$BP2$ che sia un'istanza-sì se e solo se I è un'istanza-sì, definiamo $s = \sum_{i=1}^{n} a_i$ e distinguiamo due casi.

1. Se $b \geq s/2$, I' consiste di $n+1$ oggetti di volume $a_1, \ldots, a_n, a_{n+1}$, con $a_{n+1} = 2b - s$. La capacità dei bin è b. Se I' è un'istanza-sì, allora questi $n+1$ oggetti possono essere sistemati in due bin. Poiché il volume totale degli oggetti è $2b$, i due bin sono completamente pieni. Ma allora gli oggetti sistemati nel bin che non contiene l'oggetto $n+1$ formano un sottoinsieme $S \subseteq \{1, \ldots, n\}$ tale che $\sum_{i \in S} a_i = b$, cioè I è un'istanza-sì. Viceversa, se I è un'istanza-sì allora è possibile sistemare gli $n+1$ oggetti in due bin, dunque in tal caso anche I' è un'istanza-sì.

2. Se $b < s/2$, I' consiste di $n+1$ oggetti di volume $a_1, \ldots, a_n, a_{n+1}$, con $a_{n+1} = s - 2b$. La capacità dei bin è $s - b$. Se I' è un'istanza-sì, allora questi $n+1$ oggetti possono essere sistemati in due bin. Poiché il volume totale degli oggetti è $2(s-b)$, i due bin sono completamente pieni. Ma allora gli oggetti sistemati nel bin che contiene l'oggetto $n+1$, tolto l'oggetto $n+1$ stesso, formano un sottoinsieme $S \subseteq \{1, \ldots, n\}$ tale che $\sum_{i \in S} a_i = (s-b) - a_{n+1} = b$, cioè I è un'istanza-sì. Viceversa, se I è un'istanza-sì allora è possibile sistemare gli $n+1$ oggetti in due bin, dunque in tal caso anche I' è un'istanza-sì.

Per concludere, basta osservare che la riduzione può essere effettuata in tempo polinomiale. □

Proposizione 5.3 *Se $P \neq NP$, il problema del bin packing non ammette alcun algoritmo di approssimazione di fattore $k < 3/2$ che richieda tempo polinomiale.*

Dimostrazione. Supponiamo per assurdo che esista un tale algoritmo A. Mostriamo che allora potremmo risolvere in tempo polinomiale il problema NP-completo $BP2$, contraddicendo l'ipotesi che $P \neq NP$.

Data un'istanza I del problema di bin packing, eseguiamo l'algoritmo A su I. Se $A(I) \leq 2$, allora esiste una soluzione in cui due bin sono sufficienti, dunque possiamo concludere che la risposta al problema decisionale $BP2$ è "sì". Se, al contrario, $A(I) \geq 3$, allora $OTT(I) \geq A(I)/k \geq 3/k > 2$, dunque due bin non sono sufficienti e la risposta al problema decisionale è "no". In ogni caso, troviamo la risposta corretta ad un problema NP-completo in tempo polinomiale. □

5.2.2 Algoritmi di approssimazione

Illustriamo alcuni semplicissimi algoritmi di approssimazione per il bin packing.

Algoritmo next-fit

L'algoritmo *next-fit* prevede che in ogni istante ci sia sempre solo un bin aperto. Gli oggetti vengono processati nel loro ordine originale. Ad ogni passo, l'oggetto in esame viene inserito, se possibile, nel bin attualmente aperto; quando ciò non è possibile, il bin viene chiuso (e non sarà più utilizzato) e ne viene aperto uno nuovo.

Proposizione 5.4 *L'algoritmo next-fit è un algoritmo di approssimazione di fattore 2. Di più,* $A(I) \leq 2 \cdot OTT(I) - 1$.

Dimostrazione. Assumiamo senza perdita di generalità che la capacità dei bin sia 1. Data un'istanza I, sia $N = A(I)$ il numero di bin utilizzati dalla soluzione restituita dall'algoritmo next-fit. Se consideriamo i primi due bin utilizzati, dalla struttura dell'algoritmo segue chiaramente che gli oggetti sistemati in tali bin hanno un volume totale superiore a 1. Lo stesso vale per la coppia formata dal terzo e quarto bin, dal quinto e sesto, eccetera. Poiché ci sono $\lfloor N/2 \rfloor$ coppie, otteniamo $\sum_{i=1}^{n} a_i > \lfloor N/2 \rfloor$. Allora

$$OTT(I) \geq \left\lceil \sum_{i=1}^{n} a_i \right\rceil \geq \left\lfloor \frac{N}{2} \right\rfloor + 1 \geq \frac{N+1}{2},$$

dove la prima disuguaglianza è di facile verifica. Dunque $A(I) = N \leq 2 \cdot OTT(I) - 1$. □

Esistono istanze per cui vale la relazione $A(I) = 2 \cdot OTT(I) - 1$, dunque il risultato della proposizione precedente non può essere migliorato. Un esempio è dato dall'istanza con $n = 3$, $a_1 = 0.5$, $a_2 = 0.9$, $a_3 = 0.5$: qui $A(I) = 3$ e $OTT(I) = 2$.

Algoritmo first-fit

Una variante dell'algoritmo next-fit è il cosiddetto *first-fit*: anche in questo caso gli oggetti vengono trattati nell'ordine assegnato, ma i bin non vengono mai chiusi fino al completamento dell'algoritmo; ad ogni passo, l'oggetto in esame viene inserito nel primo bin in cui c'è spazio sufficiente, mentre viene aperto un nuovo bin nel caso in cui ciò non sia possibile.

È possibile dimostrate che per questo algoritmo vale la relazione $A(I) \leq \frac{17}{10} OTT(I) + 1$.[1] Si noti che questo *non* può essere considerato un algoritmo di approssimazione di fattore 17/10 secondo la definizione che abbiamo dato, a causa della presenza del termine additivo "+1". In casi come questo si dice che l'algoritmo fornisce un'approssimazione *asintotica* di fattore 17/10.

Algoritmo first-fit decrescente

I due algoritmi discussi sopra consentono di ottenere una soluzione approssimata senza conoscere in anticipo la lista di oggetti, in quanto gli oggetti vengono trattati uno per volta esattamente nell'ordine assegnato: questi algoritmi sono detti *online*. La necessità di processare gli oggetti nell'ordine assegnato può essere una condizione imprescindibile in certe applicazioni, ma è chiaro che quando gli oggetti sono noti a priori e c'è la possibilità di scegliere in che ordine trattarli, si può sperare di ottenere un'approssimazione migliore: algoritmi che sfruttano questa possibilità sono detti algoritmi *offline*.

[1] In realtà vale la disequazione leggermente più forte $A(I) \leq \left\lceil \frac{17}{10} OTT(I) \right\rceil$.

Un esempio di algoritmo offline è il *first-fit decrescente*: prima gli oggetti vengono ordinati in modo che $a_1 \geq \cdots \geq a_n$, poi si applica l'algoritmo first-fit. In questo modo si ottiene una soluzione che soddisfa la proprietà $A(I) \leq \frac{11}{9}OTT(I) + 1$. Si noti che, sebbene il coefficiente 11/9 sia inferiore a 3/2, non stiamo contraddicendo la Proposizione 5.3, in quanto questa è una stima asintotica.

5.3 Problema dello zaino

Consideriamo ancora una volta il problema dello zaino (Sezione 2.2.1) con n oggetti, pesi $p_1, \ldots, p_n > 0$, utilità $u_1, \ldots, u_n > 0$ e portata massima $P > 0$. Attraverso alcuni passaggi, vedremo come per questo problema si possa trovare in tempo polinomiale una soluzione arbitrariamente vicina all'ottimo. In quanto segue assumeremo che $\sum_{i=1}^{n} > P$, altrimenti il problema è di ovvia soluzione.

5.3.1 Problema dello zaino frazionario

Come primo passo, consideriamo la variante frazionaria del problema dello zaino, cioè quella in cui gli oggetti possono essere caricati anche solo parzialmente nello zaino. È facile verificare che una soluzione ottima per questo problema consiste nel caricare gli oggetti dando la precedenza a quelli che garantiscono la maggiore utilità per unità di peso. Formalmente, il seguente algoritmo risolve il problema dello zaino frazionario:

1. Riordiniamo gli oggetti in modo tale che $u_1/p_1 \geq \cdots \geq u_n/p_n$.
2. Definiamo $\ell = \min\{i : p_1 + \cdots + p_i > P\}$.
3. Carichiamo per intero gli oggetti $1, \ldots, \ell - 1$ e quanto più possibile dell'oggetto ℓ.

Si noti che l'oggetto ℓ non viene mai caricato per intero, mentre è possibile che non venga caricato neanche in parte.

La verifica della correttezza dell'algoritmo è lasciata per esercizio. Il numero di operazioni richieste dall'algoritmo è $O(n \log n)$, in quanto il miglior algoritmo per ordinare n numeri richiede appunto $O(n \log n)$ operazioni.[2] Dunque l'algoritmo è polinomiale (perché $n \log n \leq n^2$ per ogni $n \geq 1$).

5.3.2 Algoritmo di approssimazione

L'algoritmo per risolvere in modo esatto il problema dello zaino frazionario permette di ottenere una soluzione per il problema dello zaino la cui utilità totale è almeno la metà del valore ottimo. L'algoritmo è il seguente:

1. Scartiamo eventuali oggetti di peso $p_i > P$.
2. Riordiniamo gli oggetti in modo tale che $u_1/p_1 \geq \cdots \geq u_n/p_n$.

[2] Si tratta dell'algoritmo *merge-sort*, non presentato in questo corso.

3. Definiamo $\ell = \min\{i : p_1 + \cdots + p_i > P\}$.

4. Scegliamo la migliore delle seguenti due soluzioni:

   - $S_1 = \{1, \ldots, \ell - 1\}$;
   - $S_2 = \{\ell\}$.

Proposizione 5.5 *L'algoritmo dato sopra è un algoritmo di approssimazione di fattore 2 per il problema dello zaino.*

Dimostrazione. Indichiamo con z_1 e z_2 i valori della funzione obiettivo associati rispettivamente alle soluzioni S_1 e S_2, e con z^* e z_F i valori ottimi del problema intero e frazionario rispettivamente. Poiché

$$z_1 + z_2 = \sum_{i=1}^{\ell} u_i > z_F \geq z^*,$$

almeno uno tra z_1 e z_2 deve essere maggiore di $z^*/2$. □

L'algoritmo è evidentemente polinomiale, in quanto il suo tempo di calcolo è $O(n \log n)$.

Il risultato della proposizione precedente non può essere migliorato, come mostrato dall'istanza con dati $n = 3$, $P = 2$, $p = (1, 1 + \varepsilon, 1)$ e $u = (1 + \varepsilon, 1 + \varepsilon, 1 - \varepsilon)$, per $\varepsilon > 0$ piccolo a piacere: la soluzione ottima è $S = \{1, 3\}$ con valore ottimo 2, mentre la soluzione restituita dall'algoritmo è una qualunque tra le soluzioni $S_1 = \{1\}$ e $S_2 = \{2\}$, in ogni caso con valore della funzione obiettivo pari a $1 + \varepsilon$.

5.3.3 Schema di approssimazione polinomiale

Mostriamo ora che è possibile trovare in tempo polinomiale una soluzione al problema dello zaino che sia arbitrariamente vicina all'ottimo. Per formalizzare quest'affermazione, definiamo il concetto di *schema di approssimazione*. Dato un problema X, uno schema di approssimazione per X è un algoritmo A che, per ogni istanza $I \in X$ e per ogni $\varepsilon > 0$, restituisce una soluzione ammissibile tale che

$$A(I, \varepsilon) \leq (1 + \varepsilon) \cdot OTT(I) \quad \text{se il problema è di minimizzazione}$$
$$A(I, \varepsilon) \geq \frac{1}{(1 + \varepsilon)} \cdot OTT(I) \quad \text{se il problema è di massimizzazione.}$$

Lo schema di approssimazione è detto polinomiale se, considerando ε una costante fissata, il numero di operazioni richieste è limitato da un polinomio in size(I).[3]

Ricordiamo che nella Sezione 4.3 abbiamo visto due algoritmi di programmazione dinamica che risolvono in modo esatto il problema dello zaino (con

[3]Tali algoritmi vengono indicati con la sigla PTAS (polynomial-time approximation scheme).