組合せ最適化問題

Definition

組合せ最適化問題 (離散最適化問題)

- 決定事項が，[　　]のいずれかである場合や，対象の集合から条件を満たす[　　]，あるいは[　　]といった組み合わせ的な構造をもった最適化問題
- 実行可能領域が[　　]の集合や[　　]で与えられている最適化問題，あるいは，そのように実行可能領域を狭めることができる最適化問題

組合せ最適化問題

最短路問題 各枝を通過するかどうかを決定

巡回セールスマン問題 巡回の順序を決定

スケジューリング問題 ジョブの処理順序と機械への割当を決定

ナップサック問題 アイテムを「選択する」か「選択しない」かを決定

ビンパッキング問題 アイテムを詰め込む組み合わせを決定

巡回セールスマン問題

Definition

セールスマンが n 都市を巡回訪問する．ある都市から出発して，残りの都市をすべて訪問し，出発した都市に戻ってくる．各都市間の移動距離が分かっているとき，移動距離を最小にするような［　　］を決定する問題．

巡回セールスマン問題

5 都市 {A, B, C, D, E} を訪問するとし，都市間の移動距離が表で与えられている巡回セールスマン問題の問題例を考える．

- A-B-C-D-E-A と訪問すると総移動距離は，［　　］(km) となる．
- A-D-C-B-E-A と訪問すると総移動距離は［　　］(km) となる．（この問題例ではこれが最小）

	B	C	D	E
A	30	30	25	10
B		30	45	20
C			25	20
D				30

(km)

ジョブスケジューリング問題

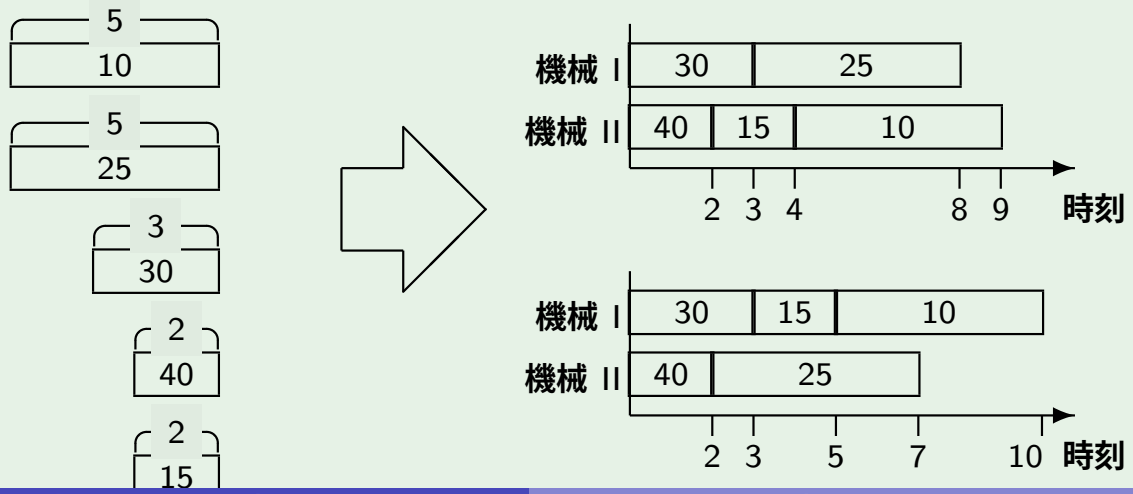
Definition

n 個のジョブを m 台の機械で処理するとき，[　　] と [　　] を決定する．

- ジョブ j の処理時間 p_j と重み w_j が分かっている
- ジョブ j の処理開始時間 s_j は，同じ機械での処理が重ならないように決定
- $\sum_{j=i}^{n} w_j(s_j + p_j)$ → 最小化: [　　]
  ジョブ j の完了時刻を C_j として，$\sum w_j C_j$ → 最小化とも記載
- $\max\{s_j + p_j \mid j = 1, \ldots, n\}$ → 最小化：[　　]
  ($\max_{j=1,\ldots,n} C_j$ → 最小化)

ジョブスケジューリング問題

5 つのジョブを 2 台の機械にスケジュールする．



- 重み付き完了時刻和はどちらも [　　]
- 最終完了時刻は上が [　　]，下が [　　]

ナップサック問題

Definition

n 個のアイテムがあり，各アイテム j の体積 a_j と価値 c_j が分かっている．これらのアイテムからナップサックに詰めるアイテムの組み合わせを決定したい．ただし，選択したアイテムの体積の総和がナップサックの容量を超えてはいけない．価値の和を最大にするような［　　］を決定する問題

ナップサック問題

5 つのアイテム $\{A, B, C, D, E\}$ のそれぞれの体積 a_j と価値 c_j が表で与えられている．これを容量が $12(\ell)$ のナップサックに詰める．

- 価値の大きい順にアイテム C, B を選択すると，これ以外のアイテムからは D のみが選択でき，このとき，価値の和は［　　］
- アイテム A, B, E の体積の和は $12(\ell)$ なのでナップサックにすべて詰めることができ，このときの価値の和は［　　］

アイテム	A	B	C	D	E
体積 a_j (ℓ)	3	4	6	1	5
価値 c_j	6	7	8	1	4

ビンパッキング問題（箱詰め問題）

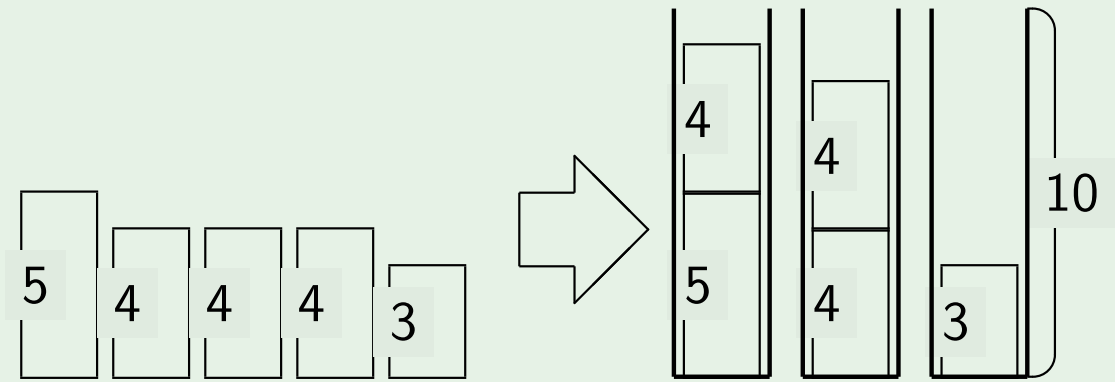
Definition

n 個のアイテムがあり，各アイテム j のサイズ s_j が分かっている．これらのアイテムを容量が S の容器に詰め込む．ただし，一つの容器に詰め込むアイテムのサイズの合計は S を超えてはいけない．容器の個数を最小にする［　　］を決定する問題

ビンパッキング問題

5 つのアイテムがあり，それぞれのサイズが 5, 4, 4, 4, 3 であったとする．これを容量が 10 の容器に詰め込む．

- アイテムのサイズの合計が 20 であるから容器は 2 つ以上必要
- 図のようにアイテムを詰め込むためには 3 つの容器が必要



ビンパッキング問題の定式化

変数

- アイテム j を容器 i に詰めるか否か

$$z_{ji} = \begin{cases} 1 & \text{\_\_\_\_\_} \\ 0 & \text{\_\_\_\_\_} \end{cases}$$

- 容器 i を使うか否かの変数

$$y_i = \begin{cases} 1 & \text{\_\_\_\_\_} \\ 0 & \text{\_\_\_\_\_} \end{cases}$$

制約条件

- 容器の容量を超えない

$$\text{\_\_\_\_\_}$$

- アイテムは必ず１つの容器に詰める

$$\text{\_\_\_\_\_}$$

目的関数

$$\text{\_\_\_\_\_}$$

線形計画問題に変数の整数条件を付加:

整数計画問題

$$
\begin{array}{ll}
\textbf{最大化} & c_1x_1 + c_2x_2 + \cdots + c_nx_n \\
\overset{\boldsymbol{x}}{\textbf{条件}} & a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \leq b_i, \quad i = 1, 2, \ldots, m \\
 & x_j \textbf{は} \_\_\_\_, \quad j = 1, 2, \cdots, n
\end{array}
$$

すべての変数 x_j が 0 か 1 の値に制限されているとき:

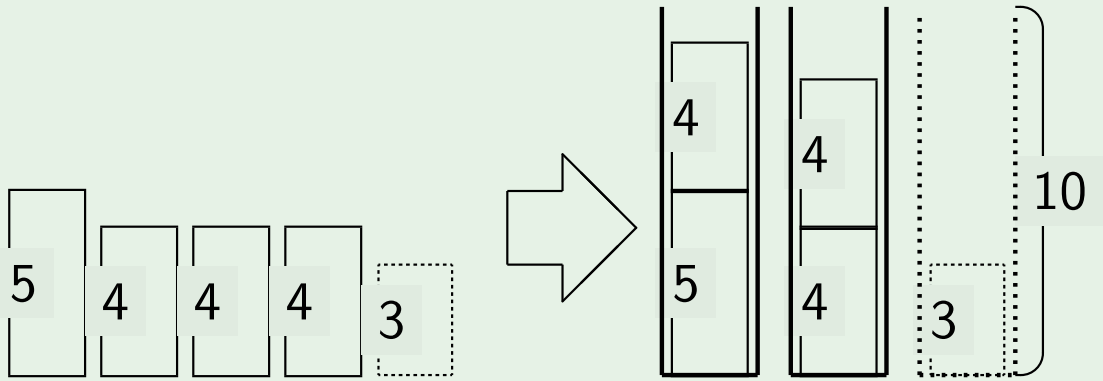
整数計画問題（組合せ最適化問題）が線形計画問題と比べてどの程度難しいのか?

連続ビンパッキング問題

Definition

ビンパッキング問題において，アイテムを切って別の容器に詰め込むことを許すときに容器の個数を最小にする詰め方を決定する問題

連続ビンパッキング問題



ビンパッキング問題 vs 連続ビンパッキング問題

k 個の容器にすべてのアイテムを詰め込めるかどうかを判断する問題

→ すべてのアイテムを詰め込める最小の k を求めれば良い．

- ビンパッキング問題（組合せ最適化問題）
  - 詰め込み方は
  - を調べればよい
- 連続ビンパッキング問題
  - 解は
  - ＬＰを解く手法の利用 （数十万変数の規模でも実用時間内に解ける）

条　件

$$s_1 z_{1i} + s_2 z_{2i} + \cdots + s_n z_{ni} \leq S, \quad i = 1, 2, \ldots, k$$
$$z_{j1} + z_{j2} + \cdots + z_{jk} = 1, \quad j = 1, 2, \ldots, n$$
$$j = 1, 2, \cdots, n, \ i = 1, 2, \ldots, k$$

n 個のアイテムを k 個の容器に詰める詰め方の総数

＊　容量制約は考えない

n 個のアイテムを k 個に分ける[　　]:

$S(n,k) = \frac{1}{k!}\sum_{i=1}^{k}(-1)^{k-i}{}_kC_i i^n$

$k = 10$ のとき

n	10	20	30	40	50	60
$S(n,k)$	1	5.9×10^{12}	1.7×10^{23}	2.4×10^{33}	2.6×10^{43}	2.7×10^{53}
$k^n/k!$ *	2.7×10^{3}	2.7×10^{13}	2.7×10^{23}	2.7×10^{33}	2.7×10^{43}	2.7×10^{53}

＊ 空の容器を許す場合

Exercise

1 秒間に *1* 京 (10^{16}) 個の詰め方の実行可能性をチェック出来る場合，全ての詰め方の実行可能性を調べるのにどのくらいかかるか？

- 1 日=[　　]秒
- 1 年=[　　]秒
- ビックバン以降=[　　]秒

組合せ最適化問題に対する解法

組合せ最適化問題の多くは実行可能解が有限個であるので，すべての解を ______ に調べれば最適解を得られる，問題のサイズが大きくなるとすべての解を列挙するのに必要な時間が天文学的に増加 ______．よって，すべての解を列挙することなく問題を解く方法が必要となる．

______ 最悪の場合，求解までに時間がかかるかもしれないが厳密に最適解を求める

- ______
- ______
- ______
- …

______ 最適解を求めることを諦めて，ある程度最適値に近い値をもつ実行可能解をなるべく早く求める

- ______
- ______
- ______

緩和問題

Definition

そのままでは解きにくい問題に対して，なんらかの条件（いくつかの制約）をなくして解きやすくすることであり，条件や制約をなくした問題.

- 制約をなくすだけでなく，なくした制約に係数をかけて目的関数に組み込む[　　]もある．
- 整数計画問題から変数の整数条件をなくした問題:[　　]
- 緩和問題の実行可能領域は元の問題の実行可能領域を含んでおり，より[　　]．このことから，以下の性質が導ける．

定理

(a) 緩和問題が実行不可能なとき，元の問題も[　　]である．

(b) 緩和問題の最適解が元の問題で実行可能なとき，この解は元の問題の[　　]である．

(c) 元の問題が最大化問題のとき，緩和問題の最適値は元の問題の最適値以上である．つまり，緩和問題の最適値は元の問題の最適値の[　　]を与える．元の問題が最小化問題のとき，緩和問題の最適値は元の問題の最適値以下であり，緩和問題の最適値は元の問題の最適値の[　　]を与える．

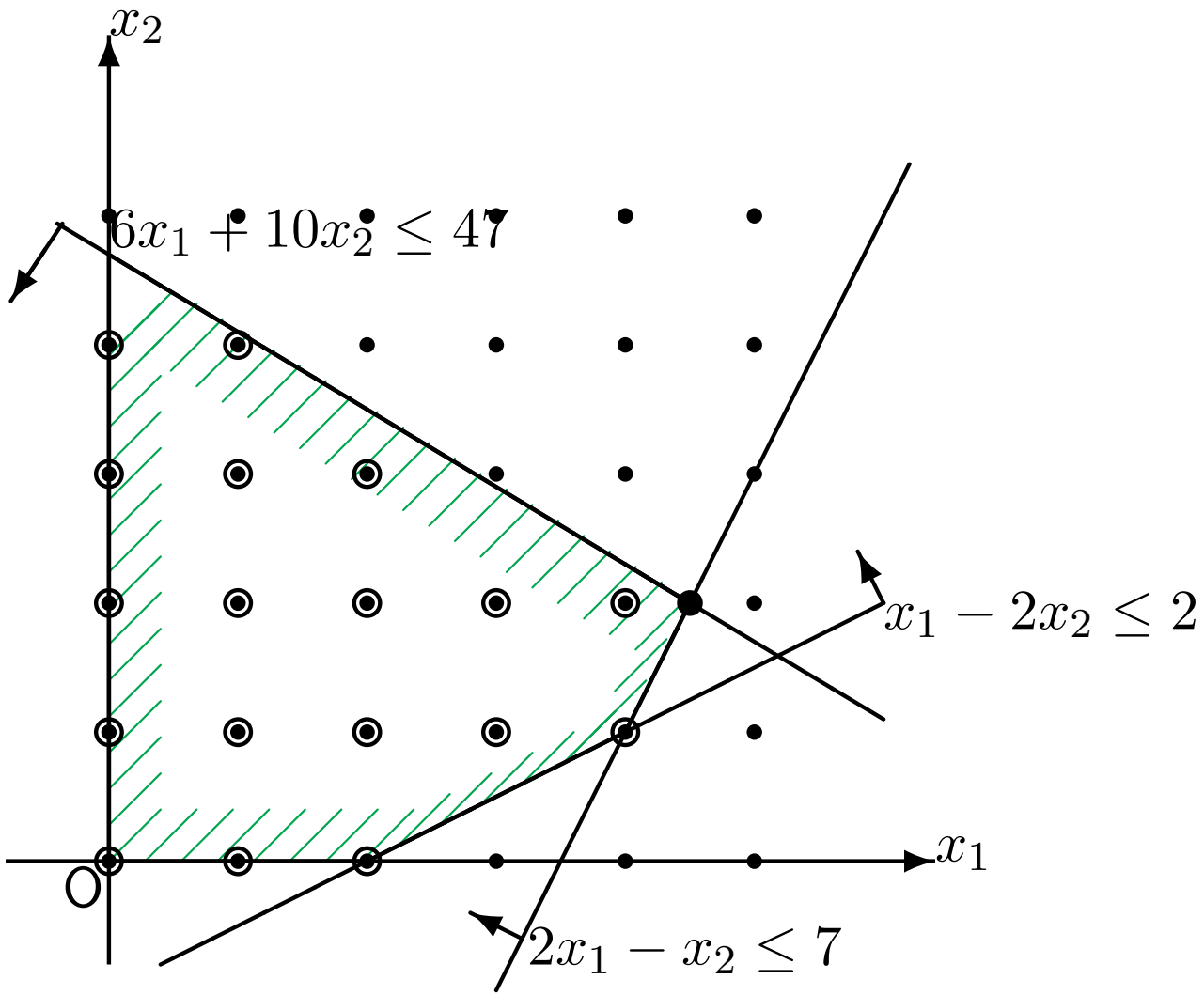
連続緩和

問題:

$$
\left|\begin{array}{lrcrcr}
\textbf{最大化} & x_1 & + & x_2 & & \\
(x_1,x_2) & & & & & \\
\textbf{条　件} & 6x_1 & + & 10x_2 & \leq & 47 \\
 & x_1 & - & 2x_2 & \leq & 2 \\
 & 2x_1 & - & x_2 & \leq & 7 \\
 & x_1, x_2 & : & \textbf{非負整数} & &
\end{array}\right.
$$

を連続緩和したとき，元の問題に対してどのような情報が得られるか．

1. **連続緩和問題を解き，元の問題の最適解と比較せよ．**
2. **目的関数を $3x_1 - 2x_2$ の最大化としたときはどうか．**
3. **目的関数を $19x_1 + 30x_2$ の最大化としたときはどうか．**


x_2
$6x_1 + 10x_2 \leq 47$
$x_1 - 2x_2 \leq 2$
x_1
O
$2x_1 - x_2 \leq 7$

ナップサック問題

Definition

n 個のアイテムがあり，各アイテム j の体積 a_j と価値 c_j が分かっている．これらのアイテムからナップサックに詰めるアイテムの組み合わせを決定したい．ただし，選択したアイテムの体積の総和がナップサックの容量を超えてはいけない．価値の和を最大にするようなアイテムの組み合わせを決定する問題

アイテムは 1 から n の番号付けがされているとする

変数 x_j:

$$x_j = \begin{cases} 1 & \text{\_\_\_\_\_} \\ 0 & \text{\_\_\_\_\_} \end{cases}$$

定式化:

$$\begin{array}{ll} \textbf{最大化} & c_1x_1 + c_2x_2 + \cdots + c_nx_n \\ \boldsymbol{x} & \\ \textbf{条件} & a_1x_1 + a_2x_2 + \cdots + a_nx_n \leq b \\ & x_j \in \{0,1\}, \quad j = 1,2,\ldots,n \end{array}$$

変数の非負整数条件以外の制約式が 1 本の整数計画問題:

すべての変数が 0 か 1 の値をとるように制限されてナップサック問題:

ナップサック問題の仮定

$$
\begin{array}{ll}
\textbf{最大化} & c_1x_1 + c_2x_2 + \cdots + c_nx_n \\
\overset{\boldsymbol{x}}{\textbf{条件}} & a_1x_1 + a_2x_2 + \cdots + a_nx_n \leq b \\
& x_j \in \{0,1\}, \quad j = 1,2,\ldots,n
\end{array}
$$

- **係数** $a_1,\ldots,a_n$ **と** $c_1,\ldots,c_n$ **はすべて** $\square$
- $a_1 + a_2 + \cdots + a_n \square$
- $a_i \square$, $i = 1,2,\ldots,n$

Exercise (特殊なナップサック問題)

すべてのアイテムの体積 a_i が等しいとき，ナップサック問題の答えは簡単にみつかる．なぜ？

連続ナップサック問題

上界値の情報

$$
\left|
\begin{array}{lll}
\textbf{最大化} & c_1x_1 + c_2x_2 + \cdots + c_nx_n & \\
\underset{x}{\textbf{条件}} & a_1x_1 + a_2x_2 + \cdots + a_nx_n \leq b & \\
 & \underline{\qquad\qquad}, & j = 1, 2, \ldots, n
\end{array}
\right.
$$

アイテム j の効率：c_j/a_j

仮定

$c_1/a_1 \geq c_2/a_2 \geq \cdots \geq c_n/a_n$

連続ナップサック問題の解法

連続ナップサック問題の解法

アイテムを $1, 2, \ldots,$ と効率の順にナップサックに詰めていき，容量があって丸ごと入れば，それを 1 個入れる．もしも最後に入れようとするアイテム p が容量不足で丸ごと入らなければ，その一部だけを入れる

$$\bar{x}_j = \begin{cases} 1 & (j = 1, \ldots, p-1) \\ & (j = p) \\ 0 & (j = p+1, \ldots, n) \end{cases}$$

(各アイテムを単位体積に分割して，すべての体積が等しいナップサック問題を解いているとも解釈できる)

Exercise

$\bar{x}$ が連続ナップサック問題の最適解であることを示せ．

連続ナップサック問題

5 つのアイテムを容量 $12(\ell)$ のナップサックに詰める連続ナップサック問題

アイテム	A	B	C	D	E
体積 a_j (ℓ)	3	4	6	1	5
価値 c_j	6	7	8	1	4
効率 c_j/a_j	2.00	1.75	1.33	1.00	0.80
$\bar{x}_j$					

最適値：

貪欲アルゴリズム

最適解でなくても，なるべく良い実行可能解を簡単に求める方法

アイテムを $1, 2, \ldots,$ と効率の順にナップサックに詰めていき，

- **容量があって丸ごと入れば，_____**
- **容量不足で丸ごと入れられないときは，_____，次のアイテムを調べる**

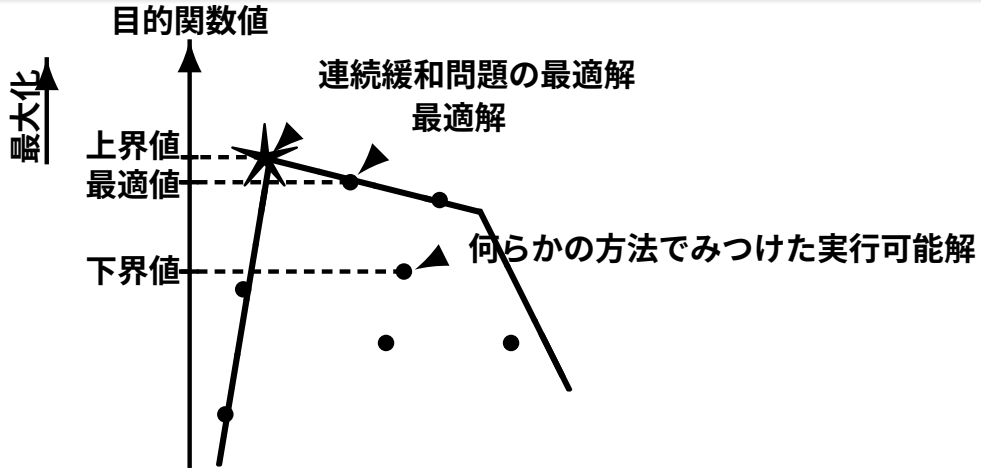
ナップサック問題に対する貪欲アルゴリズム

アイテムを効率の順にソート $(c_1/a_1 \geq c_2/a_2 \geq \cdots \geq c_n/a_n)$
$b' \leftarrow b$ **(ナップサック容量の残量を表すパラメータの初期化)**
$i \leftarrow 1$ **(アイテムを表す添え字の初期化)**
while $i \leq n$ **かつ** $b' > 0$ **do**
  if $a_i \leq b'$ **then**
    $x_i \leftarrow 1; b' \leftarrow b' - a_i$
  else
    $x_i \leftarrow 0$
  end if
  $i \leftarrow i + 1$
end while

5 つのアイテムを容量 $12(\ell)$ のナップサックに詰めるナップサック問題

アイテム	A	B	C	D	E
体積 a_j (ℓ)	3	4	6	1	5
価値 c_j	6	7	8	1	4
効率 c_j/a_j	2.00	1.75	1.33	1.00	0.80
$\bar{x}_j$					

目的関数値：



下界値：

上界値：

最適値は？

演習問題

表で与えられる 8 個のお菓子の中から 500 円以内で価値の和が最大となるように選択したい．

アイテム	ガム	あめ	りんごパフ	とろりチョコ	山盛り煎餅	いもチップ	ラムネ	しっとりクッキー
値段 a_j (十円)	5	1	13	20	15	8	7	21
価値 c_j	2	1	8	7	4	6	3	8

(a) 0-1 ナップサック問題に定式化せよ．

(b) 連続ナップサック問題による上界値と貪欲アルゴリズムによる下界値を求めよ．