3, 4$ (contenitori $1, 2$)	$(10, 8, 2)$	$2 \cdot (-1) \cdot (9 - 9 - 1) = 2$	
oggetti $3, 6$ (contenitori $1, 3$)	$(7, 9, 4)$	$2 \cdot (2) \cdot (2 - 9 + 2) = -20$	
oggetti $4, 6$ (contenitori $2, 3$)	$(9, 6, 5)$	$2 \cdot (3) \cdot (2 - 9 + 3) = -24$	
oggetti $5, 6$ (contenitori $2, 3$)	$(9, 7, 4)$	$2 \cdot (2) \cdot (2 - 9 + 2) = -20$	

Si accetta lo swap indicato.

Iterazione 2 (swap): schemi di carico $\{1, 3, 5\}, \{2, 4\}, \{6\}$, carichi $(10, 8, 2)$, $nb = 3$, $SQ = 168$.

swap	carichi	guadagno
oggetti $2, 3$ (contenitori $2, 1$)	$(9, 9, 2)$	$2 \cdot (-1) \cdot (10 - 8 - 1) = -2$
oggetti $2, 5$ (contenitori $2, 1$)	$(9, 9, 2)$	$2 \cdot (-1) \cdot (10 - 8 - 1) = -2$
oggetti $2, 6$ (contenitori $2, 3$)	$(10, 7, 3)$	$2 \cdot (1) \cdot (2 - 8 + 1) = -10$
oggetti $3, 6$ (contenitori $1, 3$)	$(8, 8, 4)$	$2 \cdot (2) \cdot (2 - 10 + 2) = -24$
oggetti $4, 6$ (contenitori $2, 3$)	$(10, 5, 5)$	$2 \cdot (3) \cdot (2 - 8 + 3) = -18$
oggetti $5, 6$ (contenitori $1, 3$)	$(8, 8, 4)$	$2 \cdot (2) \cdot (2 - 10 + 2) = -24$

Nessuno swap è migliorante.

La ricerca locale swap si ferma con schemi di carico $\{1, 3, 5\}, \{2, 4\}, \{6\}$, $SQ = 168$, ancora 3 contenitori (uno swap non cambia mai nb).

4. Iterazione 1 (move): schemi di carico $\{1, 2, 3\}, \{4, 5\}, \{6\}$, carichi $(9, 9, 2)$, $nb = 3$, $SQ = 166$.

mossa	carichi	guadagno
oggetto 1: contenitore $1 \to 3$	$(7, 9, 4)$	$2 \cdot 2 \cdot (2 - 9 + 2) = -20$
oggetto 2: contenitore $1 \to 3$	$(6, 9, 5)$	$2 \cdot 3 \cdot (2 - 9 + 3) = -24$
oggetto 3: contenitore $1 \to 3$	$(5, 9, 6)$	$2 \cdot 4 \cdot (2 - 9 + 4) = -24$
oggetto 4: contenitore $2 \to 3$	$(9, 4, 7)$	$2 \cdot 5 \cdot (2 - 9 + 5) = -20$
oggetto 5: contenitore $2 \to 3$	$(9, 5, 6)$	$2 \cdot 4 \cdot (2 - 9 + 4) = -24$

Nessuna mossa è migliorante.

Iterazione 2 (swap): schemi di carico $\{1, 2, 3\}, \{4, 5\}, \{6\}$, carichi $(9, 9, 2)$, $nb = 3$, $SQ = 166$.

swap	carichi	guadagno	
oggetti 2, 5 (contenitori 1, 2)	$(10, 8, 2)$	$2 \cdot (-1) \cdot (9 - 9 - 1) = 2$	$\leftarrow$
oggetti 2, 6 (contenitori 1, 3)	$(8, 9, 3)$	$2 \cdot (1) \cdot (2 - 9 + 1) = -12$	
oggetti 3, 4 (contenitori 1, 2)	$(10, 8, 2)$	$2 \cdot (-1) \cdot (9 - 9 - 1) = 2$	
oggetti 3, 6 (contenitori 1, 3)	$(7, 9, 4)$	$2 \cdot (2) \cdot (2 - 9 + 2) = -20$	
oggetti 4, 6 (contenitori 2, 3)	$(9, 6, 5)$	$2 \cdot (3) \cdot (2 - 9 + 3) = -24$	
oggetti 5, 6 (contenitori 2, 3)	$(9, 7, 4)$	$2 \cdot (2) \cdot (2 - 9 + 2) = -20$	

Si accetta lo swap indicato, e la ricerca torna all'intorno move.

Iterazione 3 (move): schemi di carico $\{1, 3, 5\}, \{2, 4\}, \{6\}$, carichi $(10, 8, 2)$, $nb = 3$, $SQ = 168$.

mossa	carichi	guadagno	
oggetto 1: contenitore $1 \to 2$	$(8, 10, 2)$	$2 \cdot 2 \cdot (8 - 10 + 2) = 0$	
oggetto 1: contenitore $1 \to 3$	$(8, 8, 4)$	$2 \cdot 2 \cdot (2 - 10 + 2) = -24$	
oggetto 2: contenitore $2 \to 3$	$(10, 5, 5)$	$2 \cdot 3 \cdot (2 - 8 + 3) = -18$	
oggetto 3: contenitore $1 \to 3$	$(6, 8, 6)$	$2 \cdot 4 \cdot (2 - 10 + 4) = -32$	
oggetto 4: contenitore $2 \to 3$	$(10, 3, 7)$	$2 \cdot 5 \cdot (2 - 8 + 5) = -10$	
oggetto 5: contenitore $1 \to 3$	$(6, 8, 6)$	$2 \cdot 4 \cdot (2 - 10 + 4) = -32$	
oggetto 6: contenitore $3 \to 2$	$(10, 10)$	$2 \cdot 2 \cdot (8 - 2 + 2) = 32$	$\leftarrow$

Si accetta la mossa indicata.

Iterazione 4 (move): schemi di carico $\{1, 3, 5\}, \{2, 4, 6\}$, carichi $(10, 10)$, $nb = 2$, $SQ = 200$. Non c'è nessuna mossa ammissibile.

Iterazione 5 (swap): schemi di carico $\{1, 3, 5\}, \{2, 4, 6\}$, carichi $(10, 10)$, $nb = 2$, $SQ = 200$. Non c'è nessuno swap ammissibile.

VND si ferma con schemi di carico $\{1, 3, 5\}, \{2, 4, 6\}$: $2 = \lceil 20/10 \rceil$ contenitori, una soluzione **ottima**. La soluzione di FF è un ottimo locale per move, il solo swap non può cambiare il numero di contenitori, ma lo swap prepara una move che svuota un contenitore: l'ottimo locale **dipende dall'intorno**, e combinando i due intorni si raggiunge l'ottimo.

Esercizio 3.6 — Ricerca locale BPP 3

Contenitori di capacità $c = 10$ e un insieme $\mathscr{I} = \{1, 2, 3, 4, 5, 6, 7\}$ di 7 oggetti con pesi

	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$	$j=7$
w_j	5	3	3	2	8	4	4

I contenitori sono etichettati $1, 2, \ldots$ nell'ordine in cui FF li apre, e mantengono la loro etichetta. Una *move* mette un oggetto in un altro contenitore non vuoto in cui entra (esaminata per $j = 1, \ldots, n$ e, per ogni j, per etichetta di contenitore crescente); uno *swap* scambia due oggetti $j < k$ di peso diverso in contenitori diversi, se entrambi i contenitori restano ammissibili. L'obiettivo ausiliario $SQ = \sum_b L_b^2$ si massimizza; il guadagno di un trasferimento di δ dal contenitore a al contenitore b è $2\delta(L_b - L_a + \delta)$ (una move dell'oggetto j trasferisce $\delta = w_j$, uno swap dell'oggetto j nel contenitore a e dell'oggetto k nel contenitore b trasferisce $\delta = w_j - w_k$). Best improvement risolve i pareggi a favore della mossa esaminata per prima.

Domande

1. Calcolare la soluzione di First Fit (FF) e il lower bound $\lceil \sum_j w_j / c \rceil$.
2. Applicare la variable neighbourhood descent con $N_1 =$ move e $N_2 =$ swap (obiettivo SQ, best improvement), partendo dalla soluzione di FF.
3. Trovare un riempimento che usa $\lceil \sum_j w_j / c \rceil$ contenitori.
4. Spiegare perché la ricerca locale non può raggiungere quel riempimento dal suo ottimo locale.

Soluzione

1.
   - l'oggetto 1 ($w_1 = 5$) va nel contenitore 1, aperto per lui, carico 5;
   - l'oggetto 2 ($w_2 = 3$) va nel contenitore 1, carico 8;
   - l'oggetto 3 ($w_3 = 3$) non entra in nessun contenitore aperto: si apre il contenitore 2, carico 3;
   - l'oggetto 4 ($w_4 = 2$) va nel contenitore 1, carico 10;
   - l'oggetto 5 ($w_5 = 8$) non entra in nessun contenitore aperto: si apre il contenitore 3, carico 8;
   - l'oggetto 6 ($w_6 = 4$) va nel contenitore 2, carico 7;
   - l'oggetto 7 ($w_7 = 4$) non entra in nessun contenitore aperto: si apre il contenitore 4, carico 4;

   FF usa $nb = 4$ contenitori, con schemi di carico $\{1, 2, 4\}, \{3, 6\}, \{5\}, \{7\}$ e carichi $(10, 7, 8, 4)$. Il lower bound è $\lceil 29/10 \rceil = 3$.

2. Iterazione 1 (move): schemi di carico $\{1,2,4\},\{3,6\},\{5\},\{7\}$, carichi $(10,7,8,4)$, $nb = 4$, $SQ = 229$.

mossa	carichi	guadagno	
oggetto 1: contenitore $1 \to 4$	$(5,7,8,9)$	$2 \cdot 5 \cdot (4-10+5) = -10$	
oggetto 2: contenitore $1 \to 2$	$(7,10,8,4)$	$2 \cdot 3 \cdot (7-10+3) = 0$	
oggetto 2: contenitore $1 \to 4$	$(7,7,8,7)$	$2 \cdot 3 \cdot (4-10+3) = -18$	
oggetto 3: contenitore $2 \to 4$	$(10,4,8,7)$	$2 \cdot 3 \cdot (4-7+3) = 0$	
oggetto 4: contenitore $1 \to 2$	$(8,9,8,4)$	$2 \cdot 2 \cdot (7-10+2) = -4$	
oggetto 4: contenitore $1 \to 3$	$(8,7,10,4)$	$2 \cdot 2 \cdot (8-10+2) = 0$	
oggetto 4: contenitore $1 \to 4$	$(8,7,8,6)$	$2 \cdot 2 \cdot (4-10+2) = -16$	
oggetto 6: contenitore $2 \to 4$	$(10,3,8,8)$	$2 \cdot 4 \cdot (4-7+4) = 8$	$\leftarrow$

Si accetta la mossa indicata.

Iterazione 2 (move): schemi di carico $\{1,2,4\},\{3\},\{5\},\{6,7\}$, carichi $(10,3,8,8)$, $nb = 4$, $SQ = 237$.

mossa	carichi	guadagno
oggetto 1: contenitore $1 \to 2$	$(5,8,8,8)$	$2 \cdot 5 \cdot (3-10+5) = -20$
oggetto 2: contenitore $1 \to 2$	$(7,6,8,8)$	$2 \cdot 3 \cdot (3-10+3) = -24$
oggetto 4: contenitore $1 \to 2$	$(8,5,8,8)$	$2 \cdot 2 \cdot (3-10+2) = -20$
oggetto 4: contenitore $1 \to 3$	$(8,3,10,8)$	$2 \cdot 2 \cdot (8-10+2) = 0$
oggetto 4: contenitore $1 \to 4$	$(8,3,8,10)$	$2 \cdot 2 \cdot (8-10+2) = 0$
oggetto 6: contenitore $4 \to 2$	$(10,7,8,4)$	$2 \cdot 4 \cdot (3-8+4) = -8$
oggetto 7: contenitore $4 \to 2$	$(10,7,8,4)$	$2 \cdot 4 \cdot (3-8+4) = -8$

Nessuna mossa è migliorante.

Iterazione 3 (swap): schemi di carico $\{1,2,4\},\{3\},\{5\},\{6,7\}$, carichi $(10,3,8,8)$, $nb = 4$, $SQ = 237$.

swap	carichi	guadagno
oggetti $1,3$ (contenitori $1,2$)	$(8,5,8,8)$	$2 \cdot (2) \cdot (3-10+2) = -20$
oggetti $1,6$ (contenitori $1,4$)	$(9,3,8,9)$	$2 \cdot (1) \cdot (8-10+1) = -2$
oggetti $1,7$ (contenitori $1,4$)	$(9,3,8,9)$	$2 \cdot (1) \cdot (8-10+1) = -2$
oggetti $3,5$ (contenitori $2,3$)	$(10,8,3,8)$	$2 \cdot (-5) \cdot (8-3-5) = 0$
oggetti $3,6$ (contenitori $2,4$)	$(10,4,8,7)$	$2 \cdot (-1) \cdot (8-3-1) = -8$
oggetti $3,7$ (contenitori $2,4$)	$(10,4,8,7)$	$2 \cdot (-1) \cdot (8-3-1) = -8$

Nessuno swap è migliorante.

VND si ferma con schemi di carico $\{1,2,4\},\{3\},\{5\},\{6,7\}$, carichi $(10,3,8,8)$, $SQ = 237$ e $nb = 4$ contenitori.

3. Gli schemi di carico $\{1,6\},\{4,5\},\{2,3,7\}$, di carichi $(9,10,10)$, usano 3 contenitori: poiché $3 = \lceil 29/10 \rceil$, questo riempimento è **ottimo**, e l'ottimo locale no.

4. Nel riempimento ottimo gli oggetti 1 e 2 sono in contenitori diversi, mentre nell'ottimo locale sono insieme nel contenitore pieno $\{1, 2, 4\}$. Per separarli, un oggetto deve lasciare un contenitore pieno (carico 10) per uno più leggero: per la formula del guadagno $2\delta(L_b - L_a + \delta)$ una tale mossa è migliorante solo se $L_b + \delta > 10$, il che è impossibile. L'obiettivo ausiliario protegge i contenitori pieni, il che di solito è un bene, ma qui il contenitore pieno di FF è quello sbagliato: la ricerca è intrappolata in un ottimo locale.

3.3 TSP: 2-opt

Esercizio 3.7 — Ricerca locale TSP 1

6 città con le distanze simmetriche

d_{ij}	1	2	3	4	5	6
1	—	7	6	3	4	6
2	7	—	6	5	7	4
3	6	6	—	5	5	2
4	3	5	5	—	5	3
5	4	7	5	5	—	4
6	6	4	2	3	4	—

Un giro $\sigma = (\sigma_1, \ldots, \sigma_n)$ tiene la città 1 in prima posizione. La mossa 2-opt (i, j), per $i = 1, \ldots, n-2$ e $j = i+2, \ldots, n$ (tranne $(1, n)$), esaminata in quest'ordine, toglie i lati $\{\sigma_i, \sigma_{i+1}\}$, $\{\sigma_j, \sigma_{j+1}\}$ (con $\sigma_{n+1} = \sigma_1$), inverte $\sigma_{i+1}, \ldots, \sigma_j$, e cambia la lunghezza di $\Delta = d_{\sigma_i\sigma_j} + d_{\sigma_{i+1}\sigma_{j+1}} - d_{\sigma_i\sigma_{i+1}} - d_{\sigma_j\sigma_{j+1}}$. Best improvement risolve i pareggi a favore della mossa esaminata per prima. L'euristica nearest neighbour NN risolve i pareggi a favore della città di indice minimo.

Domande

1. Calcolare il giro di NN dalla città 1 e la sua lunghezza.
2. Calcolare Δ per tutte le mosse 2-opt del giro di NN, e verificarne una calcolando direttamente la lunghezza del nuovo giro.
3. Applicare la ricerca locale 2-opt con la regola best improvement, partendo dal giro di NN.
4. Il valore ottimo è $z(I) = 23$. Il giro finale è ottimo? Quante mosse sono state eseguite, rispetto al limite $L(\sigma^0) - z(I)$ sul numero di mosse?

Soluzione

1.
   - dalla città 1 la città non visitata più vicina è 4 ($d_{1,4} = 3$);
   - dalla città 4 la città non visitata più vicina è 6 ($d_{4,6} = 3$);
   - dalla città 6 la città non visitata più vicina è 3 ($d_{6,3} = 2$);
   - dalla città 3 la città non visitata più vicina è 5 ($d_{3,5} = 5$);
   - dalla città 5 la città non visitata più vicina è 2 ($d_{5,2} = 7$);
   - il giro torna alla città 1 ($d_{2,1} = 7$).

   Il giro è $\sigma = (1, 4, 6, 3, 5, 2)$, di lunghezza 27.

2. Le 9 mosse del giro di NN sono

(i, j)	lati tolti	Δ	nuovo giro	lunghezza
$(1, 3)$	$\{1, 4\}, \{6, 3\}$	$6 + 5 - 3 - 2 = +6$	$(1, 6, 4, 3, 5, 2)$	33
$(1, 4)$	$\{1, 4\}, \{3, 5\}$	$6 + 5 - 3 - 5 = +3$	$(1, 3, 6, 4, 5, 2)$	30
$(1, 5)$	$\{1, 4\}, \{5, 2\}$	$4 + 5 - 3 - 7 = -1$	$(1, 5, 3, 6, 4, 2)$	26
$(2, 4)$	$\{4, 6\}, \{3, 5\}$	$5 + 4 - 3 - 5 = +1$	$(1, 4, 3, 6, 5, 2)$	28
$(2, 5)$	$\{4, 6\}, \{5, 2\}$	$5 + 4 - 3 - 7 = -1$	$(1, 4, 5, 3, 6, 2)$	26
$(2, 6)$	$\{4, 6\}, \{2, 1\}$	$5 + 6 - 3 - 7 = +1$	$(1, 4, 2, 5, 3, 6)$	28
$(3, 5)$	$\{6, 3\}, \{5, 2\}$	$4 + 6 - 2 - 7 = +1$	$(1, 4, 6, 5, 3, 2)$	28
$(3, 6)$	$\{6, 3\}, \{2, 1\}$	$4 + 6 - 2 - 7 = +1$	$(1, 4, 6, 2, 5, 3)$	28
$(4, 6)$	$\{3, 5\}, \{2, 1\}$	$6 + 4 - 5 - 7 = -2$	$(1, 4, 6, 3, 2, 5)$	25

   Per esempio, la mossa $(1, 3)$ dà il giro $(1, 6, 4, 3, 5, 2)$, di lunghezza $6+3+5+5+7+7 = 33 = 27 + 6$.

3. Iterazione 1: $\sigma = (1, 4, 6, 3, 5, 2)$, lunghezza 27.

(i, j)	lati tolti	Δ	nuovo giro	lunghezza	
$(1, 3)$	$\{1, 4\}, \{6, 3\}$	$6 + 5 - 3 - 2 = +6$	$(1, 6, 4, 3, 5, 2)$	33	
$(1, 4)$	$\{1, 4\}, \{3, 5\}$	$6 + 5 - 3 - 5 = +3$	$(1, 3, 6, 4, 5, 2)$	30	
$(1, 5)$	$\{1, 4\}, \{5, 2\}$	$4 + 5 - 3 - 7 = -1$	$(1, 5, 3, 6, 4, 2)$	26	
$(2, 4)$	$\{4, 6\}, \{3, 5\}$	$5 + 4 - 3 - 5 = +1$	$(1, 4, 3, 6, 5, 2)$	28	
$(2, 5)$	$\{4, 6\}, \{5, 2\}$	$5 + 4 - 3 - 7 = -1$	$(1, 4, 5, 3, 6, 2)$	26	
$(2, 6)$	$\{4, 6\}, \{2, 1\}$	$5 + 6 - 3 - 7 = +1$	$(1, 4, 2, 5, 3, 6)$	28	
$(3, 5)$	$\{6, 3\}, \{5, 2\}$	$4 + 6 - 2 - 7 = +1$	$(1, 4, 6, 5, 3, 2)$	28	
$(3, 6)$	$\{6, 3\}, \{2, 1\}$	$4 + 6 - 2 - 7 = +1$	$(1, 4, 6, 2, 5, 3)$	28	
$(4, 6)$	$\{3, 5\}, \{2, 1\}$	$6 + 4 - 5 - 7 = -2$	$(1, 4, 6, 3, 2, 5)$	25	$\leftarrow$

   La mossa $(4, 6)$ è accettata dopo 9 valutazioni.

   Iterazione 2: $\sigma = (1, 4, 6, 3, 2, 5)$, lunghezza 25.

(i,j)	lati tolti	Δ	nuovo giro	lunghezza	
$(1,3)$	$\{1,4\},\{6,3\}$	$6+5-3-2=+6$	$(1,6,4,3,2,5)$	31	
$(1,4)$	$\{1,4\},\{3,2\}$	$6+5-3-6=+2$	$(1,3,6,4,2,5)$	27	
$(1,5)$	$\{1,4\},\{2,5\}$	$7+5-3-7=+2$	$(1,2,3,6,4,5)$	27	
$(2,4)$	$\{4,6\},\{3,2\}$	$5+4-3-6=0$	$(1,4,3,6,2,5)$	25	
$(2,5)$	$\{4,6\},\{2,5\}$	$5+4-3-7=-1$	$(1,4,2,3,6,5)$	24	$\leftarrow$
$(2,6)$	$\{4,6\},\{5,1\}$	$5+6-3-4=+4$	$(1,4,5,2,3,6)$	29	
$(3,5)$	$\{6,3\},\{2,5\}$	$4+5-2-7=0$	$(1,4,6,2,3,5)$	25	
$(3,6)$	$\{6,3\},\{5,1\}$	$4+6-2-4=+4$	$(1,4,6,5,2,3)$	29	
$(4,6)$	$\{3,2\},\{5,1\}$	$5+7-6-4=+2$	$(1,4,6,3,5,2)$	27	

La mossa $(2,5)$ è accettata dopo 9 valutazioni.

Iterazione 3: $\sigma=(1,4,2,3,6,5)$, lunghezza 24.

(i,j)	lati tolti	Δ	nuovo giro	lunghezza	
$(1,3)$	$\{1,4\},\{2,3\}$	$7+5-3-6=+3$	$(1,2,4,3,6,5)$	27	
$(1,4)$	$\{1,4\},\{3,6\}$	$6+3-3-2=+4$	$(1,3,2,4,6,5)$	28	
$(1,5)$	$\{1,4\},\{6,5\}$	$6+5-3-4=+4$	$(1,6,3,2,4,5)$	28	
$(2,4)$	$\{4,2\},\{3,6\}$	$5+4-5-2=+2$	$(1,4,3,2,6,5)$	26	
$(2,5)$	$\{4,2\},\{6,5\}$	$3+7-5-4=+1$	$(1,4,6,3,2,5)$	25	
$(2,6)$	$\{4,2\},\{5,1\}$	$5+7-5-4=+3$	$(1,4,5,6,3,2)$	27	
$(3,5)$	$\{2,3\},\{6,5\}$	$4+5-6-4=-1$	$(1,4,2,6,3,5)$	23	$\leftarrow$
$(3,6)$	$\{2,3\},\{5,1\}$	$7+6-6-4=+3$	$(1,4,2,5,6,3)$	27	
$(4,6)$	$\{3,6\},\{5,1\}$	$5+6-2-4=+5$	$(1,4,2,3,5,6)$	29	

La mossa $(3,5)$ è accettata dopo 9 valutazioni.

Iterazione 4: $\sigma=(1,4,2,6,3,5)$, lunghezza 23.

(i,j)	lati tolti	Δ	nuovo giro	lunghezza
$(1,3)$	$\{1,4\},\{2,6\}$	$7+3-3-4=+3$	$(1,2,4,6,3,5)$	26
$(1,4)$	$\{1,4\},\{6,3\}$	$6+5-3-2=+6$	$(1,6,2,4,3,5)$	29
$(1,5)$	$\{1,4\},\{3,5\}$	$6+5-3-5=+3$	$(1,3,6,2,4,5)$	26
$(2,4)$	$\{4,2\},\{6,3\}$	$3+6-5-2=+2$	$(1,4,6,2,3,5)$	25
$(2,5)$	$\{4,2\},\{3,5\}$	$5+7-5-5=+2$	$(1,4,3,6,2,5)$	25
$(2,6)$	$\{4,2\},\{5,1\}$	$5+7-5-4=+3$	$(1,4,5,3,6,2)$	26
$(3,5)$	$\{2,6\},\{3,5\}$	$6+4-4-5=+1$	$(1,4,2,3,6,5)$	24
$(3,6)$	$\{2,6\},\{5,1\}$	$7+6-4-4=+5$	$(1,4,2,5,3,6)$	28
$(4,6)$	$\{6,3\},\{5,1\}$	$4+6-2-4=+4$	$(1,4,2,6,5,3)$	27

Nessuna mossa ha $\Delta<0$: $\sigma=(1,4,2,6,3,5)$ è un **ottimo locale** 2-opt (9 valutazioni).

La ricerca esegue 3 mosse: $27\to25\to24\to23$.

4. Il giro finale ha lunghezza $23=z(I)$: è **ottimo**. La ricerca ha eseguito 3 mosse, mentre la proprietà di terminazione delle note ne consente al più $27-23=4$.

Esercizio 3.8 — Ricerca locale TSP 2

6 città con le distanze simmetriche

d_{ij}	1	2	3	4	5	6
1	—	8	5	6	5	6
2	8	—	6	7	7	6
3	5	6	—	4	8	2
4	6	7	4	—	6	5
5	5	7	8	6	—	7
6	6	6	2	5	7	—

Un giro $\sigma = (\sigma_1, \ldots, \sigma_n)$ tiene la città 1 in prima posizione. La mossa 2-opt (i, j), per $i = 1, \ldots, n-2$ e $j = i+2, \ldots, n$ (tranne $(1, n)$), esaminata in quest'ordine, toglie i lati $\{\sigma_i, \sigma_{i+1}\}$, $\{\sigma_j, \sigma_{j+1}\}$ (con $\sigma_{n+1} = \sigma_1$), inverte $\sigma_{i+1}, \ldots, \sigma_j$, e cambia la lunghezza di $\Delta = d_{\sigma_i\sigma_j} + d_{\sigma_{i+1}\sigma_{j+1}} - d_{\sigma_i\sigma_{i+1}} - d_{\sigma_j\sigma_{j+1}}$. Best improvement risolve i pareggi a favore della mossa esaminata per prima. L'euristica nearest neighbour NN risolve i pareggi a favore della città di indice minimo.

Domande

1. Calcolare il giro di NN dalla città 1 e la sua lunghezza.
2. Applicare la ricerca locale 2-opt con la regola **first improvement**, partendo dal giro di NN.
3. Applicare la ricerca locale 2-opt con la regola **best improvement**, partendo dal giro di NN.
4. Il valore ottimo è $z(I) = 30$. Confrontare i due ottimi locali e il numero di valutazioni delle due regole.

Soluzione

1.
   - dalla città 1 la città non visitata più vicina è 3 ($d_{1,3} = 5$);
   - dalla città 3 la città non visitata più vicina è 6 ($d_{3,6} = 2$);
   - dalla città 6 la città non visitata più vicina è 4 ($d_{6,4} = 5$);
   - dalla città 4 la città non visitata più vicina è 5 ($d_{4,5} = 6$);
   - dalla città 5 la città non visitata più vicina è 2 ($d_{5,2} = 7$);
   - il giro torna alla città 1 ($d_{2,1} = 8$).

   Il giro è $\sigma = (1, 3, 6, 4, 5, 2)$, di lunghezza 33.
2. Iterazione 1: $\sigma = (1, 3, 6, 4, 5, 2)$, lunghezza 33.

(i,j)	lati tolti	Δ	nuovo giro	lunghezza	
$(1,3)$	$\{1,3\},\{6,4\}$	$6+4-5-5=0$	$(1,6,3,4,5,2)$	33	
$(1,4)$	$\{1,3\},\{4,5\}$	$6+8-5-6=+3$	$(1,4,6,3,5,2)$	36	
$(1,5)$	$\{1,3\},\{5,2\}$	$5+6-5-7=-1$	$(1,5,4,6,3,2)$	32	$\leftarrow$
$(2,4)$	$\{3,6\},\{4,5\}$	$4+7-2-6=+3$	$(1,3,4,6,5,2)$	36	
$(2,5)$	$\{3,6\},\{5,2\}$	$8+6-2-7=+5$	$(1,3,5,4,6,2)$	38	
$(2,6)$	$\{3,6\},\{2,1\}$	$6+6-2-8=+2$	$(1,3,2,5,4,6)$	35	
$(3,5)$	$\{6,4\},\{5,2\}$	$7+7-5-7=+2$	$(1,3,6,5,4,2)$	35	
$(3,6)$	$\{6,4\},\{2,1\}$	$6+6-5-8=-1$	$(1,3,6,2,5,4)$	32	
$(4,6)$	$\{4,5\},\{2,1\}$	$7+5-6-8=-2$	$(1,3,6,4,2,5)$	31	

La mossa $(1,5)$ è accettata dopo 3 valutazioni.

Iterazione 2: $\sigma=(1,5,4,6,3,2)$, lunghezza 32.

(i,j)	lati tolti	Δ	nuovo giro	lunghezza	
$(1,3)$	$\{1,5\},\{4,6\}$	$6+7-5-5=+3$	$(1,4,5,6,3,2)$	35	
$(1,4)$	$\{1,5\},\{6,3\}$	$6+8-5-2=+7$	$(1,6,4,5,3,2)$	39	
$(1,5)$	$\{1,5\},\{3,2\}$	$5+7-5-6=+1$	$(1,3,6,4,5,2)$	33	
$(2,4)$	$\{5,4\},\{6,3\}$	$7+4-6-2=+3$	$(1,5,6,4,3,2)$	35	
$(2,5)$	$\{5,4\},\{3,2\}$	$8+7-6-6=+3$	$(1,5,3,6,4,2)$	35	
$(2,6)$	$\{5,4\},\{2,1\}$	$7+6-6-8=-1$	$(1,5,2,3,6,4)$	31	$\leftarrow$
$(3,5)$	$\{4,6\},\{3,2\}$	$4+6-5-6=-1$	$(1,5,4,3,6,2)$	31	
$(3,6)$	$\{4,6\},\{2,1\}$	$7+6-5-8=0$	$(1,5,4,2,3,6)$	32	
$(4,6)$	$\{6,3\},\{2,1\}$	$6+5-2-8=+1$	$(1,5,4,6,2,3)$	33	

La mossa $(2,6)$ è accettata dopo 6 valutazioni.

Iterazione 3: $\sigma=(1,5,2,3,6,4)$, lunghezza 31.

(i,j)	lati tolti	Δ	nuovo giro	lunghezza	
$(1,3)$	$\{1,5\},\{2,3\}$	$8+8-5-6=+5$	$(1,2,5,3,6,4)$	36	
$(1,4)$	$\{1,5\},\{3,6\}$	$5+7-5-2=+5$	$(1,3,2,5,6,4)$	36	
$(1,5)$	$\{1,5\},\{6,4\}$	$6+6-5-5=+2$	$(1,6,3,2,5,4)$	33	
$(2,4)$	$\{5,2\},\{3,6\}$	$8+6-7-2=+5$	$(1,5,3,2,6,4)$	36	
$(2,5)$	$\{5,2\},\{6,4\}$	$7+7-7-5=+2$	$(1,5,6,3,2,4)$	33	
$(2,6)$	$\{5,2\},\{4,1\}$	$6+8-7-6=+1$	$(1,5,4,6,3,2)$	32	
$(3,5)$	$\{2,3\},\{6,4\}$	$6+4-6-5=-1$	$(1,5,2,6,3,4)$	30	$\leftarrow$
$(3,6)$	$\{2,3\},\{4,1\}$	$7+5-6-6=0$	$(1,5,2,4,6,3)$	31	
$(4,6)$	$\{3,6\},\{4,1\}$	$4+6-2-6=+2$	$(1,5,2,3,4,6)$	33	

La mossa $(3,5)$ è accettata dopo 7 valutazioni.

Iterazione 4: $\sigma=(1,5,2,6,3,4)$, lunghezza 30.

(i,j)	lati tolti	Δ	nuovo giro	lunghezza
$(1,3)$	$\{1,5\},\{2,6\}$	$8+7-5-6=+4$	$(1,2,5,6,3,4)$	34
$(1,4)$	$\{1,5\},\{6,3\}$	$6+8-5-2=+7$	$(1,6,2,5,3,4)$	37
$(1,5)$	$\{1,5\},\{3,4\}$	$5+6-5-4=+2$	$(1,3,6,2,5,4)$	32
$(2,4)$	$\{5,2\},\{6,3\}$	$7+6-7-2=+4$	$(1,5,6,2,3,4)$	34
$(2,5)$	$\{5,2\},\{3,4\}$	$8+7-7-4=+4$	$(1,5,3,6,2,4)$	34
$(2,6)$	$\{5,2\},\{4,1\}$	$6+8-7-6=+1$	$(1,5,4,3,6,2)$	31
$(3,5)$	$\{2,6\},\{3,4\}$	$6+5-6-4=+1$	$(1,5,2,3,6,4)$	31
$(3,6)$	$\{2,6\},\{4,1\}$	$7+6-6-6=+1$	$(1,5,2,4,3,6)$	31
$(4,6)$	$\{6,3\},\{4,1\}$	$5+5-2-6=+2$	$(1,5,2,6,4,3)$	32

Nessuna mossa ha $\Delta < 0$: $\sigma = (1,5,2,6,3,4)$ è un **ottimo locale** 2-opt (9 valutazioni).

First improvement esegue 3 mosse, $33 \to 32 \to 31 \to 30$, con 25 valutazioni in totale.

3. Iterazione 1: $\sigma = (1,3,6,4,5,2)$, lunghezza 33.

(i,j)	lati tolti	Δ	nuovo giro	lunghezza	
$(1,3)$	$\{1,3\},\{6,4\}$	$6+4-5-5=0$	$(1,6,3,4,5,2)$	33	
$(1,4)$	$\{1,3\},\{4,5\}$	$6+8-5-6=+3$	$(1,4,6,3,5,2)$	36	
$(1,5)$	$\{1,3\},\{5,2\}$	$5+6-5-7=-1$	$(1,5,4,6,3,2)$	32	
$(2,4)$	$\{3,6\},\{4,5\}$	$4+7-2-6=+3$	$(1,3,4,6,5,2)$	36	
$(2,5)$	$\{3,6\},\{5,2\}$	$8+6-2-7=+5$	$(1,3,5,4,6,2)$	38	
$(2,6)$	$\{3,6\},\{2,1\}$	$6+6-2-8=+2$	$(1,3,2,5,4,6)$	35	
$(3,5)$	$\{6,4\},\{5,2\}$	$7+7-5-7=+2$	$(1,3,6,5,4,2)$	35	
$(3,6)$	$\{6,4\},\{2,1\}$	$6+6-5-8=-1$	$(1,3,6,2,5,4)$	32	
$(4,6)$	$\{4,5\},\{2,1\}$	$7+5-6-8=-2$	$(1,3,6,4,2,5)$	31	$\leftarrow$

La mossa $(4,6)$ è accettata dopo 9 valutazioni.

Iterazione 2: $\sigma = (1,3,6,4,2,5)$, lunghezza 31.

(i,j)	lati tolti	Δ	nuovo giro	lunghezza
$(1,3)$	$\{1,3\},\{6,4\}$	$6+4-5-5=0$	$(1,6,3,4,2,5)$	31
$(1,4)$	$\{1,3\},\{4,2\}$	$6+6-5-7=0$	$(1,4,6,3,2,5)$	31
$(1,5)$	$\{1,3\},\{2,5\}$	$8+8-5-7=+4$	$(1,2,4,6,3,5)$	35
$(2,4)$	$\{3,6\},\{4,2\}$	$4+6-2-7=+1$	$(1,3,4,6,2,5)$	32
$(2,5)$	$\{3,6\},\{2,5\}$	$6+7-2-7=+4$	$(1,3,2,4,6,5)$	35
$(2,6)$	$\{3,6\},\{5,1\}$	$8+6-2-5=+7$	$(1,3,5,2,4,6)$	38
$(3,5)$	$\{6,4\},\{2,5\}$	$6+6-5-7=0$	$(1,3,6,2,4,5)$	31
$(3,6)$	$\{6,4\},\{5,1\}$	$7+6-5-5=+3$	$(1,3,6,5,2,4)$	34
$(4,6)$	$\{4,2\},\{5,1\}$	$6+8-7-5=+2$	$(1,3,6,4,5,2)$	33

Nessuna mossa ha $\Delta < 0$: $\sigma = (1,3,6,4,2,5)$ è un **ottimo locale** 2-opt (9 valutazioni).

Best improvement esegue 1 mossa, $33 \to 31$, con 18 valutazioni in totale.

4. First improvement raggiunge un giro di lunghezza $30 = z(I)$, un giro **ottimo**, mentre best improvement si ferma in un ottimo locale di lunghezza 31, il 3.3% sopra l'ottimo. La prima mossa più ripida ($\Delta = -2$) porta in una valle peggiore: best improvement non è sempre migliore di first improvement. First improvement esamina 25 mosse in totale, best improvement 18: qui l'ottimo locale migliore costa più valutazioni, perché first improvement esegue più mosse.

Esercizio 3.9 — Ricerca locale TSP 3

6 città con le distanze simmetriche

d_{ij}	1	2	3	4	5	6
1	—	7	8	8	9	9
2	7	—	7	4	9	8
3	8	7	—	7	3	9
4	8	4	7	—	5	6
5	9	9	3	5	—	7
6	9	8	9	6	7	—

Un giro $\sigma = (\sigma_1, \ldots, \sigma_n)$ tiene la città 1 in prima posizione. La mossa 2-opt (i, j), per $i = 1, \ldots, n-2$ e $j = i+2, \ldots, n$ (tranne $(1, n)$), esaminata in quest'ordine, toglie i lati $\{\sigma_i, \sigma_{i+1}\}$, $\{\sigma_j, \sigma_{j+1}\}$ (con $\sigma_{n+1} = \sigma_1$), inverte $\sigma_{i+1}, \ldots, \sigma_j$, e cambia la lunghezza di $\Delta = d_{\sigma_i\sigma_j} + d_{\sigma_{i+1}\sigma_{j+1}} - d_{\sigma_i\sigma_{i+1}} - d_{\sigma_j\sigma_{j+1}}$. Best improvement risolve i pareggi a favore della mossa esaminata per prima. L'euristica nearest neighbour NN risolve i pareggi a favore della città di indice minimo. Una mossa Or-opt con $\ell = 1$ prende una città σ_i $(i \geq 2)$ e la inserisce fra due città consecutive σ_k, σ_{k+1} non adiacenti a essa $(k \neq i-1, i)$; con $p = \sigma_{i-1}$, $q = \sigma_{i+1}$, $s = \sigma_i$, $a = \sigma_k$, $b = \sigma_{k+1}$ cambia la lunghezza di $\Delta = d_{pq} + d_{as} + d_{sb} - d_{ps} - d_{sq} - d_{ab}$. Le mosse sono esaminate per $i = 2, \ldots, n$ e $k = 1, \ldots, n$.

Domande

1. Calcolare il giro di NN dalla città 1 e la sua lunghezza.
2. Mostrare che il giro di NN è un ottimo locale 2-opt.
3. Applicare un'iterazione di best improvement con le mosse Or-opt con $\ell = 1$ al giro di NN.
4. Il valore ottimo è $z(I) = 35$. Il nuovo giro è ottimo? È un ottimo locale 2-opt? Che cosa mostra questo sulla combinazione dei due intorni?

Soluzione

1.
   - dalla città 1 la città non visitata più vicina è 2 ($d_{1,2} = 7$);
   - dalla città 2 la città non visitata più vicina è 4 ($d_{2,4} = 4$);

- dalla città 4 la città non visitata più vicina è 5 ($d_{4,5} = 5$);
- dalla città 5 la città non visitata più vicina è 3 ($d_{5,3} = 3$);
- dalla città 3 la città non visitata più vicina è 6 ($d_{3,6} = 9$);
- il giro torna alla città 1 ($d_{6,1} = 9$).

Il giro è $\sigma = (1, 2, 4, 5, 3, 6)$, di lunghezza 37.

2. Le mosse 2-opt del giro di NN sono

(i, j)	lati tolti	Δ	nuovo giro	lunghezza
$(1, 3)$	$\{1, 2\}, \{4, 5\}$	$8 + 9 - 7 - 5 = +5$	$(1, 4, 2, 5, 3, 6)$	42
$(1, 4)$	$\{1, 2\}, \{5, 3\}$	$9 + 7 - 7 - 3 = +6$	$(1, 5, 4, 2, 3, 6)$	43
$(1, 5)$	$\{1, 2\}, \{3, 6\}$	$8 + 8 - 7 - 9 = 0$	$(1, 3, 5, 4, 2, 6)$	37
$(2, 4)$	$\{2, 4\}, \{5, 3\}$	$9 + 7 - 4 - 3 = +9$	$(1, 2, 5, 4, 3, 6)$	46
$(2, 5)$	$\{2, 4\}, \{3, 6\}$	$7 + 6 - 4 - 9 = 0$	$(1, 2, 3, 5, 4, 6)$	37
$(2, 6)$	$\{2, 4\}, \{6, 1\}$	$8 + 8 - 4 - 9 = +3$	$(1, 2, 6, 3, 5, 4)$	40
$(3, 5)$	$\{4, 5\}, \{3, 6\}$	$7 + 7 - 5 - 9 = 0$	$(1, 2, 4, 3, 5, 6)$	37
$(3, 6)$	$\{4, 5\}, \{6, 1\}$	$6 + 9 - 5 - 9 = +1$	$(1, 2, 4, 6, 3, 5)$	38
$(4, 6)$	$\{5, 3\}, \{6, 1\}$	$7 + 8 - 3 - 9 = +3$	$(1, 2, 4, 5, 6, 3)$	40

Tutte le mosse hanno $\Delta \geq 0$: il giro di NN è un **ottimo locale** 2-opt.

3. Le mosse Or-opt con $\ell = 1$ del giro di NN sono

città s	fra	Δ	lunghezza	
2	$4, 5$	$8 + 4 + 9 - 7 - 4 - 5 = +5$	42	
2	$5, 3$	$8 + 9 + 7 - 7 - 4 - 3 = +10$	47	
2	$3, 6$	$8 + 7 + 8 - 7 - 4 - 9 = +3$	40	
2	$6, 1$	$8 + 8 + 7 - 7 - 4 - 9 = +3$	40	
4	$1, 2$	$9 + 8 + 4 - 4 - 5 - 7 = +5$	42	
4	$5, 3$	$9 + 5 + 7 - 4 - 5 - 3 = +9$	46	
4	$3, 6$	$9 + 7 + 6 - 4 - 5 - 9 = +4$	41	
4	$6, 1$	$9 + 6 + 8 - 4 - 5 - 9 = +5$	42	
5	$1, 2$	$7 + 9 + 9 - 5 - 3 - 7 = +10$	47	
5	$2, 4$	$7 + 9 + 5 - 5 - 3 - 4 = +9$	46	
5	$3, 6$	$7 + 3 + 7 - 5 - 3 - 9 = 0$	37	
5	$6, 1$	$7 + 7 + 9 - 5 - 3 - 9 = +6$	43	
3	$1, 2$	$7 + 8 + 7 - 3 - 9 - 7 = +3$	40	
3	$2, 4$	$7 + 7 + 7 - 3 - 9 - 4 = +5$	42	
3	$4, 5$	$7 + 7 + 3 - 3 - 9 - 5 = 0$	37	
3	$6, 1$	$7 + 9 + 8 - 3 - 9 - 9 = +3$	40	
6	$1, 2$	$8 + 9 + 8 - 9 - 9 - 7 = 0$	37	
6	$2, 4$	$8 + 8 + 6 - 9 - 9 - 4 = 0$	37	
6	$4, 5$	$8 + 6 + 7 - 9 - 9 - 5 = -2$	35	$\leftarrow$
6	$5, 3$	$8 + 7 + 9 - 9 - 9 - 3 = +3$	40	

La mossa migliore inserisce la città 6 fra le città 4 e 5, con $\Delta = -2$: il nuovo giro è $(1, 2, 4, 6, 5, 3)$, di lunghezza 35.

4. Il nuovo giro ha lunghezza $35 = z(I)$: è **ottimo**, quindi anche un ottimo locale 2-opt. Il giro di NN è un ottimo locale per 2-opt ma non per Or-opt: una variable neighbourhood descent con $N_1 = $ 2-opt e $N_2 = $ Or-opt si fermerebbe solo in un giro che è un ottimo locale per entrambi, e qui trova l'ottimo.

3.4 $P\|C_{\max}$

Esercizio 3.10 — Ricerca locale $P\,||\,C_{\max}$ 1

$m = 3$ macchine identiche e 6 lavori con tempi di lavorazione

	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$
t_j	5	7	3	4	4	9

Una *move* riassegna un lavoro j a un'altra macchina h (esaminata per $j = 1, \dots, n$ e $h = 1, \dots, m$); uno *swap* scambia le macchine di due lavori $j < k$ su macchine diverse con $t_j \neq t_k$. Le ricerche locali usano la regola best improvement, con i pareggi risolti a favore della mossa esaminata per prima.

Domande

1. Calcolare lo schedule del list scheduling ($LIST$) con i lavori nell'ordine $1, 2, \dots, n$, e il suo makespan.
2. Applicare la ricerca locale move con il makespan come obiettivo, partendo dallo schedule di $LIST$.
3. Calcolare il lower bound $\max\{\max_j t_j, \lceil \sum_j t_j/m \rceil\}$.
4. Dimostrare che lo schedule finale è ottimo.

Soluzione

1.
   - il lavoro 1 ($t_1 = 5$) va sulla macchina 1, la meno carica (carico 0), che arriva a 5;
   - il lavoro 2 ($t_2 = 7$) va sulla macchina 2, la meno carica (carico 0), che arriva a 7;
   - il lavoro 3 ($t_3 = 3$) va sulla macchina 3, la meno carica (carico 0), che arriva a 3;
   - il lavoro 4 ($t_4 = 4$) va sulla macchina 3, la meno carica (carico 3), che arriva a 7;
   - il lavoro 5 ($t_5 = 4$) va sulla macchina 1, la meno carica (carico 5), che arriva a 9;
   - il lavoro 6 ($t_6 = 9$) va sulla macchina 2, la meno carica (carico 7), che arriva a 16;

   Lo schedule è $J_1 = \{1, 5\}, J_2 = \{2, 6\}, J_3 = \{3, 4\}$, con carichi $(9, 16, 7)$ e $C_{\max} = 16$.
2. Iterazione 1: $J_1 = \{1, 5\}, J_2 = \{2, 6\}, J_3 = \{3, 4\}$, carichi $(9, 16, 7)$, $C_{\max} = 16$.

mossa	carichi	$C_{\max}$	
lavoro 1: $M_1 \to M_2$	$(4, 21, 7)$	21	
lavoro 1: $M_1 \to M_3$	$(4, 16, 12)$	16	
lavoro 2: $M_2 \to M_1$	$(16, 9, 7)$	16	
lavoro 2: $M_2 \to M_3$	$(9, 9, 14)$	14	$\leftarrow$
lavoro 3: $M_3 \to M_1$	$(12, 16, 4)$	16	
lavoro 3: $M_3 \to M_2$	$(9, 19, 4)$	19	
lavoro 4: $M_3 \to M_1$	$(13, 16, 3)$	16	
lavoro 4: $M_3 \to M_2$	$(9, 20, 3)$	20	
lavoro 5: $M_1 \to M_2$	$(5, 20, 7)$	20	
lavoro 5: $M_1 \to M_3$	$(5, 16, 11)$	16	
lavoro 6: $M_2 \to M_1$	$(18, 7, 7)$	18	
lavoro 6: $M_2 \to M_3$	$(9, 7, 16)$	16	

Si accetta la mossa indicata.

Iterazione 2: $J_1 = \{1, 5\}, J_2 = \{6\}, J_3 = \{2, 3, 4\}$, carichi $(9, 9, 14)$, $C_{\max} = 14$.

mossa	carichi	$C_{\max}$	
lavoro 1: $M_1 \to M_2$	$(4, 14, 14)$	14	
lavoro 1: $M_1 \to M_3$	$(4, 9, 19)$	19	
lavoro 2: $M_3 \to M_1$	$(16, 9, 7)$	16	
lavoro 2: $M_3 \to M_2$	$(9, 16, 7)$	16	
lavoro 3: $M_3 \to M_1$	$(12, 9, 11)$	12	$\leftarrow$
lavoro 3: $M_3 \to M_2$	$(9, 12, 11)$	12	
lavoro 4: $M_3 \to M_1$	$(13, 9, 10)$	13	
lavoro 4: $M_3 \to M_2$	$(9, 13, 10)$	13	
lavoro 5: $M_1 \to M_2$	$(5, 13, 14)$	14	
lavoro 5: $M_1 \to M_3$	$(5, 9, 18)$	18	
lavoro 6: $M_2 \to M_1$	$(18, 0, 14)$	18	
lavoro 6: $M_2 \to M_3$	$(9, 0, 23)$	23	

Si accetta la mossa indicata.

Iterazione 3: $J_1 = \{1, 3, 5\}, J_2 = \{6\}, J_3 = \{2, 4\}$, carichi $(12, 9, 11)$, $C_{\max} = 12$.

mossa	carichi	$C_{\max}$
lavoro 1: $M_1 \to M_2$	$(7, 14, 11)$	14
lavoro 1: $M_1 \to M_3$	$(7, 9, 16)$	16
lavoro 2: $M_3 \to M_1$	$(19, 9, 4)$	19
lavoro 2: $M_3 \to M_2$	$(12, 16, 4)$	16
lavoro 3: $M_1 \to M_2$	$(9, 12, 11)$	12
lavoro 3: $M_1 \to M_3$	$(9, 9, 14)$	14
lavoro 4: $M_3 \to M_1$	$(16, 9, 7)$	16
lavoro 4: $M_3 \to M_2$	$(12, 13, 7)$	13
lavoro 5: $M_1 \to M_2$	$(8, 13, 11)$	13
lavoro 5: $M_1 \to M_3$	$(8, 9, 15)$	15
lavoro 6: $M_2 \to M_1$	$(21, 0, 11)$	21
lavoro 6: $M_2 \to M_3$	$(12, 0, 20)$	20

Nessuna mossa è migliorante: **ottimo locale**.

La ricerca esegue 2 mosse, $16 \to 14 \to 12$, e si ferma in $J_1 = \{1, 3, 5\}, J_2 = \{6\}, J_3 = \{2, 4\}$.

3. Il bound è $\max\{9, \lceil 32/3 \rceil\} = 11$.

4. Il bound 11 non è raggiunto, ma un makespan di 11 è impossibile. Il lavoro 6 ($t_6 = 9$) può condividere la sua macchina solo con lavori di tempo totale al più $11 - 9 = 2$, e ogni altro lavoro ha $t_j \geq 3$: la sua macchina contiene solo il lavoro 6. Gli altri 5 lavori, di tempo totale $32 - 9 = 23$, dovrebbero stare su due macchine di carico al più 11, cioè al più 22: impossibile. Quindi $z(I) \geq 12$, e lo schedule finale, con $C_{\max} = 12$, è **ottimo**.

Esercizio 3.11 — Ricerca locale $P||C_{\max}$ 2

$m = 3$ macchine identiche e 7 lavori con tempi di lavorazione

	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$	$j=7$
t_j	8	4	6	7	5	9	8

Una *move* riassegna un lavoro j a un'altra macchina h (esaminata per $j = 1, \ldots, n$ e $h = 1, \ldots, m$); uno *swap* scambia le macchine di due lavori $j < k$ su macchine diverse con $t_j \neq t_k$. Le ricerche locali usano la regola best improvement, con i pareggi risolti a favore della mossa esaminata per prima.

Domande

1. Calcolare lo schedule di LPT (pareggi risolti con l'indice minimo) e il suo makespan.

2. Mostrare che lo schedule di LPT è un ottimo locale per l'intorno move.

3. Applicare la ricerca locale swap con il makespan come obiettivo, partendo dallo schedule di LPT.
4. Calcolare il lower bound $\lceil \sum_j t_j/m \rceil$ e concludere. Che cosa mostra l'esercizio?

Soluzione

1. I lavori in ordine di tempo non crescente sono $(6, 1, 7, 4, 3, 5, 2)$.
   - il lavoro 6 ($t_6 = 9$) va sulla macchina 1, la meno carica (carico 0), che arriva a 9;
   - il lavoro 1 ($t_1 = 8$) va sulla macchina 2, la meno carica (carico 0), che arriva a 8;
   - il lavoro 7 ($t_7 = 8$) va sulla macchina 3, la meno carica (carico 0), che arriva a 8;
   - il lavoro 4 ($t_4 = 7$) va sulla macchina 2, la meno carica (carico 8), che arriva a 15;
   - il lavoro 3 ($t_3 = 6$) va sulla macchina 3, la meno carica (carico 8), che arriva a 14;
   - il lavoro 5 ($t_5 = 5$) va sulla macchina 1, la meno carica (carico 9), che arriva a 14;
   - il lavoro 2 ($t_2 = 4$) va sulla macchina 1, la meno carica (carico 14), che arriva a 18;

   Lo schedule è $J_1 = \{2, 5, 6\}, J_2 = \{1, 4\}, J_3 = \{3, 7\}$, con carichi $(18, 15, 14)$ e $C_{\max} = 18$.

2. Iterazione 1: $J_1 = \{2, 5, 6\}, J_2 = \{1, 4\}, J_3 = \{3, 7\}$, carichi $(18, 15, 14)$, $C_{\max} = 18$.

mossa	carichi	$C_{\max}$
lavoro 1: $M_2 \to M_1$	$(26, 7, 14)$	26
lavoro 1: $M_2 \to M_3$	$(18, 7, 22)$	22
lavoro 2: $M_1 \to M_2$	$(14, 19, 14)$	19
lavoro 2: $M_1 \to M_3$	$(14, 15, 18)$	18
lavoro 3: $M_3 \to M_1$	$(24, 15, 8)$	24
lavoro 3: $M_3 \to M_2$	$(18, 21, 8)$	21
lavoro 4: $M_2 \to M_1$	$(25, 8, 14)$	25
lavoro 4: $M_2 \to M_3$	$(18, 8, 21)$	21
lavoro 5: $M_1 \to M_2$	$(13, 20, 14)$	20
lavoro 5: $M_1 \to M_3$	$(13, 15, 19)$	19
lavoro 6: $M_1 \to M_2$	$(9, 24, 14)$	24
lavoro 6: $M_1 \to M_3$	$(9, 15, 23)$	23
lavoro 7: $M_3 \to M_1$	$(26, 15, 6)$	26
lavoro 7: $M_3 \to M_2$	$(18, 23, 6)$	23

   Nessuna mossa è migliorante: **ottimo locale**.

   Lo schedule di LPT è un **ottimo locale per move**, con $C_{\max} = 18$.

3. Iterazione 1: $J_1 = \{2, 5, 6\}, J_2 = \{1, 4\}, J_3 = \{3, 7\}$, carichi $(18, 15, 14)$, $C_{\max} = 18$.

swap	carichi	$C_{\max}$	
lavori $1,2$ (M_2, M_1)	$(22, 11, 14)$	22	
lavori $1,3$ (M_2, M_3)	$(18, 13, 16)$	18	
lavori $1,5$ (M_2, M_1)	$(21, 12, 14)$	21	
lavori $1,6$ (M_2, M_1)	$(17, 16, 14)$	17	$\leftarrow$
lavori $2,3$ (M_1, M_3)	$(20, 15, 12)$	20	
lavori $2,4$ (M_1, M_2)	$(21, 12, 14)$	21	
lavori $2,7$ (M_1, M_3)	$(22, 15, 10)$	22	
lavori $3,4$ (M_3, M_2)	$(18, 14, 15)$	18	
lavori $3,5$ (M_3, M_1)	$(19, 15, 13)$	19	
lavori $3,6$ (M_3, M_1)	$(15, 15, 17)$	17	
lavori $4,5$ (M_2, M_1)	$(20, 13, 14)$	20	
lavori $4,6$ (M_2, M_1)	$(16, 17, 14)$	17	
lavori $4,7$ (M_2, M_3)	$(18, 16, 13)$	18	
lavori $5,7$ (M_1, M_3)	$(21, 15, 11)$	21	
lavori $6,7$ (M_1, M_3)	$(17, 15, 15)$	17	

Si accetta lo swap indicato.

Iterazione 2: $J_1 = \{1, 2, 5\}$, $J_2 = \{4, 6\}$, $J_3 = \{3, 7\}$, carichi $(17, 16, 14)$, $C_{\max} = 17$.

swap	carichi	$C_{\max}$	
lavori $1,3$ (M_1, M_3)	$(15, 16, 16)$	16	$\leftarrow$
lavori $1,4$ (M_1, M_2)	$(16, 17, 14)$	17	
lavori $1,6$ (M_1, M_2)	$(18, 15, 14)$	18	
lavori $2,3$ (M_1, M_3)	$(19, 16, 12)$	19	
lavori $2,4$ (M_1, M_2)	$(20, 13, 14)$	20	
lavori $2,6$ (M_1, M_2)	$(22, 11, 14)$	22	
lavori $2,7$ (M_1, M_3)	$(21, 16, 10)$	21	
lavori $3,4$ (M_3, M_2)	$(17, 15, 15)$	17	
lavori $3,5$ (M_3, M_1)	$(18, 16, 13)$	18	
lavori $3,6$ (M_3, M_2)	$(17, 13, 17)$	17	
lavori $4,5$ (M_2, M_1)	$(19, 14, 14)$	19	
lavori $4,7$ (M_2, M_3)	$(17, 17, 13)$	17	
lavori $5,6$ (M_1, M_2)	$(21, 12, 14)$	21	
lavori $5,7$ (M_1, M_3)	$(20, 16, 11)$	20	
lavori $6,7$ (M_2, M_3)	$(17, 15, 15)$	17	

Si accetta lo swap indicato.

Iterazione 3: $J_1 = \{2, 3, 5\}$, $J_2 = \{4, 6\}$, $J_3 = \{1, 7\}$, carichi $(15, 16, 16)$, $C_{\max} = 16$.

swap	carichi	$C_{\max}$
lavori $1,2$ (M_3, M_1)	$(19, 16, 12)$	19
lavori $1,3$ (M_3, M_1)	$(17, 16, 14)$	17
lavori $1,4$ (M_3, M_2)	$(15, 17, 15)$	17
lavori $1,5$ (M_3, M_1)	$(18, 16, 13)$	18
lavori $1,6$ (M_3, M_2)	$(15, 15, 17)$	17
lavori $2,4$ (M_1, M_2)	$(18, 13, 16)$	18
lavori $2,6$ (M_1, M_2)	$(20, 11, 16)$	20
lavori $2,7$ (M_1, M_3)	$(19, 16, 12)$	19
lavori $3,4$ (M_1, M_2)	$(16, 15, 16)$	16
lavori $3,6$ (M_1, M_2)	$(18, 13, 16)$	18
lavori $3,7$ (M_1, M_3)	$(17, 16, 14)$	17
lavori $4,5$ (M_2, M_1)	$(17, 14, 16)$	17
lavori $4,7$ (M_2, M_3)	$(15, 17, 15)$	17
lavori $5,6$ (M_1, M_2)	$(19, 12, 16)$	19
lavori $5,7$ (M_1, M_3)	$(18, 16, 13)$	18
lavori $6,7$ (M_2, M_3)	$(15, 15, 17)$	17

Nessuno swap è migliorante: **ottimo locale**.

La ricerca locale swap esegue 2 mosse, $18 \to 17 \to 16$, e si ferma in $J_1 = \{2, 3, 5\}$, $J_2 = \{4, 6\}$, $J_3 = \{1, 7\}$.

4. Il bound è $\lceil 47/3 \rceil = 16$, raggiunto dallo schedule finale: è **ottimo**. Lo schedule di LPT è un ottimo locale per move e non per swap: l'ottimo locale **dipende dall'intorno**.

Esercizio 3.12 — Ricerca locale $P||C_{\max}$ 3

$m = 3$ macchine identiche e 6 lavori con tempi di lavorazione

	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$
t_j	5	5	7	9	2	7

Una *move* riassegna un lavoro j a un'altra macchina h (esaminata per $j = 1, \ldots, n$ e $h = 1, \ldots, m$); uno *swap* scambia le macchine di due lavori $j < k$ su macchine diverse con $t_j \neq t_k$. Le ricerche locali usano la regola best improvement, con i pareggi risolti a favore della mossa esaminata per prima.

Domande

1. Calcolare lo schedule del list scheduling ($LIST$) con i lavori nell'ordine $1, 2, \ldots, n$, e il suo makespan. Quante macchine sono critiche?

2. Applicare la ricerca locale move con il makespan come obiettivo, partendo dallo schedule di $LIST$. Perché si ferma?

3. Applicare la ricerca locale move con l'obiettivo ausiliario $\sum_h L_h^2$ (da minimizzare), partendo dallo schedule di $LIST$.

4. Calcolare il lower bound $\max\{\max_j t_j, \lceil \sum_j t_j/m \rceil\}$ e concludere.

Soluzione

1.
   - il lavoro 1 ($t_1 = 5$) va sulla macchina 1, la meno carica (carico 0), che arriva a 5;
   - il lavoro 2 ($t_2 = 5$) va sulla macchina 2, la meno carica (carico 0), che arriva a 5;
   - il lavoro 3 ($t_3 = 7$) va sulla macchina 3, la meno carica (carico 0), che arriva a 7;
   - il lavoro 4 ($t_4 = 9$) va sulla macchina 1, la meno carica (carico 5), che arriva a 14;
   - il lavoro 5 ($t_5 = 2$) va sulla macchina 2, la meno carica (carico 5), che arriva a 7;
   - il lavoro 6 ($t_6 = 7$) va sulla macchina 2, la meno carica (carico 7), che arriva a 14;

   Lo schedule è $J_1 = \{1, 4\}, J_2 = \{2, 5, 6\}, J_3 = \{3\}$, con carichi $(14, 14, 7)$ e $C_{\max} = 14$: 2 macchine sono critiche.

2. Iterazione 1: $J_1 = \{1, 4\}, J_2 = \{2, 5, 6\}, J_3 = \{3\}$, carichi $(14, 14, 7)$, $C_{\max} = 14$.

mossa	carichi	$C_{\max}$
lavoro 1: $M_1 \to M_2$	$(9, 19, 7)$	19
lavoro 1: $M_1 \to M_3$	$(9, 14, 12)$	14
lavoro 2: $M_2 \to M_1$	$(19, 9, 7)$	19
lavoro 2: $M_2 \to M_3$	$(14, 9, 12)$	14
lavoro 3: $M_3 \to M_1$	$(21, 14, 0)$	21
lavoro 3: $M_3 \to M_2$	$(14, 21, 0)$	21
lavoro 4: $M_1 \to M_2$	$(5, 23, 7)$	23
lavoro 4: $M_1 \to M_3$	$(5, 14, 16)$	16
lavoro 5: $M_2 \to M_1$	$(16, 12, 7)$	16
lavoro 5: $M_2 \to M_3$	$(14, 12, 9)$	14
lavoro 6: $M_2 \to M_1$	$(21, 7, 7)$	21
lavoro 6: $M_2 \to M_3$	$(14, 7, 14)$	14

   Nessuna mossa è migliorante: **ottimo locale**.

   Due macchine hanno carico 14. Una move cambia solo due carichi, e abbassa il makespan solo se parte dall'unica macchina critica: qui nessuna mossa può diminuire entrambi i carichi critici, quindi lo schedule di $LIST$ è un ottimo locale. Diverse mosse mantengono $C_{\max} = 14$: la ricerca è su un **plateau**.

3. Iterazione 1: $J_1 = \{1, 4\}, J_2 = \{2, 5, 6\}, J_3 = \{3\}$, carichi $(14, 14, 7)$, $C_{\max} = 14$, $\sum_h L_h^2 = 441$.

mossa	carichi	$C_{\max}$	$\sum_h L_h^2$	
lavoro 1: $M_1 \to M_2$	$(9, 19, 7)$	19	491	
lavoro 1: $M_1 \to M_3$	$(9, 14, 12)$	14	421	$\leftarrow$
lavoro 2: $M_2 \to M_1$	$(19, 9, 7)$	19	491	
lavoro 2: $M_2 \to M_3$	$(14, 9, 12)$	14	421	
lavoro 3: $M_3 \to M_1$	$(21, 14, 0)$	21	637	
lavoro 3: $M_3 \to M_2$	$(14, 21, 0)$	21	637	
lavoro 4: $M_1 \to M_2$	$(5, 23, 7)$	23	603	
lavoro 4: $M_1 \to M_3$	$(5, 14, 16)$	16	477	
lavoro 5: $M_2 \to M_1$	$(16, 12, 7)$	16	449	
lavoro 5: $M_2 \to M_3$	$(14, 12, 9)$	14	421	
lavoro 6: $M_2 \to M_1$	$(21, 7, 7)$	21	539	
lavoro 6: $M_2 \to M_3$	$(14, 7, 14)$	14	441	

Si accetta la mossa indicata.

Iterazione 2: $J_1 = \{4\}, J_2 = \{2, 5, 6\}, J_3 = \{1, 3\}$, carichi $(9, 14, 12)$, $C_{\max} = 14$, $\sum_h L_h^2 = 421$.

mossa	carichi	$C_{\max}$	$\sum_h L_h^2$	
lavoro 1: $M_3 \to M_1$	$(14, 14, 7)$	14	441	
lavoro 1: $M_3 \to M_2$	$(9, 19, 7)$	19	491	
lavoro 2: $M_2 \to M_1$	$(14, 9, 12)$	14	421	
lavoro 2: $M_2 \to M_3$	$(9, 9, 17)$	17	451	
lavoro 3: $M_3 \to M_1$	$(16, 14, 5)$	16	477	
lavoro 3: $M_3 \to M_2$	$(9, 21, 5)$	21	547	
lavoro 4: $M_1 \to M_2$	$(0, 23, 12)$	23	673	
lavoro 4: $M_1 \to M_3$	$(0, 14, 21)$	21	637	
lavoro 5: $M_2 \to M_1$	$(11, 12, 12)$	12	409	$\leftarrow$
lavoro 5: $M_2 \to M_3$	$(9, 12, 14)$	14	421	
lavoro 6: $M_2 \to M_1$	$(16, 7, 12)$	16	449	
lavoro 6: $M_2 \to M_3$	$(9, 7, 19)$	19	491	

Si accetta la mossa indicata.

Iterazione 3: $J_1 = \{4, 5\}, J_2 = \{2, 6\}, J_3 = \{1, 3\}$, carichi $(11, 12, 12)$, $C_{\max} = 12$, $\sum_h L_h^2 = 409$.

mossa	carichi	$C_{\max}$	$\sum_h L_h^2$
lavoro 1: $M_3 \to M_1$	$(16, 12, 7)$	16	449
lavoro 1: $M_3 \to M_2$	$(11, 17, 7)$	17	459
lavoro 2: $M_2 \to M_1$	$(16, 7, 12)$	16	449
lavoro 2: $M_2 \to M_3$	$(11, 7, 17)$	17	459
lavoro 3: $M_3 \to M_1$	$(18, 12, 5)$	18	493
lavoro 3: $M_3 \to M_2$	$(11, 19, 5)$	19	507
lavoro 4: $M_1 \to M_2$	$(2, 21, 12)$	21	589
lavoro 4: $M_1 \to M_3$	$(2, 12, 21)$	21	589
lavoro 5: $M_1 \to M_2$	$(9, 14, 12)$	14	421
lavoro 5: $M_1 \to M_3$	$(9, 12, 14)$	14	421
lavoro 6: $M_2 \to M_1$	$(18, 5, 12)$	18	493
lavoro 6: $M_2 \to M_3$	$(11, 5, 19)$	19	507

Nessuna mossa è migliorante: **ottimo locale**.

La ricerca esegue 2 mosse, con makespan $14 \to 14 \to 12$ e $\sum_h L_h^2$: $441 \to 421 \to 409$. La prima mossa non cambia il makespan, ma prepara la seconda.

4. Il bound è $\max\{9, \lceil 35/3 \rceil\} = 12$, raggiunto dallo schedule finale: è **ottimo**. L'obiettivo ausiliario dà una pendenza al plateau del makespan.

Capitolo 4

Metaeuristiche

4.1 Simulated annealing

Esercizio 4.1 — Simulated annealing 1

Un'istanza del TSP con 6 città e le distanze simmetriche

d_{ij}	1	2	3	4	5	6
1	—	5	1	7	7	4
2	5	—	4	5	2	1
3	1	4	—	6	6	4
4	7	5	6	—	7	6
5	7	2	6	7	—	2
6	4	1	4	6	2	—

Le mosse 2-opt di un giro di 6 città sono numerate

$$1:(1,3), 2:(1,4), 3:(1,5), 4:(2,4), 5:(2,5), 6:(2,6), 7:(3,5), 8:(3,6), 9:(4,6)$$

e i numeri casuali da usare, in questo ordine, sono 0.13, 0.76, 0.46, 0.53, 0.22, 0.05, 0.68, 0.68, 0.93, 0.38, 0.52, 0.83.

Domande

1. Calcolare il giro nearest neighbour dalla città 1 (pareggi: indice minimo) e la sua lunghezza, e mostrare che è un ottimo locale per l'intorno 2-opt.
2. Applicare il simulated annealing con l'intorno 2-opt a partire da questo giro, con $T_0 = 5$, $\alpha = 0.8$, $L = 2$ e $T_{\min} = 3$. Quante iterazioni vengono eseguite?
3. A ogni iterazione il primo numero casuale u sceglie la mossa $1 + \lfloor 9u \rfloor$ e il secondo, v, serve nel test di Metropolis. Mostrare ogni iterazione: mossa, δ, test di accettazione, giro corrente e giro migliore.
4. Confrontare il giro migliore trovato con quello ottimo, e spiegare quale mossa accettata ha permesso di uscire dall'ottimo locale di partenza.

Soluzione

1. Il giro nearest neighbour è $(1, 3, 2, 6, 5, 4)$, di lunghezza 22. I suoi 9 vicini hanno δ: $(1,3)$: $+7$, $(1,4)$: $+7$, $(1,5)$: $+5$, $(2,4)$: $+0$, $(2,5)$: $+0$, $(2,6)$: $+0$, $(3,5)$: $+0$, $(3,6)$: $+1$, $(4,6)$: $+4$. Nessun δ è negativo: il giro è un ottimo locale 2-opt.
2. Le temperature sono 5, 4 e 3.2; poi $T = 2.56 \leq 3$ ferma la ricerca: $3 \cdot 2 = 6$ iterazioni, che usano 12 numeri casuali.
3. - Iterazione 1 ($T = 5$): $u = 0.13$ dà la mossa $1 + \lfloor 9 \cdot 0.13 \rfloor = 2$, cioè $(1,4)$, con $\delta = 7$; $v = 0.76 \geq e^{-7/5} = 0.2466$: rifiutata. Giro corrente $(1, 3, 2, 6, 5, 4)$, lunghezza 22, migliore 22.
   - Iterazione 2 ($T = 5$): $u = 0.46$ dà la mossa $1 + \lfloor 9 \cdot 0.46 \rfloor = 5$, cioè $(2,5)$, con $\delta = 0$; $\delta \leq 0$: accettata. Giro corrente $(1, 3, 5, 6, 2, 4)$, lunghezza 22, migliore 22.

- Iterazione 3 ($T = 4$): $u = 0.22$ dà la mossa $1 + \lfloor 9 \cdot 0.22 \rfloor = 2$, cioè $(1, 4)$, con $\delta = 6$; $v = 0.05 < e^{-6/4} = 0.2231$: accettata. Giro corrente $(1, 6, 5, 3, 2, 4)$, lunghezza 28, migliore 22.
- Iterazione 4 ($T = 4$): $u = 0.68$ dà la mossa $1 + \lfloor 9 \cdot 0.68 \rfloor = 7$, cioè $(3, 5)$, con $\delta = -3$; $\delta \leq 0$: accettata. Giro corrente $(1, 6, 5, 2, 3, 4)$, lunghezza 25, migliore 22.
- Iterazione 5 ($T = 3.2$): $u = 0.93$ dà la mossa $1 + \lfloor 9 \cdot 0.93 \rfloor = 9$, cioè $(4, 6)$, con $\delta = -5$; $\delta \leq 0$: accettata. Giro corrente $(1, 6, 5, 2, 4, 3)$, lunghezza 20, migliore 20.
- Iterazione 6 ($T = 3.2$): $u = 0.52$ dà la mossa $1 + \lfloor 9 \cdot 0.52 \rfloor = 5$, cioè $(2, 5)$, con $\delta = 4$; $v = 0.83 \geq e^{-4/3.2} = 0.2865$: rifiutata. Giro corrente $(1, 6, 5, 2, 4, 3)$, lunghezza 20, migliore 20.

4. Il giro migliore trovato è $(1, 6, 5, 2, 4, 3)$ di lunghezza 20; enumerando i 60 giri, il giro ottimo è $(1, 3, 4, 2, 5, 6)$, di lunghezza 20: il simulated annealing trova l'ottimo. L'uscita è dovuta alle mosse accettate con $\delta \geq 0$ (iterazioni 2, 3), che portano la ricerca lontano dall'ottimo locale 22, da dove mosse miglioranti raggiungono 20.

Esercizio 4.2 — Simulated annealing 2

Uno zaino di capacità $c = 16$ e 5 oggetti:

	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$
p_j	11	5	18	8	8
w_j	5	3	11	6	7

Il simulated annealing con l'intorno flip massimizza

$$f_\beta(x) = \sum_j p_j x_j - \beta \max\left\{0, \sum_j w_j x_j - c\right\},$$

con $T_0 = 4$, $\alpha = 0.8$, $L = 2$, $T_{\min} = 1.7$. A ogni iterazione u sceglie l'oggetto $1 + \lfloor 5u \rfloor$ da invertire e v serve nel test di accettazione. I numeri casuali sono 0.63, 0.11, 0.17, 0.66, 0.38, 0.94, 0.58, 0.59, 0.69, 0.67, 0.39, 0.62, 0.69, 0.07, 0.59, 0.42.

Domande

1. Calcolare la soluzione x dell'algoritmo greedy per efficienza, e mostrare che è un ottimo locale per l'intorno flip fra le soluzioni ammissibili.
2. Applicare il simulated annealing da x con $\beta = 1$. Qual è la migliore soluzione ammissibile trovata, e dove si trova la ricerca alla fine?
3. Applicare il simulated annealing da x con $\beta = 3$, con gli stessi numeri casuali. Qual è la migliore soluzione ammissibile trovata?
4. Spiegare il diverso comportamento con i due valori di β, e confrontare con il valore ottimo.

Soluzione

1. Le efficienze sono $\frac{11}{5}$, $\frac{5}{3}$, $\frac{18}{11}$, $\frac{8}{6}$, $\frac{8}{7}$, già in ordine non crescente. Il greedy seleziona gli oggetti 1, 2 ($rc = 8$), salta l'oggetto 3 ($w_3 = 11$), seleziona l'oggetto 4 ($rc = 2$) e salta l'oggetto 5: $x = (1, 1, 0, 1, 0)$, peso 14, profitto 24. I flip ammissibili possono solo togliere oggetti (aggiungere l'oggetto 3 o 5 supera la capacità), quindi ogni vicino ammissibile ha profitto minore: x è un ottimo locale.

2. Con $\beta = 1$ le 8 iterazioni ($T = 4, 4, 3.2, 3.2, 2.56, 2.56, 2.048, 2.048$) sono:

   - Iterazione 1 ($T = 4$): $u = 0.63$ inverte l'oggetto 4: $y = (1, 1, 0, 0, 0)$, peso 8, profitto 16, $f(y) = 16$, $\delta = 8$; $v = 0.11 < e^{-8/4} = 0.1353$: accettata. Miglior profitto ammissibile: 24.
   - Iterazione 2 ($T = 4$): $u = 0.17$ inverte l'oggetto 1: $y = (0, 1, 0, 0, 0)$, peso 3, profitto 5, $f(y) = 5$, $\delta = 11$; $v = 0.66 \geq e^{-11/4} = 0.0639$: rifiutata. Miglior profitto ammissibile: 24.
   - Iterazione 3 ($T = 3.2$): $u = 0.38$ inverte l'oggetto 2: $y = (1, 0, 0, 0, 0)$, peso 5, profitto 11, $f(y) = 11$, $\delta = 5$; $v = 0.94 \geq e^{-5/3.2} = 0.2096$: rifiutata. Miglior profitto ammissibile: 24.
   - Iterazione 4 ($T = 3.2$): $u = 0.58$ inverte l'oggetto 3: $y = (1, 1, 1, 0, 0)$, peso 19, profitto 34, $f(y) = 31$, $\delta = -15$; $\delta \leq 0$: accettata. Miglior profitto ammissibile: 24.
   - Iterazione 5 ($T = 2.56$): $u = 0.69$ inverte l'oggetto 4: $y = (1, 1, 1, 1, 0)$, peso 25, profitto 42, $f(y) = 33$, $\delta = -2$; $\delta \leq 0$: accettata. Miglior profitto ammissibile: 24.
   - Iterazione 6 ($T = 2.56$): $u = 0.39$ inverte l'oggetto 2: $y = (1, 0, 1, 1, 0)$, peso 22, profitto 37, $f(y) = 31$, $\delta = 2$; $v = 0.62 \geq e^{-2/2.56} = 0.4578$: rifiutata. Miglior profitto ammissibile: 24.
   - Iterazione 7 ($T = 2.048$): $u = 0.69$ inverte l'oggetto 4: $y = (1, 1, 1, 0, 0)$, peso 19, profitto 34, $f(y) = 31$, $\delta = 2$; $v = 0.07 < e^{-2/2.048} = 0.3766$: accettata. Miglior profitto ammissibile: 24.
   - Iterazione 8 ($T = 2.048$): $u = 0.59$ inverte l'oggetto 3: $y = (1, 1, 0, 0, 0)$, peso 8, profitto 16, $f(y) = 16$, $\delta = 15$; $v = 0.42 \geq e^{-15/2.048} = 0.0007$: rifiutata. Miglior profitto ammissibile: 24.

   Il miglior profitto ammissibile resta 24, e alla fine la soluzione corrente è $(1, 1, 1, 0, 0)$, di peso $19 > 16$: la ricerca termina nella regione inammissibile.

3. Con $\beta = 3$:

   - Iterazione 1 ($T = 4$): $u = 0.63$ inverte l'oggetto 4: $y = (1, 1, 0, 0, 0)$, peso 8, profitto 16, $f(y) = 16$, $\delta = 8$; $v = 0.11 < e^{-8/4} = 0.1353$: accettata. Miglior profitto ammissibile: 24.
   - Iterazione 2 ($T = 4$): $u = 0.17$ inverte l'oggetto 1: $y = (0, 1, 0, 0, 0)$, peso 3, profitto 5, $f(y) = 5$, $\delta = 11$; $v = 0.66 \geq e^{-11/4} = 0.0639$: rifiutata. Miglior profitto ammissibile: 24.

- Iterazione 3 ($T = 3.2$): $u = 0.38$ inverte l'oggetto 2: $y = (1, 0, 0, 0, 0)$, peso 5, profitto 11, $f(y) = 11$, $\delta = 5$; $v = 0.94 \geq e^{-5/3.2} = 0.2096$: rifiutata. Miglior profitto ammissibile: 24.
- Iterazione 4 ($T = 3.2$): $u = 0.58$ inverte l'oggetto 3: $y = (1, 1, 1, 0, 0)$, peso 19, profitto 34, $f(y) = 25$, $\delta = -9$; $\delta \leq 0$: accettata. Miglior profitto ammissibile: 24.
- Iterazione 5 ($T = 2.56$): $u = 0.69$ inverte l'oggetto 4: $y = (1, 1, 1, 1, 0)$, peso 25, profitto 42, $f(y) = 15$, $\delta = 10$; $v = 0.67 \geq e^{-10/2.56} = 0.0201$: rifiutata. Miglior profitto ammissibile: 24.
- Iterazione 6 ($T = 2.56$): $u = 0.39$ inverte l'oggetto 2: $y = (1, 0, 1, 0, 0)$, peso 16, profitto 29, $f(y) = 29$, $\delta = -4$; $\delta \leq 0$: accettata. Miglior profitto ammissibile: 29.
- Iterazione 7 ($T = 2.048$): $u = 0.69$ inverte l'oggetto 4: $y = (1, 0, 1, 1, 0)$, peso 22, profitto 37, $f(y) = 19$, $\delta = 10$; $v = 0.07 \geq e^{-10/2.048} = 0.0076$: rifiutata. Miglior profitto ammissibile: 29.
- Iterazione 8 ($T = 2.048$): $u = 0.59$ inverte l'oggetto 3: $y = (1, 0, 0, 0, 0)$, peso 5, profitto 11, $f(y) = 11$, $\delta = 18$; $v = 0.42 \geq e^{-18/2.048} = 0.0002$: rifiutata. Miglior profitto ammissibile: 29.

La migliore soluzione ammissibile è $(1, 0, 1, 0, 0)$, di profitto 29.

4. Con $\beta = 1$ un peso in eccesso costa solo 1 per unità, e le soluzioni inammissibili che contengono l'oggetto 3 hanno un valore penalizzato grande (per esempio $f_1(1, 1, 1, 1, 0) = 42 - 9 = 33$): la ricerca resta nella regione inammissibile e non torna mai a una soluzione ammissibile migliore. Con $\beta = 3$ le soluzioni inammissibili sono solo un ponte: la ricerca le attraversa e raggiunge $\{1, 3\}$, di profitto 29, che è ottimo (per enumerazione). Il coefficiente di penalità deve essere abbastanza grande da rendere le soluzioni inammissibili meno attraenti di quelle ammissibili buone.

Esercizio 4.3 — Simulated annealing 3

Un'istanza del TSP con 6 città e le distanze simmetriche

d_{ij}	1	2	3	4	5	6
1	—	5	1	7	7	4
2	5	—	4	5	2	1
3	1	4	—	6	6	4
4	7	5	6	—	7	6
5	7	2	6	7	—	2
6	4	1	4	6	2	—

Il giro di partenza è il giro nearest neighbour $(1, 3, 2, 6, 5, 4)$, di lunghezza 22, un ottimo locale 2-opt. Le mosse 2-opt sono numerate

$$1 : (1, 3), 2 : (1, 4), 3 : (1, 5), 4 : (2, 4), 5 : (2, 5), 6 : (2, 6), 7 : (3, 5), 8 : (3, 6), 9 : (4, 6)$$

e i numeri casuali da usare sono 0.13, 0.76, 0.46, 0.53, 0.22, 0.05, 0.68, 0.68, 0.93, 0.38, 0.52, 0.83.

Domande

1. Calcolare la probabilità di accettazione $e^{-\delta/T}$ di un peggioramento $\delta = 6$ a $T = 5$ e a $T = 0.5$.
2. Con lo schema geometrico, quante temperature si usano con $T_0 = 0.5$, $\alpha = 0.8$, $T_{\min} = 0.3$? E con $T_0 = 5$, $\alpha = 0.95$, $T_{\min} = 0.5$?
3. Applicare il simulated annealing con $T_0 = 0.5$, $\alpha = 0.8$, $L = 2$, $T_{\min} = 0.3$ (mossa $1 + \lfloor 9u \rfloor$, test di Metropolis con v). Mostrare ogni iterazione.
4. Gli stessi numeri casuali con $T_0 = 5$, $T_{\min} = 3$ portano al giro ottimo di lunghezza 20. Spiegare perché l'esecuzione a bassa temperatura non ci arriva.

Soluzione

1. $e^{-6/5} = 0.3012$ e $e^{-6/0.5} = e^{-12} = 0.000006$: a bassa temperatura un peggioramento di 6 non è praticamente mai accettato.
2. Con $T_0 = 0.5$: 0.5, 0.4, 0.32 sono maggiori di 0.3, e 0.256 no: 3 temperature, 6 iterazioni con $L = 2$. Con $T_0 = 5$, $\alpha = 0.95$: il più piccolo K con $5 \cdot 0.95^K \leq 0.5$ è $K = \lceil \ln(0.1)/\ln(0.95) \rceil = 45$ temperature.
3. - Iterazione 1 ($T = 0.5$): $u = 0.13$ dà la mossa $1 + \lfloor 9 \cdot 0.13 \rfloor = 2$, cioè $(1, 4)$, con $\delta = 7$; $v = 0.76 \geq e^{-7/0.5} = 0.0000$: rifiutata. Giro corrente $(1, 3, 2, 6, 5, 4)$, lunghezza 22, migliore 22.
   - Iterazione 2 ($T = 0.5$): $u = 0.46$ dà la mossa $1 + \lfloor 9 \cdot 0.46 \rfloor = 5$, cioè $(2, 5)$, con $\delta = 0$; $\delta \leq 0$: accettata. Giro corrente $(1, 3, 5, 6, 2, 4)$, lunghezza 22, migliore 22.
   - Iterazione 3 ($T = 0.4$): $u = 0.22$ dà la mossa $1 + \lfloor 9 \cdot 0.22 \rfloor = 2$, cioè $(1, 4)$, con $\delta = 6$; $v = 0.05 \geq e^{-6/0.4} = 0.0000$: rifiutata. Giro corrente $(1, 3, 5, 6, 2, 4)$, lunghezza 22, migliore 22.
   - Iterazione 4 ($T = 0.4$): $u = 0.68$ dà la mossa $1 + \lfloor 9 \cdot 0.68 \rfloor = 7$, cioè $(3, 5)$, con $\delta = 1$; $v = 0.68 \geq e^{-1/0.4} = 0.0821$: rifiutata. Giro corrente $(1, 3, 5, 6, 2, 4)$, lunghezza 22, migliore 22.
   - Iterazione 5 ($T = 0.32$): $u = 0.93$ dà la mossa $1 + \lfloor 9 \cdot 0.93 \rfloor = 9$, cioè $(4, 6)$, con $\delta = 3$; $v = 0.38 \geq e^{-3/0.32} = 0.0001$: rifiutata. Giro corrente $(1, 3, 5, 6, 2, 4)$, lunghezza 22, migliore 22.
   - Iterazione 6 ($T = 0.32$): $u = 0.52$ dà la mossa $1 + \lfloor 9 \cdot 0.52 \rfloor = 5$, cioè $(2, 5)$, con $\delta = 0$; $\delta \leq 0$: accettata. Giro corrente $(1, 3, 2, 6, 5, 4)$, lunghezza 22, migliore 22.
4. A $T \leq 0.5$ le mosse peggioranti del giro di partenza ($\delta \geq 1$) sono accettate con probabilità al più $e^{-2} \approx 0.14$, e con questi numeri casuali sono tutte rifiutate: sono accettate solo le mosse con $\delta = 0$, e la ricerca resta su giri di lunghezza 22. Il simulated annealing a bassa temperatura è una ricerca locale: non può lasciare la valle dell'ottimo locale di partenza. Con $T_0 = 5$ la mossa peggiorante dell'iterazione 3 ($\delta = 6$) è accettata, e la ricerca raggiunge l'ottimo.

4.2 Tabu search

Esercizio 4.4 — Tabu search 1

$m = 2$ macchine identiche e 5 lavori con tempi di esecuzione $t = (3, 3, 2, 2, 2)$. La tabu search usa l'intorno move, confronta le soluzioni in modo lessicografico secondo $(C_{\max}, \sum_h L_h^2)$, vieta per *tenure* iterazioni di riportare un lavoro sulla macchina che ha appena lasciato (aspirazione: una mossa tabu è ammessa se dà un makespan minore del migliore), e si ferma quando $C_{\max}$ è uguale al lower bound. Pareggi fra mosse: lavoro di indice minimo, poi macchina di indice minimo.

Domande

1. Calcolare la soluzione LPT e il lower bound $LB = \max\{\max_j t_j, \lceil \sum_j t_j/m \rceil\}$.
2. Mostrare che la soluzione LPT è un ottimo locale per l'intorno move.
3. Applicare la tabu search con *tenure* $= 3$ dalla soluzione LPT, mostrando a ogni iterazione la valutazione dei vicini, la mossa eseguita e la lista tabu.
4. La soluzione finale è ottima? A quale iterazione la ricerca accetta una soluzione peggiore, e perché?

Soluzione

1. LPT assegna i lavori $1, 2$ (tempo 3) a M_1 e M_2, poi il lavoro 3 a M_1 (5), il lavoro 4 a M_2 (5), il lavoro 5 a M_1 (7): $J_1 = \{1, 3, 5\}$, $J_2 = \{2, 4\}$, carichi $(7, 5)$, $C_{\max} = 7$. $LB = \max\{3, \lceil 12/2 \rceil\} = 6$.
2. I vicini sono: lavoro $1 \to M_2$: $(8, 80)$, lavoro $2 \to M_1$: $(10, 104)$, lavoro $3 \to M_2$: $(7, 74)$, lavoro $4 \to M_1$: $(9, 90)$, lavoro $5 \to M_2$: $(7, 74)$. Nessun vicino ha makespan minore di 7, né makespan 7 e $\sum_h L_h^2 < 74$: la soluzione LPT è un ottimo locale.
3. - Iterazione 1. Vicini $(C_{\max}, \sum_h L_h^2)$: $1 \to M_2$: $(8, 80)$, $2 \to M_1$: $(10, 104)$, $3 \to M_2$: $(7, 74)$, $4 \to M_1$: $(9, 90)$, $5 \to M_2$: $(7, 74)$. Spostamento del lavoro 3 da M_1 a M_2: carichi $(5, 7)$, $C_{\max} = 7$; l'attributo $(3, M_1)$ è tabu fino all'iterazione 4; makespan migliore 7.
   - Iterazione 2. Vicini $(C_{\max}, \sum_h L_h^2)$: $1 \to M_2$: $(10, 104)$, $2 \to M_1$: $(8, 80)$, $3 \to M_1$: $(7, 74)$ tabu, $4 \to M_1$: $(7, 74)$, $5 \to M_2$: $(9, 90)$. Spostamento del lavoro 4 da M_2 a M_1: carichi $(7, 5)$, $C_{\max} = 7$; l'attributo $(4, M_2)$ è tabu fino all'iterazione 5; makespan migliore 7.
   - Iterazione 3. Vicini $(C_{\max}, \sum_h L_h^2)$: $1 \to M_2$: $(8, 80)$, $2 \to M_1$: $(10, 104)$, $3 \to M_1$: $(9, 90)$ tabu, $4 \to M_2$: $(7, 74)$ tabu, $5 \to M_2$: $(7, 74)$. Spostamento del lavoro 5 da M_1 a M_2: carichi $(5, 7)$, $C_{\max} = 7$; l'attributo $(5, M_1)$ è tabu fino all'iterazione 6; makespan migliore 7.
   - Iterazione 4. Vicini $(C_{\max}, \sum_h L_h^2)$: $1 \to M_2$: $(10, 104)$, $2 \to M_1$: $(8, 80)$, $3 \to M_1$: $(7, 74)$ tabu, $4 \to M_2$: $(9, 90)$ tabu, $5 \to M_1$: $(7, 74)$ tabu. Spostamento del

lavoro 2 da M_2 a M_1: carichi $(8, 4)$, $C_{\max} = 8$; l'attributo $(2, M_2)$ è tabu fino all'iterazione 7; makespan migliore 7.

- Iterazione 5. Vicini $(C_{\max}, \sum_h L_h^2)$: $1 \to M_2$: $(7, 74)$, $2 \to M_2$: $(7, 74)$ tabu, $3 \to M_1$: $(10, 104)$, $4 \to M_2$: $(6, 72)$ tabu, $5 \to M_1$: $(10, 104)$ tabu. Spostamento del lavoro 4 da M_1 a M_2: carichi $(6, 6)$, $C_{\max} = 6$; l'attributo $(4, M_1)$ è tabu fino all'iterazione 8; makespan migliore 6.

4. All'iterazione 5 il makespan raggiunge $LB = 6$: la soluzione $J_1 = \{1, 2\}$, $J_2 = \{3, 4, 5\}$ è ottima e la ricerca si ferma. La ricerca accetta soluzioni peggiori (nell'ordine lessicografico) all'iterazione 4, perché il miglior vicino non tabu è peggiore della soluzione corrente: è così che la tabu search lascia il plateau di makespan 7.

Esercizio 4.5 — Tabu search 2

$m = 2$ macchine identiche e 5 lavori con tempi di esecuzione $t = (3, 3, 2, 2, 2)$. La soluzione di partenza è la soluzione LPT $J_1 = \{1, 3, 5\}$, $J_2 = \{2, 4\}$. La tabu search usa l'intorno move e il confronto lessicografico secondo $(C_{\max}, \sum_h L_h^2)$; dopo aver spostato il lavoro j dalla macchina h, l'attributo (j, h) è tabu per *tenure* iterazioni. Pareggi: lavoro di indice minimo, poi macchina di indice minimo.

Domande

1. Applicare 4 iterazioni della tabu search con *tenure* $= 0$ (senza memoria). Che cosa succede?
2. Applicare 6 iterazioni con *tenure* $= 1$. Che cosa succede?
3. Spiegare perché liste tabu corte non impediscono alla ricerca di ciclare su questa istanza.
4. La stessa ricerca con *tenure* $= 3$ raggiunge l'ottimo 6 in 5 iterazioni. Qual è il ruolo della lista tabu, e che cosa può andare storto con una lista tabu troppo lunga?

Soluzione

1.
   - Iterazione 1: lavoro 3 da M_1 a M_2, carichi $(5, 7)$.
   - Iterazione 2: lavoro 3 da M_2 a M_1, carichi $(7, 5)$.
   - Iterazione 3: lavoro 3 da M_1 a M_2, carichi $(5, 7)$.
   - Iterazione 4: lavoro 3 da M_2 a M_1, carichi $(7, 5)$.

   La ricerca sposta il lavoro 3 avanti e indietro: cicla fra due soluzioni di makespan 7.

2.
   - Iterazione 1: lavoro 3 da M_1 a M_2, carichi $(5, 7)$.
   - Iterazione 2: lavoro 4 da M_2 a M_1, carichi $(7, 5)$.
   - Iterazione 3: lavoro 5 da M_1 a M_2, carichi $(5, 7)$.
   - Iterazione 4: lavoro 3 da M_2 a M_1, carichi $(7, 5)$.

- Iterazione 5: lavoro 4 da M_1 a M_2, carichi $(5, 7)$.
- Iterazione 6: lavoro 5 da M_2 a M_1, carichi $(7, 5)$.

La ricerca visita ancora solo soluzioni di makespan 7, e ripete la stessa sequenza di mosse: un ciclo più lungo.

3. Tutti i vicini con makespan 7 spostano un lavoro di tempo 2 fra le macchine. Con *tenure* ≤ 2 l'attributo appena creato scade prima che la ricerca si sia spostata abbastanza lontano, e la migliore mossa non tabu la riporta a una soluzione già visitata: il ciclo è più lungo della memoria.

4. La lista tabu vieta di annullare le mosse recenti, così che la ricerca è costretta a fare mosse peggioranti e a lasciare il plateau (con *tenure* $= 3$ le mosse peggioranti portano all'ottimo). Una lista tabu troppo lunga vieta troppe mosse, magari quelle buone; al limite, tutte le mosse sono tabu e la ricerca è guidata solo dal criterio di aspirazione. La durata tabu va calibrata.

Esercizio 4.6 — Tabu search 3

Un'istanza del TSP con 6 città e le distanze simmetriche

d_{ij}	1	2	3	4	5	6
1	—	5	1	7	7	4
2	5	—	4	5	2	1
3	1	4	—	6	6	4
4	7	5	6	—	7	6
5	7	2	6	7	—	2
6	4	1	4	6	2	—

Tabu search con l'intorno 2-opt: a ogni iterazione si valutano tutte le mosse (i, j), e si esegue la migliore mossa non tabu anche se non è migliorante (pareggi: prima mossa in ordine lessicografico). I due lati tolti da una mossa non possono essere rimessi per *tenure* $= 2$ iterazioni, a meno che il nuovo giro non sia più corto del migliore.

Domande

1. Calcolare il giro nearest neighbour σ dalla città 1 e mostrare che è un ottimo locale per 2-opt.

2. Eseguire la prima iterazione della tabu search da σ: quale mossa viene scelta, e quali lati diventano tabu?

3. Eseguire la seconda e la terza iterazione.

4. Confrontare il giro migliore trovato con l'ottimo locale della prima domanda.

Soluzione

1. $\sigma = (1, 3, 2, 6, 5, 4)$, di lunghezza 22; le variazioni di lunghezza dei suoi vicini sono $(1, 3)$: $+7$, $(1, 4)$: $+7$, $(1, 5)$: $+5$, $(2, 4)$: $+0$, $(2, 5)$: $+0$, $(2, 6)$: $+0$, $(3, 5)$: $+0$, $(3, 6)$: $+1$, $(4, 6)$: $+4$: nessuno è negativo, quindi σ è un ottimo locale 2-opt.

2. Da $(1, 3, 2, 6, 5, 4)$ i vicini hanno δ: $(1, 3)$: $+7$, $(1, 4)$: $+7$, $(1, 5)$: $+5$, $(2, 4)$: $+0$, $(2, 5)$: $+0$, $(2, 6)$: $+0$, $(3, 5)$: $+0$, $(3, 6)$: $+1$, $(4, 6)$: $+4$. La migliore mossa non tabu è $(2, 4)$ con $\delta = 0$: toglie i lati $(3, 2)$, $(6, 5)$, che diventano tabu fino all'iterazione 3; nuovo giro $(1, 3, 6, 2, 5, 4)$ di lunghezza 22. La ricerca si sposta anche se nessuna mossa migliora: lascia l'ottimo locale passando per un giro della stessa lunghezza.

3. - Da $(1, 3, 6, 2, 5, 4)$ i vicini hanno δ: $(1, 3)$: $+6$, $(1, 4)$: $+8$, $(1, 5)$: $+5$, $(2, 4)$: $+0$, $(2, 5)$: $+1$, $(2, 6)$: -1, $(3, 5)$: -1, $(3, 6)$: $+3$, $(4, 6)$: $+3$. La migliore mossa non tabu è $(2, 6)$ con $\delta = -1$: toglie i lati $(3, 6)$, $(4, 1)$, che diventano tabu fino all'iterazione 4; nuovo giro $(1, 3, 4, 5, 2, 6)$ di lunghezza 21.
   - Da $(1, 3, 4, 5, 2, 6)$ i vicini hanno δ: $(1, 3)$: $+5$, $(1, 4)$: $+8$, $(1, 5)$: $+7$, $(2, 4)$: $+3$, $(2, 5)$: $+3$, $(2, 6)$: $+1$, $(3, 5)$: -1, $(3, 6)$: $+2$, $(4, 6)$: $+1$. La migliore mossa non tabu è $(3, 5)$ con $\delta = -1$: toglie i lati $(4, 5)$, $(2, 6)$, che diventano tabu fino all'iterazione 5; nuovo giro $(1, 3, 4, 2, 5, 6)$ di lunghezza 20.

4. Dopo 3 iterazioni il giro migliore è $(1, 3, 4, 2, 5, 6)$, di lunghezza 20: è il giro ottimo (enumerando i 60 giri), migliore dell'ottimo locale 22 in cui si ferma la ricerca locale 2-opt.

4.3 Iterated local search e VNS

Esercizio 4.7 — Iterated local search e VNS 1

Un'istanza del TSP con 6 città e le distanze simmetriche

d_{ij}	1	2	3	4	5	6
1	—	5	1	7	7	4
2	5	—	4	5	2	1
3	1	4	—	6	6	4
4	7	5	6	—	7	6
5	7	2	6	7	—	2
6	4	1	4	6	2	—

L'ottimo locale 2-opt è il giro nearest neighbour $(1, 3, 2, 6, 5, 4)$ di lunghezza 22. L'iterated local search usa il kick double bridge con posizioni di taglio $p_1 = 1 + \lfloor u_1(n-3) \rfloor$, $p_2 = p_1 + 1 + \lfloor u_2(n-2-p_1) \rfloor$, $p_3 = p_2 + 1 + \lfloor u_3(n-1-p_2) \rfloor$ ($A = \sigma_1..\sigma_{p_1}$, $B = \sigma_{p_1+1}..\sigma_{p_2}$, $C = \sigma_{p_2+1}..\sigma_{p_3}$, $D = \sigma_{p_3+1}..\sigma_n$, nuovo giro $A\,C\,B\,D$), la ricerca locale 2-opt con best improvement, e accetta un nuovo ottimo locale se è più corto. I numeri casuali sono 0.13, 0.76, 0.46.

Domande

1. Calcolare le posizioni di taglio p_1, p_2, p_3 e i quattro segmenti del giro.
2. Costruire il giro perturbato, elencare i lati tolti e aggiunti, e calcolarne la lunghezza.
3. Applicare la ricerca locale 2-opt (best improvement) al giro perturbato.
4. Il nuovo ottimo locale viene accettato? Confrontarlo con il giro ottimo, di lunghezza 20.

Soluzione

1. $p_1 = 1 + \lfloor 0.13 \cdot 3 \rfloor = 1$, $p_2 = 1 + 1 + \lfloor 0.76 \cdot 3 \rfloor = 4$, $p_3 = 4 + 1 + \lfloor 0.46 \cdot 1 \rfloor = 5$. Segmenti $A = (1)$, $B = (3, 2, 6)$, $C = (5)$, $D = (4)$.
2. Il giro perturbato $A\,C\,B\,D$ è $(1, 5, 3, 2, 6, 4)$; toglie i lati $(4, 5)$, $(5, 6)$, $(1, 3)$ e aggiunge $(1, 5)$, $(4, 6)$, $(3, 5)$; la sua lunghezza è 31.
3. Da $(1, 5, 3, 2, 6, 4)$: la mossa migliore è $(1, 3)$ con $\delta = -8$, che dà $(1, 3, 5, 2, 6, 4)$ di lunghezza 23; la mossa migliore è $(2, 4)$ con $\delta = -1$, che dà $(1, 3, 2, 5, 6, 4)$ di lunghezza 22; la mossa migliore è $(2, 5)$ con $\delta = -1$, che dà $(1, 3, 6, 5, 2, 4)$ di lunghezza 21; la mossa migliore è $(2, 6)$ con $\delta = -1$, che dà $(1, 3, 4, 2, 5, 6)$ di lunghezza 20; poi nessuna mossa migliora il giro. Il nuovo ottimo locale è $(1, 3, 4, 2, 5, 6)$, di lunghezza 20.
4. Sì: $20 < 22$, il nuovo ottimo locale sostituisce quello corrente. La sua lunghezza è 20: è il giro ottimo. Un kick e una ricerca locale sono bastati a uscire dall'ottimo locale in cui si ferma la sola ricerca locale 2-opt.

Esercizio 4.8 — Iterated local search e VNS 2

Un'istanza del TSP con 6 città e le distanze simmetriche

d_{ij}	1	2	3	4	5	6
1	—	5	1	7	7	4
2	5	—	4	5	2	1
3	1	4	—	6	6	4
4	7	5	6	—	7	6
5	7	2	6	7	—	2
6	4	1	4	6	2	—

L'ottimo locale 2-opt è il giro nearest neighbour $(1, 3, 2, 6, 5, 4)$ di lunghezza 22. L'iterated local search usa il kick double bridge con posizioni di taglio $p_1 = 1 + \lfloor u_1(n-3) \rfloor$, $p_2 = p_1 + 1 + \lfloor u_2(n-2-p_1) \rfloor$, $p_3 = p_2 + 1 + \lfloor u_3(n-1-p_2) \rfloor$ ($A = \sigma_1..\sigma_{p_1}$, $B = \sigma_{p_1+1}..\sigma_{p_2}$, $C = \sigma_{p_2+1}..\sigma_{p_3}$, $D = \sigma_{p_3+1}..\sigma_n$, nuovo giro $A\,C\,B\,D$), la ricerca locale 2-opt con best improvement, e accetta un nuovo ottimo locale se è più corto. I numeri casuali sono 0.16, 0.13, 0.01.

Domande

1. Calcolare le posizioni di taglio p_1, p_2, p_3 e i quattro segmenti del giro.
2. Costruire il giro perturbato, elencare i lati tolti e aggiunti, e calcolarne la lunghezza.
3. Applicare la ricerca locale 2-opt (best improvement) al giro perturbato.
4. Il nuovo ottimo locale viene accettato? Che cosa succederebbe con il criterio di accettazione «accettare sempre»?

Soluzione

1. $p_1 = 1 + \lfloor 0.16 \cdot 3 \rfloor = 1$, $p_2 = 1 + 1 + \lfloor 0.13 \cdot 3 \rfloor = 2$, $p_3 = 2 + 1 + \lfloor 0.01 \cdot 3 \rfloor = 3$. Segmenti $A = (1)$, $B = (3)$, $C = (2)$, $D = (6, 5, 4)$.
2. Il giro perturbato $A\,C\,B\,D$ è $(1, 2, 3, 6, 5, 4)$; toglie i lati $(1, 3)$, $(2, 6)$ e aggiunge $(3, 6)$, $(1, 2)$; la sua lunghezza è 29.
3. Da $(1, 2, 3, 6, 5, 4)$: la mossa migliore è $(1, 3)$ con $\delta = -7$, che dà $(1, 3, 2, 6, 5, 4)$ di lunghezza 22; poi nessuna mossa migliora il giro. Il nuovo ottimo locale è $(1, 3, 2, 6, 5, 4)$, di lunghezza 22.
4. Il nuovo ottimo locale è il giro di partenza stesso, di lunghezza 22: non è più corto, e la soluzione corrente non cambia. Il kick era troppo debole: con segmenti B e C di una sola città cambia solo due lati, il che è una mossa 2-opt, e la ricerca locale 2-opt lo annulla subito. Con «accettare sempre» non cambierebbe niente comunque; il problema qui è l'intensità del kick, non il criterio di accettazione.

Esercizio 4.9 — Iterated local search e VNS 3

$m = 3$ macchine identiche e 7 lavori con tempi di esecuzione $t = (4, 4, 3, 3, 2, 2, 2)$. La variable neighbourhood search parte dalla soluzione LPT, usa $k_{\max} = 3$, esegue lo shaking con k spostamenti casuali (ogni spostamento: il lavoro $j = 1 + \lfloor 7u \rfloor$ va sulla g-esima macchina diversa dalla sua, $g = 1 + \lfloor 2u \rfloor$, con due numeri casuali consecutivi), usa come ricerca locale la discesa move con best improvement nell'ordine lessicografico $(C_{\max}, \sum_h L_h^2)$, e si ferma quando $C_{\max} = LB$. I numeri casuali sono 0.26, 0.51, 0.92, 0.07, 0.44, 0.09, 0.36, 0.36, 0.87, 0.77, 0.04, 0.66.

Domande

1. Calcolare la soluzione LPT e il lower bound LB, e mostrare che LPT è un ottimo locale per l'intorno move.
2. Spiegare perché lo swap di due lavori appartiene all'intorno N_2 usato dallo shaking.
3. Applicare la VNS, mostrando per ogni iterazione lo shaking, la ricerca locale e la decisione.
4. La soluzione finale è ottima? Quale valore di k è stato necessario?

Soluzione

1. LPT assegna i lavori per tempo non crescente alla macchina meno carica: carichi $(8, 6, 6)$, $J_1 = \{1, 5, 7\}$, $J_2 = \{2, 6\}$, $J_3 = \{3, 4\}$, $C_{\max} = 8$; $LB = \max\{4, \lceil 20/3 \rceil\} = 7$. Un lavoro spostato su una macchina di carico 6 la porta ad almeno 8, e un lavoro spostato su M_1 la porta sopra 8: nessuno spostamento diminuisce il makespan, e gli spostamenti con makespan 8 (per esempio il lavoro 5 da M_1 a M_2, carichi $(6, 8, 6)$) hanno $\sum_h L_h^2 \geq 136$, il valore di LPT. LPT è un ottimo locale.

2. Scambiare il lavoro j su M_a con il lavoro j' su M_b equivale a due spostamenti: j su M_b e j' su M_a. Quindi ogni vicino swap è in N_2, e lo shaking con $k = 2$ può produrlo.

3. - Iterazione 1, $k = 1$: shaking lavoro 2: $M_2 \to M_3$, carichi $(8, 2, 10)$; MoveDescent: lavoro 2 da M_3 a M_2, carichi $(8, 6, 6)$; risultato carichi $(8, 6, 6)$, non migliore $(k := k + 1)$.
   - Iterazione 2, $k = 2$: shaking lavoro 7: $M_1 \to M_2$; lavoro 4: $M_3 \to M_1$, carichi $(9, 8, 3)$; MoveDescent: lavoro 4 da M_1 a M_3, carichi $(6, 8, 6)$; risultato carichi $(6, 8, 6)$, non migliore $(k := k + 1)$.
   - Iterazione 3, $k = 3$: shaking lavoro 3: $M_3 \to M_1$; lavoro 7: $M_1 \to M_3$; lavoro 1: $M_1 \to M_3$, carichi $(5, 6, 9)$; MoveDescent: lavoro 7 da M_3 a M_1, carichi $(7, 6, 7)$; risultato carichi $(7, 6, 7)$, migliore, accettato $(k := 1)$.

4. Il makespan finale è $7 = LB$: la soluzione è ottima. Gli intorni N_1 e N_2 hanno fallito, ed è stato necessario N_3 (tre spostamenti casuali).

4.4 GRASP

Esercizio 4.10 — GRASP 1

Uno zaino di capacità $c = 16$ e 5 oggetti:

	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$
p_j	11	5	18	8	8
w_j	5	3	11	6	7

GRASP con $\alpha = 0.5$: la RCL contiene gli oggetti che entrano con efficienza almeno $e_{\max} - \alpha(e_{\max} - e_{\min})$, ordinati per efficienza non crescente, e si sceglie l'oggetto in posizione $1 + \lfloor u|RCL| \rfloor$; la ricerca locale usa flip e swap ammissibili con best improvement. I numeri casuali sono 0.39, 0.27, 0.38, 0.60, 0.66.

Domande

1. Calcolare la soluzione greedy per efficienza e la soluzione della ricerca locale avviata da essa.

2. Eseguire la prima iterazione della GRASP: costruzione e ricerca locale.

3. Eseguire la seconda iterazione.
4. Confrontare la migliore soluzione della GRASP con l'ottimo, e spiegare il ruolo della randomizzazione.

Soluzione

1. Il greedy dà $(1,1,0,1,0)$ di profitto 24 (oggetti $1,2,4$, peso 14); la ricerca locale non trova flip o swap miglioranti: si ferma a 24.
2. - Iterazione 1: $rc = 16$, candidati $\{1,2,3,4,5\}$, soglia 1.6714, $RCL = (1)$, $u = 0.39$ sceglie la posizione 1: oggetto 1; $rc = 11$, candidati $\{2,3,4,5\}$, soglia 1.4048, $RCL = (2,3)$, $u = 0.27$ sceglie la posizione 1: oggetto 2; $rc = 8$, candidati $\{4,5\}$, soglia 1.2381, $RCL = (4)$, $u = 0.38$ sceglie la posizione 1: oggetto 4. Costruzione: $(1,1,0,1,0)$, profitto 24. Ricerca locale: nessuna mossa migliorante. Risultato: profit 24.
3. - Iterazione 2: $rc = 16$, candidati $\{1,2,3,4,5\}$, soglia 1.6714, $RCL = (1)$, $u = 0.60$ sceglie la posizione 1: oggetto 1; $rc = 11$, candidati $\{2,3,4,5\}$, soglia 1.4048, $RCL = (2,3)$, $u = 0.66$ sceglie la posizione 2: oggetto 3. Costruzione: $(1,0,1,0,0)$, profitto 29. Ricerca locale: nessuna mossa migliorante. Risultato: profit 29.
4. La migliore soluzione è $(1,0,1,0,0)$ di profitto 29, che è ottima (per enumerazione: gli oggetti $\{1,3\}$ riempiono esattamente la capacità). L'algoritmo greedy deterministico non seleziona mai l'oggetto 3, perché dopo gli oggetti 1 e 2 non entra; la costruzione randomizzata sceglie talvolta l'oggetto 3 prima dell'oggetto 2, e raggiunge una valle diversa.

Esercizio 4.11 — GRASP 2

Un'istanza del TSP con 6 città e le distanze simmetriche

d_{ij}	1	2	3	4	5	6
1	—	5	1	7	7	4
2	5	—	4	5	2	1
3	1	4	—	6	6	4
4	7	5	6	—	7	6
5	7	2	6	7	—	2
6	4	1	4	6	2	—

GRASP per il TSP: nearest neighbour randomizzato dalla città 1 con $\alpha = 0.5$ (la RCL contiene le città non visitate j con $d_{\ell j} \le d_{\min} + \alpha(d_{\max} - d_{\min})$, ordinate per distanza, pareggi per indice; posizione $1 + \lfloor u|RCL| \rfloor$), seguito dalla ricerca locale 2-opt con best improvement. I numeri casuali sono 0.45, 0.31, 0.05, 0.86, 0.41, 0.52, 0.47, 0.47, 0.28, 0.22.

Domande

1. Eseguire la prima iterazione della GRASP.
2. Eseguire la seconda iterazione.
3. Qual è il giro migliore trovato? Confrontarlo con il giro nearest neighbour.
4. Che cosa darebbe la GRASP con $\alpha = 0$, per un numero qualsiasi di iterazioni?

Soluzione

1. Costruzione: da 1: soglia 4, $RCL = (3, 6)$, $u = 0.45$: città 3; da 3: soglia 5, $RCL = (2, 6)$, $u = 0.31$: città 2; da 2: soglia 3, $RCL = (6, 5)$, $u = 0.05$: città 6; da 6: soglia 4, $RCL = (5)$, $u = 0.86$: città 5; da 5: soglia 7, $RCL = (4)$, $u = 0.41$: città 4. Giro $(1, 3, 2, 6, 5, 4)$, lunghezza 22. Ricerca locale: nessuna mossa 2-opt migliora il giro: è già un ottimo locale 2-opt. Risultato: lunghezza 22.
2. Costruzione: da 1: soglia 4, $RCL = (3, 6)$, $u = 0.52$: città 6; da 6: soglia 3.5, $RCL = (2, 5)$, $u = 0.47$: città 2; da 2: soglia 3.5, $RCL = (5)$, $u = 0.47$: città 5; da 5: soglia 6.5, $RCL = (3)$, $u = 0.28$: città 3; da 3: soglia 6, $RCL = (4)$, $u = 0.22$: città 4. Giro $(1, 6, 2, 5, 3, 4)$, lunghezza 26. Ricerca locale: la mossa migliore è $(4, 6)$ con $\delta = -5$, che dà $(1, 6, 2, 5, 4, 3)$ di lunghezza 21; la mossa migliore è $(2, 4)$ con $\delta = -1$, che dà $(1, 6, 5, 2, 4, 3)$ di lunghezza 20; poi nessuna mossa migliora il giro. Risultato: lunghezza 20.
3. Il giro migliore è $(1, 6, 5, 2, 4, 3)$, di lunghezza 20, l'ottimo. Il giro nearest neighbour, $(1, 3, 2, 6, 5, 4)$ di lunghezza 22, è un ottimo locale 2-opt: la seconda costruzione, peggiore del nearest neighbour (26), porta la ricerca locale in una valle migliore.
4. Con $\alpha = 0$ la RCL contiene solo le città più vicine: la costruzione è l'euristica nearest neighbour (a meno di pareggi), ogni iterazione costruisce $(1, 3, 2, 6, 5, 4)$, e la ricerca locale si ferma a 22.

Esercizio 4.12 — GRASP 3

Uno zaino di capacità $c = 16$ e 5 oggetti:

	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$
p_j	11	5	18	8	8
w_j	5	3	11	6	7

GRASP con 2 iterazioni, RCL per soglia di efficienza $e_{\max} - \alpha(e_{\max} - e_{\min})$ (posizione $1 + \lfloor u|RCL| \rfloor$), ricerca locale con flip e swap ammissibili (best improvement). I numeri casuali sono 0.92, 0.29, 0.21, 0.73, 0.53, 0.33, 0.75, 0.76, 0.54, 0.68, 0.64, 0.82.

Domande

1. Quali oggetti formano la RCL al primo passo con $\alpha = 0$, $\alpha = 0.5$ e $\alpha = 1$?
2. Applicare la GRASP con $\alpha = 0$.

3. Applicare la GRASP con $\alpha = 1$, usando i numeri casuali dall'inizio.
4. Confrontare le due esecuzioni: quale diversifica, e quale intensifica?

Soluzione

1. $e_{\max} = \frac{11}{5} = 2.2$, $e_{\min} = \frac{8}{7} \approx 1.14$. Con $\alpha = 0$: $RCL = (1)$; con $\alpha = 0.5$ (soglia 1.6714): $RCL = (1)$; con $\alpha = 1$: tutti gli oggetti, $RCL = (1, 2, 3, 4, 5)$.

2. - Iterazione 1: $rc = 16$, candidati $\{1, 2, 3, 4, 5\}$, soglia 2.2000, $RCL = (1)$, $u = 0.92$ sceglie la posizione 1: oggetto 1; $rc = 11$, candidati $\{2, 3, 4, 5\}$, soglia 1.6667, $RCL = (2)$, $u = 0.29$ sceglie la posizione 1: oggetto 2; $rc = 8$, candidati $\{4, 5\}$, soglia 1.3333, $RCL = (4)$, $u = 0.21$ sceglie la posizione 1: oggetto 4. Costruzione: $(1, 1, 0, 1, 0)$, profitto 24. Ricerca locale: nessuna mossa migliorante. Risultato: profit 24.
   - Iterazione 2: $rc = 16$, candidati $\{1, 2, 3, 4, 5\}$, soglia 2.2000, $RCL = (1)$, $u = 0.73$ sceglie la posizione 1: oggetto 1; $rc = 11$, candidati $\{2, 3, 4, 5\}$, soglia 1.6667, $RCL = (2)$, $u = 0.53$ sceglie la posizione 1: oggetto 2; $rc = 8$, candidati $\{4, 5\}$, soglia 1.3333, $RCL = (4)$, $u = 0.33$ sceglie la posizione 1: oggetto 4. Costruzione: $(1, 1, 0, 1, 0)$, profitto 24. Ricerca locale: nessuna mossa migliorante. Risultato: profit 24.

   Con $\alpha = 0$ entrambe le iterazioni costruiscono la soluzione greedy e si fermano a 24: i numeri casuali sono irrilevanti (ogni RCL ha un solo elemento).

3. - Iterazione 1: $rc = 16$, candidati $\{1, 2, 3, 4, 5\}$, soglia 1.1429, $RCL = (1, 2, 3, 4, 5)$, $u = 0.92$ sceglie la posizione 5: oggetto 5; $rc = 9$, candidati $\{1, 2, 4\}$, soglia 1.3333, $RCL = (1, 2, 4)$, $u = 0.29$ sceglie la posizione 1: oggetto 1; $rc = 4$, candidati $\{2\}$, soglia 1.6667, $RCL = (2)$, $u = 0.21$ sceglie la posizione 1: oggetto 2. Costruzione: $(1, 1, 0, 0, 1)$, profitto 24. Ricerca locale: nessuna mossa migliorante. Risultato: profit 24.
   - Iterazione 2: $rc = 16$, candidati $\{1, 2, 3, 4, 5\}$, soglia 1.1429, $RCL = (1, 2, 3, 4, 5)$, $u = 0.73$ sceglie la posizione 4: oggetto 4; $rc = 10$, candidati $\{1, 2, 5\}$, soglia 1.1429, $RCL = (1, 2, 5)$, $u = 0.53$ sceglie la posizione 2: oggetto 2; $rc = 7$, candidati $\{1, 5\}$, soglia 1.1429, $RCL = (1, 5)$, $u = 0.33$ sceglie la posizione 1: oggetto 1. Costruzione: $(1, 1, 0, 1, 0)$, profitto 24. Ricerca locale: nessuna mossa migliorante. Risultato: profit 24.

   Profitto migliore: 24.

4. Con $\alpha = 0$ la GRASP è una sola ricerca locale dalla soluzione greedy: pura intensificazione, sempre 24. Con $\alpha = 1$ le costruzioni sono casuali e diverse (diversificazione), e la ricerca locale intensifica attorno a ciascuna di esse; qui il risultato migliore è 24, mentre l'ottimo è 29.

4.5 Large neighbourhood search

Esercizio 4.13 — Large neighbourhood search 1

Contenitori di capacità $c = 10$ e 8 oggetti con pesi $w = (2, 7, 5, 5, 3, 2, 6, 6)$. La LNS svuota il contenitore meno carico (pareggi: indice minimo) e un altro contenitore scelto a caso (il contenitore in posizione $1 + \lfloor u \cdot (nb - 1) \rfloor$ fra gli altri), reimpacchetta gli oggetti tolti con FFD in contenitori nuovi, conserva gli altri contenitori, e accetta il nuovo impacchettamento se usa meno contenitori, o altrettanti contenitori e una somma dei quadrati dei carichi non minore. Si ferma quando nb raggiunge $\lceil \sum_j w_j/c \rceil$. I numeri casuali sono 0.13, 0.76, 0.46.

Domande

1. Calcolare l'impacchettamento First Fit e il lower bound $\lceil \sum_j w_j/c \rceil$.
2. Eseguire la prima iterazione della LNS.
3. Proseguire finché la ricerca non si ferma.
4. L'impacchettamento finale è ottimo? Perché si accettano impacchettamenti con lo stesso numero di contenitori?

Soluzione

1. First Fit dà $P_1 = \{1, 2\}$, $P_2 = \{3, 4\}$, $P_3 = \{5, 6\}$, $P_4 = \{7\}$, $P_5 = \{8\}$: 5 contenitori. $\sum_j w_j = 36$, quindi servono almeno $\lceil 36/10 \rceil = 4$ contenitori.
2. - Iterazione 1: carichi $(9, 10, 5, 6, 6)$; contenitori svuotati P_3, P_1; oggetti tolti $\{1, 2, 5, 6\}$; reimpacchettamento FFD: $P_1 = \{3, 4\}$, $P_2 = \{7\}$, $P_3 = \{8\}$, $P_4 = \{2, 5\}$, $P_5 = \{1, 6\}$; $nb' = 5$, $\sum_b (\cdot)^2$: $278 \to 288$: accettato.
3. - Iterazione 2: carichi $(10, 6, 6, 10, 4)$; contenitori svuotati P_5, P_4; oggetti tolti $\{1, 2, 5, 6\}$; reimpacchettamento FFD: $P_1 = \{3, 4\}$, $P_2 = \{7\}$, $P_3 = \{8\}$, $P_4 = \{2, 5\}$, $P_5 = \{1, 6\}$; $nb' = 5$, $\sum_b (\cdot)^2$: $288 \to 288$: accettato.
   - Iterazione 3: carichi $(10, 6, 6, 10, 4)$; contenitori svuotati P_5, P_2; oggetti tolti $\{1, 6, 7\}$; reimpacchettamento FFD: $P_1 = \{3, 4\}$, $P_2 = \{8\}$, $P_3 = \{2, 5\}$, $P_4 = \{1, 6, 7\}$; $nb' = 4$, $\sum_b (\cdot)^2$: $288 \to 336$: accettato.
4. L'impacchettamento finale $P_1 = \{3, 4\}$, $P_2 = \{8\}$, $P_3 = \{2, 5\}$, $P_4 = \{1, 6, 7\}$ usa 4 contenitori, quanti il lower bound: è ottimo. Accettare impacchettamenti con lo stesso numero di contenitori e una somma dei quadrati dei carichi maggiore (o uguale) permette alla ricerca di muoversi sul plateau dell'obiettivo piatto, verso impacchettamenti con contenitori più pieni e un contenitore quasi vuoto, più facile da eliminare.

Esercizio 4.14 — Large neighbourhood search 2

Contenitori di capacità $c = 10$ e 8 oggetti con pesi $w = (3, 3, 6, 5, 4, 4, 2, 3)$. La LNS parte dall'impacchettamento FFD, svuota il contenitore meno carico e altri due contenitori a caso ($q = 3$; ciascuno è il contenitore in posizione $1 + \lfloor uK \rfloor$ fra i K contenitori non ancora scelti), e reimpacchetta gli oggetti tolti in contenitori nuovi, con FFD oppure in modo ottimo. I numeri casuali sono 0.66, 0.78.

Domande

1. Calcolare l'impacchettamento FFD e il lower bound.
2. Eseguire un'iterazione della LNS con la riparazione FFD.
3. Eseguire la stessa iterazione con la riparazione esatta: trovare il numero minimo di contenitori per gli oggetti tolti.
4. Confrontare le due riparazioni. La LNS con riparazione FFD può migliorare in questo modo un impacchettamento FFD?

Soluzione

1. FFD (pesi $6, 5, 4, 4, 3, 3, 3, 2$) dà $P_1 = \{3, 5\}$, $P_2 = \{4, 6\}$, $P_3 = \{1, 2, 8\}$, $P_4 = \{7\}$: 4 contenitori; $\lceil 30/10 \rceil = 3$.
2. - Iterazione 1: carichi $(10, 9, 9, 2)$; contenitori svuotati P_4, P_2, P_3; oggetti tolti $\{1, 2, 4, 6, 7, 8\}$; reimpacchettamento FFD: $P_1 = \{3, 5\}$, $P_2 = \{4, 6\}$, $P_3 = \{1, 2, 8\}$, $P_4 = \{7\}$; $nb' = 4$, $\sum_b (\cdot)^2$: $266 \to 266$: accettato.
3. Gli oggetti tolti $\{1, 2, 4, 6, 7, 8\}$ hanno peso totale $20 > 10$, quindi servono almeno 2 contenitori, e 2 contenitori bastano:
   - Iterazione 1: carichi $(10, 9, 9, 2)$; contenitori svuotati P_4, P_2, P_3; oggetti tolti $\{1, 2, 4, 6, 7, 8\}$; reimpacchettamento esatto: $P_1 = \{3, 5\}$, $P_2 = \{1, 4, 7\}$, $P_3 = \{2, 6, 8\}$; $nb' = 3$, $\sum_b (\cdot)^2$: $266 \to 300$: accettato.
4. La riparazione FFD reimpacchetta gli oggetti tolti come fa FFD, e trova per essi 3 contenitori; la riparazione esatta ne trova 2, e l'impacchettamento diventa ottimo (3 contenitori). Poiché gli oggetti tolti sono reimpacchettati con la stessa regola che ha costruito l'impacchettamento, la LNS con riparazione FFD tende a ricostruire gli stessi contenitori: con 200 iterazioni ($seed = 1$) non scende mai sotto 4. La potenza di un intorno grande dipende dal metodo che lo esplora.

Esercizio 4.15 — Large neighbourhood search 3

Un'istanza del TSP con 6 città e le distanze simmetriche

d_{ij}	1	2	3	4	5	6
1	—	5	1	7	7	4
2	5	—	4	5	2	1
3	1	4	—	6	6	4
4	7	5	6	—	7	6
5	7	2	6	7	—	2
6	4	1	4	6	2	—

La LNS parte dal giro nearest neighbour $(1, 3, 2, 6, 5, 4)$ (lunghezza 22), toglie $q = 2$ città a caso (la città in posizione $2 + \lfloor u(|\sigma| - 1) \rfloor$ del giro parziale corrente, così che la città 1 non venga mai tolta), le reinserisce nell'ordine di rimozione con cheapest insertion (pareggi: prima posizione), e accetta il nuovo giro se è più corto. I numeri casuali sono 0.68, 0.16, 0.84, 0.92.

Domande

1. Eseguire la prima iterazione (distruzione e riparazione) e decidere se il nuovo giro viene accettato.
2. Eseguire la seconda iterazione.
3. Calcolare tutti i costi di inserimento della prima città tolta nella seconda iterazione.
4. Il giro finale è ottimo? Quanti lati del giro di partenza sono stati cambiati?

Soluzione

1. Città tolte: $5, 3$; giro parziale $(1, 2, 6, 4)$; inserimenti: città 5 fra 2 e 6 (costo 3); città 3 fra 1 e 2 (costo 0); nuovo giro $(1, 3, 2, 5, 6, 4)$ di lunghezza 22: rifiutato.
2. Città tolte: $4, 5$; giro parziale $(1, 3, 2, 6)$; inserimenti: città 4 fra 3 e 2 (costo 7); città 5 fra 2 e 6 (costo 3); nuovo giro $(1, 3, 4, 2, 5, 6)$ di lunghezza 20: accettato.
3. Inserimento della città 4 in $(1, 3, 2, 6)$: $(1, 3)$: 12, $(3, 2)$: 7, $(2, 6)$: 10, $(6, 1)$: 9; il costo minimo è 7, fra le città 3 e 2.
4. Il giro finale $(1, 3, 4, 2, 5, 6)$ ha lunghezza 20: è il giro ottimo. Ha 2 lati in comune con il giro di partenza: un solo passo di distruzione e riparazione ha cambiato 4 lati, una mossa che 2-opt non può eseguire in un passo.

4.6 Algoritmi genetici

Esercizio 4.16 — Algoritmi genetici 1

Uno zaino di capacità $c = 16$ e 5 oggetti:

	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$
p_j	11	5	18	8	8
w_j	5	3	11	6	7

Algoritmo genetico steady-state con popolazione $pop[1] = (0,0,0,1,1)$, $pop[2] = (0,1,0,0,1)$, $pop[3] = (0,1,0,1,1)$, $pop[4] = (1,1,0,1,0)$; torneo binario (individui $1+\lfloor 4u \rfloor$, vince il più adatto, pareggi: il primo), crossover a un punto con taglio $1+\lfloor 4u \rfloor$ (geni $1..cut$ da a, gli altri da b), mutazione di ogni gene se $u < p_m = 0.2$, riparazione togliendo l'oggetto selezionato di efficienza minima finché la soluzione non è ammissibile; il figlio sostituisce l'individuo peggiore (pareggi: ultimo indice) se è migliore e non è già presente. Ogni iterazione usa 10 numeri casuali; sono 0.58, 0.07, 0.50, 0.39, 0.63, 0.56, 0.15, 0.15, 0.22, 0.60, 0.23, 0.97, 0.41, 0.64, 0.36, 0.05, 0.09, 0.60, 0.80, 0.01.

Domande

1. Calcolare la fitness della popolazione iniziale. Quali oggetti non vi compaiono mai?
2. Eseguire la prima iterazione dell'algoritmo genetico.
3. Eseguire la seconda iterazione.
4. Il solo crossover può mai produrre una soluzione che contiene l'oggetto 3? Quale operatore lo rende possibile?

Soluzione

1. Fitness: 16, 13, 21, 24 (pesi 13, 10, 16, 14). L'oggetto 3 non è in nessun individuo.
2. - Iterazione 1: il torneo $(3,1)$ seleziona $a = 3$, il torneo $(3,2)$ seleziona $b = 3$; taglio $= 3$: figlio $(0,1,0,1,1)$; mutazione dei geni $\{2,3\}$: $(0,0,1,1,1)$; la riparazione toglie gli oggetti $\{5,4\}$: $(0,0,1,0,0)$, profitto 18; individuo peggiore 2; inserito. Fitness della popolazione: $(16,18,21,24)$.
3. - Iterazione 2: il torneo $(1,4)$ seleziona $a = 4$, il torneo $(2,3)$ seleziona $b = 3$; taglio $= 2$: figlio $(1,1,0,1,1)$; mutazione dei geni $\{1,2,5\}$: $(0,0,0,1,0)$; ammissibile, profitto 8; individuo peggiore 1; non inserito. Fitness della popolazione: $(16,18,21,24)$.
4. No: il crossover copia geni dai genitori, e il gene 3 vale 0 in ogni individuo. Solo la mutazione può porlo a 1, come nell'iterazione 1; ma il figlio è inammissibile, e la riparazione toglie gli oggetti meno efficienti, non l'oggetto 3, quindi il risultato è

la soluzione $(0, 0, 1, 0, 0)$ di profitto 18. Il miglior individuo dopo due iterazioni ha profitto $24 < 29$.

Esercizio 4.17 — Algoritmi genetici 2

Uno zaino di capacità $c = 16$ e 5 oggetti:

	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$
p_j	11	5	18	8	8
w_j	5	3	11	6	7

Algoritmo genetico steady-state con popolazione $pop[1] = (0, 0, 0, 1, 1)$, $pop[2] = (0, 1, 0, 0, 1)$, $pop[3] = (0, 1, 0, 1, 1)$, $pop[4] = (1, 1, 0, 1, 0)$; torneo binario (individui $1 + \lfloor 4u \rfloor$, vince il più adatto, pareggi: il primo), crossover a un punto con taglio $1 + \lfloor 4u \rfloor$ (geni $1..cut$ da a, gli altri da b), mutazione di ogni gene se $u < p_m = 0.2$, riparazione togliendo l'oggetto selezionato di efficienza minima finché la soluzione non è ammissibile; il figlio sostituisce l'individuo peggiore (pareggi: ultimo indice) se è migliore e non è già presente. Ogni iterazione usa 10 numeri casuali; sono 0.58, 0.07, 0.50, 0.39, 0.63, 0.56, 0.15, 0.15, 0.22, 0.60, 0.23, 0.97, 0.41, 0.64, 0.36, 0.05, 0.09, 0.60, 0.80, 0.01.

Domande

1. Applicare la ricerca locale (flip e swap ammissibili, best improvement) a ogni individuo della popolazione iniziale, come fa l'algoritmo memetico.
2. Eseguire la prima iterazione dell'algoritmo memetico con gli stessi numeri casuali.
3. Eseguire la seconda iterazione.
4. Confrontare con l'algoritmo genetico senza ricerca locale, che dopo le stesse due iterazioni ha profitto migliore 24.

Soluzione

1. Dopo la ricerca locale la popolazione è $(1, 1, 0, 0, 1)$ (24), $(1, 1, 0, 0, 1)$ (24), $(1, 1, 0, 0, 1)$ (24), $(1, 1, 0, 1, 0)$ (24).
2. - Iterazione 1: il torneo $(3, 1)$ seleziona $a = 3$, il torneo $(3, 2)$ seleziona $b = 3$; taglio = 3: figlio $(1, 1, 0, 0, 1)$; mutazione dei geni $\{2, 3\}$: $(1, 0, 1, 0, 1)$; la riparazione toglie gli oggetti $\{5\}$: $(1, 0, 1, 0, 0)$; ricerca locale: $(1, 0, 1, 0, 0)$, profitto 29; individuo peggiore 4; inserito. Fitness della popolazione: $(24, 24, 24, 29)$.
3. - Iterazione 2: il torneo $(1, 4)$ seleziona $a = 4$, il torneo $(2, 3)$ seleziona $b = 2$; taglio = 2: figlio $(1, 0, 0, 0, 1)$; mutazione dei geni $\{1, 2, 5\}$: $(0, 1, 0, 0, 0)$; ammissibile; ricerca locale: $(1, 0, 1, 0, 0)$, profitto 29; individuo peggiore 3; non inserito. Fitness della popolazione: $(24, 24, 24, 29)$.

4. L'algoritmo memetico raggiunge l'ottimo 29 (oggetti $\{1, 3\}$) alla prima iterazione: la mutazione porta l'oggetto 3 nel figlio, e la ricerca locale trasforma il figlio riparato in un ottimo locale che contiene l'oggetto 3. Senza ricerca locale lo stesso figlio viene solo riparato, e il suo valore è scarso. La ricerca locale aggiunge intensificazione alla diversificazione di crossover e mutazione.

Capitolo 5

Euristiche MIP per i problemi classici

5.1 Arrotondamento e diving

Esercizio 5.1 — Arrotondamento e diving sul KP (KP1)

Consideriamo uno zaino di capacità $c = 16$ e un insieme $\mathscr{I} = \{1, 2, 3, 4, 5\}$ di 5 oggetti, con i seguenti profitti e pesi:

	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$
p_j	11	15	18	14	10
w_j	3	5	7	10	9

Domande

1. Verificare che gli oggetti sono ordinati per efficienza non crescente e calcolare la soluzione ottima $\tilde{x}$ del rilassamento lineare del modello del KP e il suo valore $z(\mathrm{LP}_{\mathrm{KP}})$.
2. Applicare l'arrotondamento: arrotondare $\tilde{x}$ al punto intero più vicino $\bar{x}$, verificare se $\bar{x}$ è ammissibile e indicarne il profitto.
3. Applicare il fractional diving: a ogni passo risolvere il rilassamento lineare con le variabili fissate fino a quel momento, fissare la variabile meno frazionaria al suo intero più vicino e ripetere finché la soluzione LP è intera.
4. Il diving è un cammino dell'albero di branch-and-bound. Calcolare il valore del rilassamento lineare dei rami che il diving non visita e usarli per stabilire se la soluzione del diving è ottima.

Soluzione

1. Le efficienze sono

$$\frac{11}{3} \approx 3.67, \quad \frac{15}{5} = 3, \quad \frac{18}{7} \approx 2.57, \quad \frac{14}{10} = 1.4, \quad \frac{10}{9} \approx 1.11,$$

non crescenti. Il LP prende gli oggetti $1, 2, 3$ (peso $3 + 5 + 7 = 15$), e la capacità residua 1 è riempita dalla frazione $\frac{1}{10}$ dell'oggetto 4:

$$\tilde{x} = \Big(1, 1, 1, \frac{1}{10}, 0\Big), \qquad z(\mathrm{LP_{KP}}) = 11 + 15 + 18 + \frac{14}{10} = \frac{227}{5} = 45.4.$$

2. $\left[\frac{1}{10}\right] = 0$, quindi $\bar{x} = (1, 1, 1, 0, 0)$: peso $15 \leq 16$, ammissibile, profitto 44.

3. **Passo 1.** L'unica variabile frazionaria è $x_4 = \frac{1}{10}$: fissare $x_4 := 0$.

   Passo 2. Senza l'oggetto 4, il LP prende gli oggetti $1, 2, 3$ e la frazione $\frac{1}{9}$ dell'oggetto 5: $\tilde{x} = \big(1, 1, 1, 0, \frac{1}{9}\big)$, di valore $44 + \frac{10}{9} = \frac{406}{9} \approx 45.11$. Fissare $x_5 := \left[\frac{1}{9}\right] = 0$.

   Passo 3. La soluzione LP $(1, 1, 1, 0, 0)$, di valore 44, è intera: il diving restituisce $S = \{1, 2, 3\}$, di profitto 44.

4. Il diving non visita i rami $x_4 = 1$ e $(x_4 = 0,\ x_5 = 1)$.

   - $x_4 = 1$: l'oggetto 4 usa 10 unità di capacità, profitto 14; il LP aggiunge l'oggetto 1 (residuo 3) e la frazione $\frac{3}{5}$ dell'oggetto 2: valore $14 + 11 + 9 = 34$.
   - $x_4 = 0,\ x_5 = 1$: l'oggetto 5 usa 9, profitto 10; il LP aggiunge l'oggetto 1 (residuo 4) e la frazione $\frac{4}{5}$ dell'oggetto 2: valore $10 + 11 + 12 = 33$.

   Entrambi i valori sono minori di 44: nessuna soluzione nei rami abbandonati è migliore della soluzione del diving, che è quindi **ottima**, $z(I) = 44$. Il diving, insieme ai bound dei rami che abbandona, è un branch-and-bound completo.

Esercizio 5.2 — Arrotondamento e diving sul KP (KP2)

Consideriamo uno zaino di capacità $c = 15$ e un insieme $\mathscr{I} = \{1, 2, 3, 4, 5\}$ di 5 oggetti, con i seguenti profitti e pesi:

	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$
p_j	13	12	14	6	5
w_j	2	5	10	6	7

Domande

1. Calcolare la soluzione ottima $\tilde{x}$ del rilassamento lineare e il suo valore $z(\mathrm{LP_{KP}})$ (gli oggetti sono ordinati per efficienza non crescente).
2. Arrotondare $\tilde{x}$ al punto intero più vicino e stabilire se è ammissibile. Arrotondarla poi per difetto e indicarne il profitto.
3. Applicare il fractional diving, passo per passo, e indicare il profitto della soluzione che restituisce.

4. Ripetere il diving partendo dalla decisione opposta al primo passo, e proseguire con la regola del fractional diving. Usare poi i bound LP dei rami per dimostrare che la migliore delle due soluzioni è ottima.

Soluzione

1. Le efficienze sono 6.5, 2.4, 1.4, 1, $\frac{5}{7} \approx 0.71$. Il LP prende gli oggetti $1, 2$ (peso 7) e la frazione $\frac{8}{10} = \frac{4}{5}$ dell'oggetto 3:

$$\tilde{x} = \left(1, 1, \frac{4}{5}, 0, 0\right), \qquad z(\mathrm{LP_{KP}}) = 13 + 12 + 14 \cdot \frac{4}{5} = \frac{181}{5} = 36.2.$$

2. $\left\lfloor \frac{4}{5} \right\rfloor = 1$ dà $(1, 1, 1, 0, 0)$, di peso $17 > 15$: **inammissibile**. Arrotondando per difetto si ottiene $(1, 1, 0, 0, 0)$, ammissibile (per l'osservazione sull'arrotondamento per difetto del Capitolo 10 «Euristiche MIP» del modulo *Programmazione lineare intera*), di profitto 25.

3. **Passo 1.** Fissare $x_3 := \left\lfloor \frac{4}{5} \right\rfloor = 1$. L'oggetto 3 usa 10; il LP prende l'oggetto 1 (residuo 3) e la frazione $\frac{3}{5}$ dell'oggetto 2: $\tilde{x} = \left(1, \frac{3}{5}, 1, 0, 0\right)$, di valore $14 + 13 + \frac{36}{5} = \frac{171}{5} = 34.2$.

   Passo 2. Fissare $x_2 := \left\lfloor \frac{3}{5} \right\rfloor = 1$. Gli oggetti $2, 3$ riempiono lo zaino $(5 + 10 = 15)$: la soluzione LP è $(0, 1, 1, 0, 0)$, intera, di valore 26. Il diving restituisce $S = \{2, 3\}$, di profitto 26.

4. **Passo 1'.** Fissare $x_3 := 0$. Il LP prende gli oggetti $1, 2, 4$ (peso 13) e la frazione $\frac{2}{7}$ dell'oggetto 5: valore $13 + 12 + 6 + \frac{10}{7} = \frac{227}{7} \approx 32.43$.

   Passo 2'. Fissare $x_5 := \left\lfloor \frac{2}{7} \right\rfloor = 0$: la soluzione LP $(1, 1, 0, 1, 0)$ è intera, di profitto 31.

   Le quattro foglie dell'albero sono:

ramo	valore LP	
$x_3 = 1$, $x_2 = 1$	26	intera
$x_3 = 1$, $x_2 = 0$	30	oggetti 3, 1 e metà dell'oggetto 4; $30 < 31$
$x_3 = 0$, $x_5 = 0$	31	intera
$x_3 = 0$, $x_5 = 1$	31	non migliore di 31

   Nessun ramo può contenere una soluzione di profitto maggiore di 31, quindi $\{1, 2, 4\}$ è **ottima**: $z(I) = 31$. Il primo diving (26) è fallito perché la sua prima decisione, la più naturale per il LP ($x_3 = \frac{4}{5} \to 1$), era sbagliata.

5.2 RINS, local branching e proximity search

Esercizio 5.3 — RINS, local branching e proximity search sul KP (KP3)

Consideriamo uno zaino di capacità $c = 22$ e un insieme $\mathscr{I} = \{1, 2, \ldots, 6\}$ di 6 oggetti, ordinati per efficienza non crescente, con i seguenti profitti e pesi:

	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$
p_j	18	21	4	19	10	7
w_j	8	10	2	12	7	10

La soluzione incumbent è la soluzione del greedy per *profitto* GR_p.

Domande

1. Calcolare la soluzione incumbent $\bar{x}$ e il suo profitto, e la soluzione ottima $\tilde{x}$ del rilassamento lineare $\mathrm{LP_{KP}}$ con il suo valore.
2. Applicare RINS: elencare le variabili fissate da RINS, scrivere il sotto-MIP con il vincolo di cutoff $\sum_j p_j x_j \geq f(\bar{x}) + 1$ e risolverlo per enumerazione.
3. Mostrare che il local branching con raggio $k = 2$ intorno a $\bar{x}$ non trova soluzioni migliori.
4. Applicare la proximity search con $\theta = 1$ a partire da $\bar{x}$ (la soluzione ottima di questa istanza ha profitto 43, come si verifica enumerando i 2^6 sottoinsiemi). Confrontare i tre metodi.

Soluzione

1. Per profitto non crescente l'ordine è 2 (21), 4 (19), 1 (18), 5, 6, 3. L'oggetto 2 è preso ($rc = 12$), l'oggetto 4 è preso ($rc = 0$), nient'altro entra: $\bar{x} = (0, 1, 0, 1, 0, 0)$, profitto 40. Il LP prende gli oggetti $1, 2, 3$ (peso 20) e la frazione $\frac{2}{12} = \frac{1}{6}$ dell'oggetto 4:

$$\tilde{x} = \left(1, 1, 1, \frac{1}{6}, 0, 0\right), \qquad z(\mathrm{LP_{KP}}) = 43 + \frac{19}{6} = \frac{277}{6} \approx 46.17.$$

2. $\tilde{x}$ e $\bar{x}$ concordano su $x_2 = 1$, $x_5 = 0$, $x_6 = 0$, che vengono fissate; x_1, x_3, x_4 sono libere. Il sotto-MIP è

$$\begin{aligned} \max \quad & 21 + 18\,x_1 + 4\,x_3 + 19\,x_4 & & \\ \text{soggetto a} \quad & 8\,x_1 + 2\,x_3 + 12\,x_4 & \leq & \ 12 \\ & 21 + 18\,x_1 + 4\,x_3 + 19\,x_4 & \geq & \ 41 \\ & x_1,\ x_3,\ x_4 & \in & \ \{0,1\} \end{aligned}$$

Se $x_4 = 1$ la capacità impone $x_1 = x_3 = 0$ e il profitto è $40 < 41$. Se $x_4 = 0$, la scelta migliore è $x_1 = x_3 = 1$ (peso $10 \leq 12$), di profitto 43. RINS restituisce $\{1, 2, 3\}$, di profitto 43, ossia l'**ottimo**.

3. La soluzione incumbent riempie lo zaino ($10 + 12 = 22$), quindi aggiungere un oggetto è inammissibile, e togliere oggetti diminuisce il profitto. Le soluzioni a distanza 2 che mantengono il peso ammissibile sono gli scambi (un oggetto esce, uno entra):

esce \ entra	1	3	5	6
2	37	23	29	26
4	39	25	31	28

tutti con profitto minore di 41: il sotto-MIP con $\Delta(x, \bar{x}) \leq 2$ e il cutoff è inammissibile. L'ottimo $(1, 1, 1, 0, 0, 0)$ differisce da $\bar{x}$ in x_1, x_3, x_4: è a distanza 3.

4. **Iterazione 1.** Nessuna soluzione di profitto almeno 41 è a distanza ≤ 2 (Domanda 3). A distanza 3 ce ne sono due: $(1, 0, 1, 1, 0, 0)$, di peso 22 e profitto 41, e $(1, 1, 1, 0, 0, 0)$, di profitto 43. Il sotto-MIP di proximity può restituire l'una o l'altra.

   - Se restituisce $(1, 1, 1, 0, 0, 0)$, il sotto-MIP successivo chiede profitto almeno 44 ed è inammissibile: il metodo si ferma, e 43 è dimostrato ottimo.
   - Se restituisce $(1, 0, 1, 1, 0, 0)$, il sotto-MIP successivo chiede profitto almeno 42: la soluzione più vicina è $(1, 1, 1, 0, 0, 0)$, a distanza 2 (esce l'oggetto 4, entra l'oggetto 2); poi il profitto 44 è inammissibile.

   RINS (hard fixing) ha successo perché le variabili su cui $\tilde{x}$ e $\bar{x}$ concordano mantengono i loro valori ottimi; il local branching con $k = 2$ fallisce perché l'ottimo è a tre cambiamenti di distanza; la proximity search lo trova, poiché l'ampiezza della mossa è decisa dal sotto-MIP.

Esercizio 5.4 — RINS, local branching e proximity search sul KP (KP4)

Consideriamo uno zaino di capacità $c = 13$ e un insieme $\mathscr{I} = \{1, 2, \ldots, 6\}$ di 6 oggetti, ordinati per efficienza non crescente, con i seguenti profitti e pesi:

	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$
p_j	16	13	8	16	10	9
w_j	2	2	4	9	6	8

La soluzione incumbent è la soluzione dell'algoritmo greedy GR (efficienza non crescente).

Domande

1. Calcolare la soluzione incumbent $\bar{x}$ e la soluzione ottima $\tilde{x}$ del rilassamento lineare.

2. Applicare RINS a partire da $\bar{x}$: indicare quali variabili sono fissate e che cosa restituisce il sotto-MIP.

3. Applicare il local branching a partire da $\bar{x}$ con $k = 1$ e poi, dopo aver aggiunto il vincolo inverso, con $k = 2$. Scrivere esplicitamente i vincoli di local branching.

4. Dimostrare che la soluzione trovata nella Domanda 3 è ottima, usando il rilassamento lineare.

Soluzione

1. *GR* prende gli oggetti $1, 2, 3$ (peso 8, $rc = 5$); gli oggetti $4, 5, 6$ non entrano: $\bar{x} = (1, 1, 1, 0, 0, 0)$, profitto 37. Il LP prende gli oggetti $1, 2, 3$ e la frazione $\frac{5}{9}$ dell'oggetto 4:
$$\tilde{x} = \left(1, 1, 1, \frac{5}{9}, 0, 0\right), \qquad z(\mathrm{LP_{KP}}) = 37 + \frac{80}{9} = \frac{413}{9} \approx 45.89.$$

2. Le due soluzioni concordano su $x_1 = x_2 = x_3 = 1$ e $x_5 = x_6 = 0$: solo x_4 è libera. Il sotto-MIP chiede $8 + 9x_4 \leq 13$ e $37 + 16x_4 \geq 38$: il primo vincolo impone $x_4 = 0$, il secondo $x_4 = 1$. È inammissibile: **RINS fallisce** e restituisce $\bar{x}$.

3. $k = 1$**.** Il vincolo è
$$(1 - x_1) + (1 - x_2) + (1 - x_3) + x_4 + x_5 + x_6 \leq 1.$$
I vicini sono i flip: aggiungere l'oggetto 4, 5 o 6 dà peso 17, 14, 16 > 13; togliere un oggetto diminuisce il profitto. Nessuna soluzione ha profitto almeno 38: aggiungiamo il vincolo inverso $(1 - x_1) + (1 - x_2) + (1 - x_3) + x_4 + x_5 + x_6 \geq 2$ e poniamo $k = 2$.

   $k = 2$**.** Ora $2 \leq \Delta(x, \bar{x}) \leq 2$. Due rimozioni perdono profitto, due aggiunte superano la capacità; gli scambi (esce uno fra $1, 2, 3$, entra uno fra $4, 5, 6$) che entrano nello zaino sono

esce \ entra	4	5	6
1	— (peso 15)	31	— (peso 14)
2	— (peso 15)	34	— (peso 14)
3	45	39	38

   Il migliore è lo scambio dell'oggetto 3 con l'oggetto 4: $(1, 1, 0, 1, 0, 0)$, di peso 13 e profitto 45.

4. Il rilassamento lineare dà $z(I) \leq \frac{413}{9} \approx 45.89$ e, poiché i profitti sono interi, $z(I) \leq 45$. La soluzione del local branching raggiunge questo bound: è **ottima**. RINS ha fissato $x_3 = 1$, la variabile che doveva cambiare; il local branching (soft fixing) lascia al solver la scelta delle variabili da cambiare.

Esercizio 5.5 — Proximity search su $P\,||\,C_{\max}$ (PCmax1)

Consideriamo un'istanza di $P\,||\,C_{\max}$ con $m = 2$ macchine e 6 lavori con tempi di lavorazione

	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$
t_j	9	3	9	5	8	2

e il modello MIP con variabili x_{jh} e $C_{\max}$. La soluzione incumbent è la soluzione del list scheduling ($LIST$) nell'ordine $1, 2, \ldots, 6$, con ogni lavoro assegnato alla macchina meno carica (la macchina 1 in caso di pareggio).

Domande

1. Calcolare la soluzione incumbent e il suo makespan. Calcolare poi il lower bound $\max\{\max_j t_j, \lceil \sum_j t_j/m \rceil\}$.
2. Applicare la prima iterazione della proximity search con $\theta = 1$: trovare la soluzione con $C_{\max} \le f(\bar{x}) - 1$ a distanza minima $\Delta(x, \bar{x})$ dalla soluzione incumbent. (Ricordiamo che $\Delta = 2r$, dove r è il numero di lavori che cambiano macchina.)
3. Applicare la seconda iterazione.
4. Applicare la terza iterazione e concludere.

Soluzione

1. Lavoro $1 \to M_1$ $(9, 0)$, lavoro $2 \to M_2$ $(9, 3)$, lavoro $3 \to M_2$ $(9, 12)$, lavoro $4 \to M_1$ $(14, 12)$, lavoro $5 \to M_2$ $(14, 20)$, lavoro $6 \to M_1$ $(16, 20)$. La soluzione incumbent è $J_1 = \{1, 4, 6\}$, $J_2 = \{2, 3, 5\}$, con carichi 16 e 20: makespan 20. Il lower bound è $\max\{9, \lceil 36/2 \rceil\} = 18$.
2. Il cutoff è $C_{\max} \le 19$. Con $r = 1$ (distanza 2) si sposta un solo lavoro; spostare un lavoro su M_2 rende il suo carico maggiore di 20, quindi il lavoro passa da M_2 a M_1, e servono $20 - t_j \le 19$ e $16 + t_j \le 19$, ossia $1 \le t_j \le 3$: solo il lavoro 2. La nuova soluzione incumbent è $J_1 = \{1, 2, 4, 6\}$, $J_2 = \{3, 5\}$, con carichi 19, 17: makespan 19, a distanza 2.
3. Il cutoff è $C_{\max} \le 18$: poiché il totale è 36, entrambi i carichi devono essere esattamente 18, ossia M_1 deve perdere esattamente 1. Un solo spostamento richiederebbe un lavoro con $t_j = 1$: non ce ne sono (la distanza 2 è impossibile). Con due lavori ($r = 2$, distanza 4): due lavori che lasciano M_1 richiederebbero $t_i + t_j = 1$, impossibile; uno scambio di $i \in J_1$ con $j \in J_2$ richiede $t_i - t_j = 1$, e l'unica coppia è il lavoro 1 (9) con il lavoro 5 (8). La nuova soluzione incumbent è $J_1 = \{2, 4, 5, 6\}$, $J_2 = \{1, 3\}$, con carichi 18, 18: makespan 18, a distanza 4.
4. Il cutoff è $C_{\max} \le 17$, ma $C_{\max} \ge 36/2 = 18$ per ogni assegnamento: il sotto-MIP di proximity è inammissibile, e il metodo si ferma. Per la proposizione sulla proximity

search del Capitolo 10 «Euristiche MIP» del modulo *Programmazione lineare intera*, il makespan 18 è **ottimo**; raggiunge anche il lower bound della Domanda 1.