Variant of *KP01*:

Knapsack Problem with Minimization Objective Function (*KP01-Min*)

Given:

n	**items,**
P_j	**"profit" of item j, $j = 1, \ldots, n$ ($P_j > 0$),**
W_j	**"weight" of item j, $j = 1, \ldots, n$ ($W_j > 0$),**
one	**container ("knapsack") with "threshold" B:**

Knapsack Problem with Minimization Objective Function (*KP01-Min*)

Given:

- ***n*** **items,**
- P_j **"profit" of item** j**,** $j = 1, \ldots, n$ $(P_j > 0)$**,**
- W_j **"weight" of item** j**,** $j = 1, \ldots, n$ $(W_j > 0)$**,**
- ***one*** **container ("knapsack") with "threshold"** B**:**

determine a subset of the n **items so as to minimize the global profit, and such that the global weight is not smaller than the knapsack threshold** B**.**

KP01-Min **is NP-Hard**

Feasibility Problem **of** ***KP01-Min?***

Mathematical Model of *KP01-Min*

$$y_j = \begin{cases} 1 & \text{if item } j \text{ is inserted in the knapsack} \\ 0 & \text{otherwise} \end{cases} \qquad (j = 1, \ldots, n)$$

$$\min \quad \Sigma_{j=1,n} \; P_j \, y_j$$

$$\Sigma_{j=1,n} \; W_j \, y_j \geq B$$

$$y_j \in \{0, 1\} \qquad (j = 1, \ldots, n)$$

BLP Model (Binary Linear Programming Model)

***KP01-Min* is "equivalent" to *KP01*.**

KP01-Min is "equivalent" to *KP01*.

Set $y_j = 1 - x_j$ $(j = 1, \ldots, n)$ **and replace** y_j **with** $1 - x_j$

1) $\min\ T = \sum_{j=1,n} P_j\, y_j = \sum_{j=1,n} P_j\,(1 - x_j) =$

$$P - \max \sum_{j=1,n} P_j\, x_j$$

where $P = \sum_{j=1,n} P_j$

KP01-Min is "equivalent" to *KP01* (2).

2) $\sum_{j=1,n} W_j\, y_j = \sum_{j=1,n} W_j\,(1 - x_j) =$

$$\sum_{j=1,n} W_j - \sum_{j=1,n} W_j\, x_j \geq B$$

$$\sum_{j=1,n} W_j\, x_j \leq C'$$

where $C' = \sum_{j=1,n} W_j - B$

KP01-Min is "equivalent" to *KP01* (3).

$$\textbf{Min } T = P - \max \sum_{j=1,n} P_j x_j$$

$$\sum_{j=1,n} W_j x_j \leq C'$$

$$x_j \in \{0, 1\} \qquad (j = 1, \ldots, n)$$

where: $P = \sum_{j=1,n} P_j$; $C' = \sum_{j=1,n} W_j - B$

- ***Problem KP01***
- ***KP01-Min* is *NP-Hard***

Variant of *KP01*: *Equality-KP01* (*E-KP01*)

- **Same input data as for the *KP01*: n, C, (P_j), (W_j)**

* **Determine a subset of the n items so that the global profit is maximum and the global weight is equal to the knapsack capacity C.**

$$\max \quad \sum_{j=1,n} P_j \, x_j$$

$$\sum_{j=1,n} W_j \, x_j = C$$

$$x_j \in \{0, 1\} \quad (j = 1, \ldots, n) \qquad \text{(BLP Model)}$$

The *Feasibility Problem* of *E-KP01* is NP-Hard

***E-KP01* is NP-Hard**

Feasibility Problem of *E-KP01* (*F-E-KP01*)

- **Input data: *n, C, (W_j)***
- **Determine a subset of the *n* items so that the global weight is equal to the knapsack capacity *C*.**

- ***F-E-KP01* is NP-Hard**
- **Input: *n, C, (W_j)* :**

  Size: 2 + *n* : *n*
- ***Binary Decision Tree of KP01:***

  F-E-KP01 $\in$ ***Class NP***

F-E-KP01 is NP-Hard

- Input data: n, C, (W_j)
- Determine a subset of the n items so that the global weight is equal to the knapsack capacity C.
- $PP \propto$ *F-E-KP01* :
- Given any instance of PP: t, (a_j), b (Size: t)

1) Define (in time $O(t)$) an instance $(n, C, (W_j))$ of *F-E-KP01*:
   - $n := t$
   - $C := b$
   - $W_j := a_j$ ($j = 1, \ldots, n$).
2) Determine (if it exists) a feasible solution (x_j) of *F-E-KP01*.
3) If a feasible solution (x_j) of *F-E-KP01* exists, then PP has a feasible solution (x_j)

   Otherwise: PP has no feasible solution.

Computing time $O(n)$ (hence $O(t)$, polynomial in the size of PP)

Variant of *KP01*:
Subset Sum Problem (*SSP*)

- **Item j has weight W_j and profit $P_j = W_j$ ($j = 1, \ldots, n$): Determine a subset of the n items so that the global weight is maximum and not greater than C.**

Cut of metal bars with minimization of the waste.

$$\max \quad \sum_{j=1,n} W_j \, x_j$$

$$\sum_{j=1,n} W_j \, x_j \le C$$

$$x_j \in \{0, 1\} \quad (j = 1, \ldots, n) \qquad \text{(BLP Model)}$$

***SSP* is NP-Hard**

Variant of *KP01*:
Subset Sum Problem (*SSP*)

- **Item j has weight W_j and profit $P_j = W_j$ ($j = 1, \ldots, n$):**
  Determine a subset of the n items so that the global weight is maximum and not greater than C.

Cut of metal bars with minimization of the waste.

$$\max \quad \sum_{j=1,n} W_j \, x_j$$

$$\sum_{j=1,n} W_j \, x_j \le C$$

$$x_j \in \{0, 1\} \quad (j = 1, \ldots, n) \qquad \text{(BLP Model)}$$

***SSP* is NP-Hard** ***Feasibility Problem* of *SSP*?**

SSP is NP-Hard

- ***SSP***: **input data: n, C, (W_j).**
- **Determine a subset of the n items so that the global weight is maximum and not greater than C.**
- **Size: $2 + n$: n**
- ***Binary Decision Tree of KP01:*** ***SSP*** $\in$ ***Class NP***

* ***PP*** $\propto$ ***SSP***

Given any instance of *PP*: t, (a_j), b (Size: t)

1) Define (in time $O(t)$) an instance $(n, (W_j), C)$ of *SSP*:

* $n := t$; $C := b$; $W_j := a_j$ ($j = 1, \ldots, n$).

2) Determine the optimal solution $(x_1, x_2, \ldots, x_n, z)$ of *SSP*.

3) If $z = C$: *PP* has a feasible solution $(x_1, x_2, \ldots, x_n)$

If $z < C$: *PP* has no feasible solution

Computing time $O(n)$ (hence $O(t)$, polynomial in the size of *PP*)

Subset Sum Problem (SSP)

- **Item j has weight W_j and profit $P_j = W_j$ ($j = 1, \ldots, n$):**
- **Determine a subset of the n items so that the global weight is maximum and not greater than C.**

$$\max \quad \sum_{j=1,n} W_j x_j$$

$$\sum_{j=1,n} W_j x_j \leq C$$

$$x_j \in \{0, 1\} \quad (j = 1, \ldots, n) \qquad \textbf{(BLP Model)}$$

***SSP* is NP-Hard.**

SSP* is a special case of *KP01

The *feasibility problem* of *SSP* is polynomial

Variant of *KP01*:
Change Making Problem (*CMP*)

- Given *n banknotes* and *a cheque (check)*,

* W_j is the *value* of banknote j ($j = 1, \ldots, n$), with $W_j > 0$,

- C is the value of the cheque:

- select a *minimum cardinality* subset of banknotes so that the global value is equal to C.

$$\min \quad \sum_{j=1,n} x_j$$

$$\sum_{j=1,n} W_j \, x_j = C$$

$$x_j \in \{0, 1\} \quad (j = 1, \ldots, n) \qquad \textbf{(BLP Model)}$$

CMP is NP-Hard

Variant of *KP01*:
Change Making Problem (*CMP*)

- **Given *n banknotes* and *a cheque* *(check)*,**

* **W_j is the *value* of banknote j ($j = 1, \ldots, n$), with $W_j > 0$,**

- **C is the value of the cheque:**
- **select a *minimum cardinality* subset of banknotes so that the global value is equal to C.**

$$\min \quad \sum_{j=1,n} x_j$$

$$\sum_{j=1,n} W_j \, x_j = C$$

$$x_j \in \{0, 1\} \quad (j = 1, \ldots, n) \qquad \textbf{(BLP Model)}$$

***CMP* is NP-Hard (its *Feasibility Problem* is NP-Hard)**

Feasibility Problem of *CMP* (*F-CMP*)

- **Input: n, C, (W_j)**
- **select a subset of banknotes so that the global value is equal to C.**

- ***F-CMP* is NP-Hard:**
- **“Banknotes” correspond to “items”;**
- **“Cheque value” corresponds to “capacity C”;**

- ***F-CMP is identical to F-E-KP01.***

- ***CMP* is NP-Hard**

Variant of *KP01*:
Two-Constrained KP (*2C-KP*)

Given:

n items,

P_j "profit" of item j, $j = 1, \dots, n$ ($P_j > 0$),

W_j "weight" of item j, $j = 1, \dots, n$ ($W_j > 0$),

V_j "volume" of item j, $j = 1, \dots, n$ ($V_j > 0$),

one **container ("knapsack") with:**

*** "weight capacity" C, and "volume capacity" D:**

determine a subset of the n items so as to maximize the global profit, and such that the global weight is not greater than the weight capacity C *and* the global volume is not greater than the volume capacity D.

2C-KP **is NP-Hard**

Mathematical Model for 2C-KP

$$\max \quad \sum_{j=1,n} P_j \, x_j$$

$$\sum_{j=1,n} W_j \, x_j \leq C$$

$$\sum_{j=1,n} V_j \, x_j \leq D$$

$$x_j \in \{0, 1\} \quad (j = 1, \ldots, n) \qquad \textbf{(BLP Model)}$$

2C-KP is NP-Hard

Input: n, C, D , (P_j), (W_j), (V_j) $j = 1, \ldots, n$;

determine a subset of the n items so as to maximize the global profit, and such that the global weight is not greater than the weight capacity C *and* the global volume is not greater than the volume capacity D.

- **Size: $3 + 3n$: n**
- ***Binary Decision Tree of KP01:***

2C-KP $\in$ ***Class NP***

* ***2C-KP is a generalization of KP01:***

(*KP01* $\propto$ *2C-KP*)

The feasibility problem of *2C-KP* is polynomial.

Variant of *KP01*: *Bounded-KP* (*BKP*)

In addition to the input data for KP01:

*** d_j = number of available items of item-type j (j = 1, …, n)**

Input: n, C, (P_j), (W_j), (d_j) j = 1, …, n ; (P_j > 0), (W_j > 0)

determine for each item-type j (j = 1, …, n) the number of items to be inserted in the knapsack so as to maximize the global profit, and such that the global weight is not greater than the capacity C.

Variant of *KP01*: *Bounded-KP* (*BKP*)

In addition to the input data for KP01:

* d_j = number of available items of item-type j ($j = 1, \ldots, n$)
* x_j = number of items selected for item-type j ($j = 1, \ldots, n$)

$$\max \quad \sum_{j=1,n} P_j \, x_j$$

$$\sum_{j=1,n} W_j \, x_j \le C$$

$$0 \le x_j \le d_j \quad \text{INTEGER} \quad (j = 1, \ldots, n)$$

ILP Model; *BKP* is NP-Hard

BKP is NP-Hard

Input: n, C, (P_j), (W_j), (d_j) $j = 1, \ldots, n$;

determine for each item-type j ($j = 1, \ldots, n$) the number of items to be inserted in the knapsack so as to maximize the global profit, and such that the global weight is not greater than the weight capacity C.

- **Size: $2 + 3n$: n (number of "symbols" required to represent the input data)**
- ***Decision Tree*: *n levels*, one for each item-type j ;**

* **$(d_j + 1)$ *descendent nodes* (one node for each posssible number of inserted items of item-type j) *and constant time for each node*:**

***BKP* $\in$ *Class NP* (???)**

* **is d_j a polynomial function of the size n?**

***BKP* is NP-Hard**

Input: n, C, (P_j), (W_j), (d_j) $j = 1, \ldots, n$;

determine for each item-type j ($j = 1, \ldots, n$) the number of items to be inserted in the knapsack so as to maximize the global profit, and such that the global weight is not greater than the weight capacity C.

- **Size: $2 + 3\,n$: n (number of “symbols” required to represent the input data)**
- ***Decision Tree*: *n levels*, one for each item-type j ;**

$(d_j + 1)$ *descendent nodes* (one node for each possible number of inserted items of item-type j) *and constant time for each node*:

BKP $\in$ ***Class NP*** **(???)**

* **is d_j a polynomial function of the size n?**

* $$B = \max\{d_j : j = 1, \ldots, n\} \leq (2)^k$$

where k is the number of bits needed to represent B

* **Size: $2 + 2\,n + k*n$: $k*n$ (number of “bits”)**

* **d_j is defined by an exponential function of the size $k*n$**

Transformation of an ILP model with n variables into a BLP model with $n*k$ variables ($B \leq (2)^k$):

- **x_j integer variable with $x_j \geq 0$, $x_j \leq d_j$ (with $d_j \leq B$).**
- **for each variable x_j ($j = 1, \ldots, n$) introduce k binary variables t_{jh} , with $h = 1, \ldots, k$**

$$x_j = \sum_{h=1,k} 2^{h-1} t_{jh}$$

$$t_{jh} \in \{0, 1\} \quad h = 1, \ldots, k \quad (j = 1, \ldots, n)$$

$$k = \lceil z \rceil \qquad \text{with} \quad z = \log_2 (B + 1) \qquad\qquad B \leq (2)^k$$

BLP model with $n*k$ variables:

Binary Decision Tree with $n*k$ levels (polynomial function of the size $k*n$).

BKP is NP-Hard

Input: n, C, (P_j), (W_j), (d_j) $j = 1, \ldots, n$;

determine for each item-type j ($j = 1, \ldots, n$) the number of items to be inserted in the knapsack so as to maximize the global profit, and such that the global weight is not greater than the weight capacity C.

- **Size:** $2 + 2n + k*n$: $k*n$
- **for each variable x_j ($j = 1, \ldots, n$) introduce k binary variables t_{jh}, with $h = 1, \ldots, k$**

***Binary Decision Tree*: *k*n levels* (one for each binary variable t_{jh});**

* 2 ***descendent nodes and constant time for each node*:**
BKP $\in$ ***Class NP***

* ***BKP is a generalization of KP-01*** (***KP-01*** $\propto$ ***BKP***)

The feasibility problem of *BKP* is polynomial.

Variant of *KP01*: *Unbounded-KP* (*UKP*)

No limit on the number of items available for each item-type j (j = 1, …, n). $(P_j > 0)$, $(W_j > 0)$

- x_j = **number of items selected for item-type j (j = 1, …, n)**

$$\max \quad \sum_{j=1,n} P_j\, x_j$$

$$\sum_{j=1,n} W_j\, x_j \le C$$

$$x_j \ge 0 \qquad \text{INTEGER} \quad (j = 1, \ldots, n)$$

ILP Model

It is known that *UKP* is NP-Hard

Variant of *KP01*:
Unbounded-KP (*UKP*)

No limit on the number of items available for each item-type ($d_j = \infty$, $j = 1, \ldots, n$)

- **x_j = number of items selected for item-type j ($j = 1, \ldots, n$)**

$$\max \quad \sum_{j=1,n} P_j \, x_j$$

$$\sum_{j=1,n} W_j \, x_j \le C$$

$$x_j \ge 0 \qquad \text{INTEGER} \qquad (j = 1, \ldots, n)$$

***UKP* is a special case of *BKP*: $d_j = int\,(C / W_j)$ $j = 1, \ldots, n$**

The feasibility problem of *UKP* is polynomial.

Variant of *KP01*
Multiple Choice KP (*MCKP*)

In addition to the input data for KP01:

the set of the n items is *partitioned* into k disjoint subsets $N_1, N_2, \ldots, N_k$.

- **determine a subset of the n items, with at most one item for each subset N_h ($h = 1, \ldots, k$), so as to maximize the global profit, and such that the global weight is not larger than the knapsack capacity C.**

- P_j ***positive*?** W_j **positive?**
- ***Feasibility Problem of MCKP ??***

BLP Model for *MCKP* (2)

- determine a subset of the n items, with at most one item for each subset N_h (h = 1, …, k), so as to maximize the global profit, and such that the global weight is not larger than the knapsack capacity C.

$$\max \quad \sum_{j=1,n} P_j \, x_j$$

$$\sum_{j=1,n} W_j \, x_j \le C$$

$$\sum_{j \in N_h} x_j \le 1 \qquad (h = 1, \ldots, k)$$

$$x_j \in \{0, 1\} \qquad (j = 1, \ldots, n)$$

BLP Model **MCKP is NP-Hard**

BLP Model for *MCKP* (3)

* Define the *Binary Matrix* A_{hj} $(h = 1, \dots, k;\ j = 1, \dots, n)$, with:

- $A_{hj} = 1$ if $j \in N_h$
- $A_{hj} = 0$ otherwise.
- *Matrix* A_{hj} belongs to the input data of the instance

$$\max \quad \sum_{j=1,n} P_j\, x_j$$

$$\sum_{j=1,n} W_j\, x_j \le C$$

$$\sum_{j=1,n} A_{hj}\, x_j \le 1 \qquad (h = 1, \dots, k)$$

$$x_j \in \{0, 1\} \qquad (j = 1, \dots, n)$$

Multiple Choice KP (MCKP) is NP-Hard

MCKP: in addition to the input data for KP01:

the set of the n items is *partitioned* into k disjoint subsets N_1, $N_2, \ldots, N_k$.

- determine a subset of the n items, with at most one item for each subset N_h ($h = 1, \ldots, k$), so as to maximize the global profit, and such that the global weight is not larger than the knapsack capacity C.

- Input: n, C, k, (P_j), (W_j) ($j = 1, \ldots, n$), N_h ($h = 1, \ldots, k$)
- Size: $3 + 2n + k * n$ (matrix A_{hj}), with $k \leq n$: $n * n$
- Size: $3 + 2n + n$ (partition of the set $\{1, 2, \ldots, n\}$) : n .
- *Binary Decision Tree*: similar to the decision tree of *KP-01*: *n levels, 2 descendent nodes and constant time for each node*:
- *MCKP* $\in$ *Class NP* ;
- *MCKP* is a "generalization" of *KP-01* : *KP-01* $\propto$ *MCKP*

BLP Model for *MCKP*

* *Binary Matrix* A_{hj} ($h = 1, \ldots, k$; $j = 1, \ldots, n$), with:

- $A_{hj} = 1$ if $j \in N_h$; $A_{hj} = 0$ otherwise.

$$\max \quad \sum_{j=1,n} P_j\, x_j$$

$$\sum_{j=1,n} W_j\, x_j \le C$$

$$\sum_{j=1,n} A_{hj}\, x_j \le 1 \qquad (h = 1, \ldots, k)$$

$$x_j \in \{0, 1\} \qquad (j = 1, \ldots, n)$$

The *BLP Model* has a number of binary variables x_j **polynomial** in the size of *MCKP*:

MCKP $\in$ *Class NP*

Multiple Knapsack Problem (MKP01)

Given: n **items**, m **containers (knapsacks)**

P_j **profit of item** j **(** $j = 1, \ldots, n$**)**

W_j **weight of item** j **(** $j = 1, \ldots, n$**)**

C_i **capacity of container** i **(** $i = 1, \ldots, m$**)**

insert a subset of the n items in each container in order to maximize the global profit of the items inserted in the containers, and in such a way that the sum of the weights of the items inserted in each container i (i = 1, …, m) is not greater than the corresponding capacity C_i

Each item can be inserted in at most one container.

MKP01 (2)

Given: n items, m containers (knapsacks)

P_j profit of item j $\quad (j = 1, \ldots, n)$

W_j weight of item j $\quad (j = 1, \ldots, n)$

C_i capacity of container i $\quad (i = 1, \ldots, m)$

insert a subset of the n items in each container in order to maximize the global profit of the items inserted in the containers, and in such a way that the sum of the weights of the items inserted in each container i (i = 1, …, m) is not greater than the corresponding capacity C_i

$P_j > 0 \quad (j = 1, \ldots, n)$

$W_j > 0 \quad (j = 1, \ldots, n)$???

MKP01 (3)

Given: n items, m containers (knapsacks)

P_j profit of item j $\quad (j = 1, \ldots, n)$

W_j weight of item j $\quad (j = 1, \ldots, n)$

C_i capacity of container i $\quad (i = 1, \ldots, m)$

$P_j > 0 \;\; (j = 1, \ldots, n); \;\; W_j > 0 \;\; (j = 1, \ldots, n)$

$\sum_{j=1,n} W_j > \max\{ C_i : i = 1, \ldots, m\}$

$W_j \leq \max\{ C_i : i = 1, \ldots, m\} \quad (j = 1, \ldots, n)$

$\min\{ C_i : i = 1, \ldots, m\} \geq \min\{ W_j : j = 1, \ldots, n\}$

Mathematical Model of MKP01

$$x_{ij} = \begin{cases} 1 & \text{if item } j \text{ is inserted in container } i \\ 0 & \text{otherwise} \end{cases} \qquad (i = 1, \ldots, m;\ j = 1, \ldots, n)$$

$$\max \quad \Sigma_{j=1,n}\ P_j\ (\Sigma_{i=1,m}\ x_{ij})$$

$$\Sigma_{j=1,n}\ W_j\ x_{ij} \leq C_i \qquad (i = 1, \ldots, m)$$

$$\Sigma_{i=1,m}\ x_{ij} \leq 1 \qquad (j = 1, \ldots, n)$$

$$x_{ij} \in \{0,1\} \qquad (i = 1, \ldots, m;\ j = 1, \ldots, n)$$

BLP Model

MKP01 is NP-Hard

Feasibility Problem of MKP01 ??

MKP01 is NP-Hard

MKP01: given: n items, m containers (knapsacks),
P_j profit of item j, W_j weight of item j ($j = 1, \dots, n$),
C_i capacity of container i ($i = 1, \dots, m$):

insert a subset of the n items in each of the m containers in order to maximize the global profit of the inserted items, and in such a way that the global weight of the items inserted in each container i (i = 1, …, m) is not greater than the corresponding capacity C_i

Input: n, m, (P_j), (W_j) ($j = 1, \dots, n$), (C_i) ($i = 1, \dots, m$)

- Size: $2 + 2\,n + m$: $n + m$, ($m \le n$: Size n) ***
- *Decision Tree*: *n levels* (one for each item j);

(*m* +1) *descendent nodes* (*insert item j in knapsack 1, or 2, …, or m, or in no knapsack*) *and constant time for each node*:

MKP01 $\in$ *Class NP* ;

(*BLP model* with (m * n) binary variables x_{ij})

- *MKP01* is a "generalization" of *KP-01* :

KP-01 $\propto$ *MKP01*

Generalized Assignment Problem (GAP)

Given: ***m machines*** **(persons) and *n jobs* (tasks):**

c_{ij} **cost for assigning job j to machine i ($i = 1, \ldots, m$; $j = 1, \ldots, n$);**

r_{ij} ***amount of resource utilized for*** **assigning job j to machine i ($i = 1, \ldots, m$; $j = 1, \ldots, n$); $r_{ij} \geq 0$; *****

b_i **amount of resource available for machine i ($i = 1, \ldots, m$), $b_i > 0$.**

Assign each job to a machine so as to minimize the global cost, and in such a way that the global resource utilized by each machine i is not greater than the corresponding available resource b_i

Generalized Assignment Problem (GAP)

Assign each job to a machine so as to minimize the global cost, and in such a way that the global resource utilized by each machine i is not greater than the corresponding available resource b_i.

***GAP* is NP-Hard**

The *Feasibility Problem* of *GAP* is NP-Hard

Decisional binary variables:

x_{ij} = 1 if job j is assigned to machine i;

x_{ij} = 0 otherwise; $(i = 1, \ldots, m;\ j = 1, \ldots, n)$

Mathematical Model of *GAP*

- **Objective function (minimum cost)**

  $$\min \quad \Sigma_{i=1,m} \Sigma_{j=1,n}\, c_{ij}\, x_{ij}$$

- **One machine assigned to each job:**

  $$\Sigma_{\,i=1,m}\; x_{ij} \;=\; 1 \qquad\qquad (\,j = 1, \ldots, n)$$

- **Resource utilized for each machine:**

  $$\Sigma_{\,j=1,n}\; r_{ij}\, x_{ij} \;\leq\; b_i \qquad\qquad (\,i = 1, \ldots, m)$$

$$x_{ij} \in \{0, 1\} \qquad\qquad (i = 1, \ldots, m,\; j = 1, \ldots, n)$$

BLP Model

GAP is NP-Hard

Given: ***m* machines and *n* jobs:**

c_{ij} cost (*r_{ij} amount of resource utilized*) for assigning job j to machine i ($i = 1, \ldots, m$; $j = 1, \ldots, n$);

b_i amount of resource available for machine i ($i = 1, \ldots, m$):

Assign each job to a machine so as to minimize the global cost, and in such a way that the global resource utilized by each machine i is not greater than the corresponding available resource b_i.

Input: m, n, (c_{ij}), (r_{ij}) ($i = 1, \ldots, m$; $j = 1, \ldots, n$);

(b_i) ($i = 1, \ldots, m$)

Size: $2 + 2\, m * n + m$: $m * n$ ***

The *Feasibility Problem of GAP (F-GAP)* is NP-Hard.

Feasibility Problem of GAP (F-GAP)

Given: ***m machines*** **and** ***n jobs*:**

r_{ij} ***amount of resource utilized*** **for assigning job** j **to machine** i **(**$i = 1, \ldots, m$**;** $j = 1, \ldots, n$**);**

b_i **amount of resource available for machine** i **(**$i = 1, \ldots, m$**):**

Assign each job to a machine in such a way that the global resource utilized by each machine i is not greater than the corresponding available resource b_i**.**

Input: $m, n, (r_{ij})$ $(i = 1, \ldots, m; j = 1, \ldots, n)$; (b_i) $(i = 1, \ldots, m)$:

Size: $m * n$

- ***Decision Tree*: *n levels* (one for each job *j*);**

***** ***m descendent nodes*** **(*insert job j in machine 1, or 2, …, or m*)** ***and constant time for each node*:**

F-GAP $\in$ ***Class NP***

Also *GAP* $\in$ ***Class NP*** **(same *Size* and *Decision Tree* as *F-GAP*);**
(*BLP model* with (*m* * *n*) binary variables x_{ij} **)**

Feasibility Problem of GAP (F-GAP)

Given: ***m machines*** **and** ***n jobs*****:**

r_{ij} ***amount of resource utilized*** **for assigning job** j **to machine** i **(**$i = 1, \ldots, m$**;** $j = 1, \ldots, n$**);**

b_i **amount of resource available for machine** i **(**$i = 1, \ldots, m$**):**

PP $\propto$ ***F-GAP*** **:**

- **Given any instance of** ***PP*****:** t, (a_j), b **(Size:** t**)**

1) Define (in time $O(t)$**) an instance (**$m, n, (r_{ij}), (b_i)$**) of** ***F-GAP*****:**

    * $n := t$

    * $m := 2$; $b_1 := b$; $b_2 := \Sigma_{j=1,t}\, a_j - b$

    * $r_{1j} := a_j$; $r_{2j} := a_j$ ($j = 1, \ldots, n$).

2) Determine (if it exists) a feasible solution (x_{1j} , x_{2j}**) of** ***F-GAP.***

3) If a feasible solution of ***F-GAP*** **exists, then** ***PP*** **has a feasible solution:**

Feasibility Problem of GAP (F-GAP)

Given: m **machines and** n **jobs:**

r_{ij} ***amount of resource utilized*** **for assigning job** j **to machine** i ($i = 1, \ldots, m$; $j = 1, \ldots, n$);

b_i **amount of resource available for machine** i ($i = 1, \ldots, m$):

$PP \propto F\text{-}GAP$ **:**

- **Given any instance of** PP**:** t, (a_j), b **(Size:** t**)**

1) Define (in time $O(t)$**) an instance** $(m, n, (r_{ij}), (b_i))$ **of** $F\text{-}GAP$**:**

* $n := t$

* $m := 2$; $b_1 := b$; $b_2 := \Sigma_{j=1,t}\, a_j - b$

* $r_{1j} := a_j$; $r_{2j} := a_j$ $(j = 1, \ldots, n)$.

2) Determine (if it exists) a feasible solution (x_{1j}, x_{2j}) **of** $F\text{-}GAP$.

3) If a feasible solution of $F\text{-}GAP$ **exists, then** PP **has a feasible solution: select value** a_j **iff** $x_{1j} = 1$

Otherwise: PP **has no feasible solution.**

Computing time $O(n)$ **(hence** $O(t)$**, polynomial in the size of** PP**).**

***** $F\text{-}GAP$ **is NP-Hard**

Bin Packing Problem (BPP)

Given:

n items;

W_j weight of item j ($j = 1, \ldots, n$) ($W_j > 0$); ***

m containers (bins), each with capacity C:

insert *all the n items in the containers in order to minimize the number of used containers, and in such a way that the sum of the weights of the items inserted in a container is not greater than the capacity C.*

$W_j < C \quad j = 1, \ldots, n$

$\sum_{j=1,n} W_j > C$

Bin Packing Problem (BPP)

Given:

n **items;**

W_j **weight of item** j **(** $j = 1, \ldots, n$ **)** **(**$W_j > 0$**);**

m **containers (bins), each with capacity** C**:**

insert ***all the n items in the containers in order to minimize the number of used containers, and in such a way that the sum of the weights of the items inserted in a container is not greater than the capacity C.***

BPP is NP-Hard

The ***Feasibility Problem*** **of** ***BPP*** **is NP-Hard**

Mathematical Model of BPP

$$x_{ij} = \begin{cases} 1 & \text{if item } j \text{ is inserted in container } i \\ 0 & \text{otherwise} \end{cases} \qquad (i = 1, \ldots, m;\ j = 1, \ldots, n)$$

(**)

$$y_i = \begin{cases} 1 & \text{if container } i \text{ is used} \\ 0 & \text{otherwise} \end{cases} \qquad (i = 1, \ldots, m)$$

Mathematical Model of BPP (2)

(M1) $\min \sum_{i=1,m} y_i$

$$\sum_{j=1,n} W_j x_{ij} \le C \qquad (i = 1, \ldots, m)$$

$$\sum_{i=1,m} x_{ij} = 1 \qquad (j = 1, \ldots, n)$$

$$x_{ij} \le y_i \qquad (i = 1, \ldots, m;\ j = 1, \ldots, n)$$

$$y_i \in \{0, 1\} \qquad (i = 1, \ldots, m)$$

$$x_{ij} \in \{0, 1\} \qquad (i = 1, \ldots, m;\ j = 1, \ldots, n)$$

BLP Model

Feasibility Problem of BPP (F-BPP)

Given: n items; m bins (each with capacity C);

W_j weight of item j ($j = 1, \ldots, n$):

insert *all the n items in the m bins in such a way that the global weight of the items inserted in a bin is not greater than the capacity C.*

F-BPP is NP-Hard

- **Input: $n, m, C, (W_j)$ ($j = 1, \ldots, n$); Size: $3 + n$: n**
- ***Decision Tree*: *n levels* (one for each item *j*);**

**** m descendent nodes (insert item j in bin 1, or 2, …, or m) and constant time for each node ($m \leq n$): ******

***F-BPP* $\in$ *Class NP* ;**

Also *BPP* $\in$ *Class NP* (same Size and *Decision Tree* as *F-BPP*);

(*BLP model* with ($m * n + m$) binary variables x_{ij}, y_i)

F-BBP is NP-Hard

Given: n **items;** m **bins (each with capacity** C**);**

W_j **weight of item** j ($j = 1, \ldots, n$)**:**

insert *all the n items in the m bins in such a way that the global weight of the items inserted in a bin is not greater than the capacity C.*

- ***PP*** $\propto$ ***F-BPP* :**
- **Given any instance of *PP*:** t, (a_j), b ***with*** $b = \Sigma_{j=1,\,t}\ a_j\ /\ 2$ **(Size:** t**)**

1) Define (in time $O(t)$**) an instance** $(n, (W_j), m, C)$ **of *F-BPP*:**

    * $n := t$

    * $C := b$

    * $m := 2$

    * $W_j := a_j$ ($j = 1, \ldots, n$).

2) Determine (if it exists) a feasible solution (x_{ij}) **of *F-BPP.***

3) If a feasible solution (x_{1j}, x_{2j}) **of *F-BPP* exists, then *PP* has a feasible solution** (x_{1j})

Otherwise: *PP* has no feasible solution.

Computing time $O(n)$ **(hence** $O(t)$**, polynomial in the size of *PP*)**

F-BPP is a particular case of F-GAP

***F-GAP*: given: *n* *jobs* and *m* *machines*:**

r_{ij} *amount of resource utilized* for assigning job j to machine i (i = 1, …, m; j = 1, …, n);

b_i amount of resource available for machine i (i = 1, …, m):

assign each job to a machine so that the global resource utilized by each machine i is not greater than the available resource b_i.

***F-BPP*: given: *n* items; *m* bins (each with capacity *C*);**

W_j weight of item j (j = 1, …, n):

insert *all the n items in the m bins so that the global weight of the items inserted in a bin is not greater than the capacity C.*

Arising when:

r_{ij} := W_j (i = 1, …, m; j = 1, …, n);

b_i := b (i = 1, …, m)

Mathematical Model of BPP (2)

(M1) $\min \sum_{i=1,m} y_i$

$\sum_{j=1,n} w_j x_{ij} \le C$ $\quad (i = 1, \ldots, m)$

$\sum_{i=1,m} x_{ij} = 1$ $\quad (j = 1, \ldots, n)$

$x_{ij} \le y_i$ $\quad (i = 1, \ldots, m; j = 1, \ldots, n)$

$y_i \in \{0, 1\}$ $\quad (i = 1, \ldots, m)$

$x_{ij} \in \{0, 1\}$ $\quad (i = 1, \ldots, m; j = 1, \ldots, n)$

$(m + n + m\,n)$ constraints

Alternative Models of BPP

(M2) $\min \sum_{i=1,m} y_i$

$\sum_{j=1,n} w_j \, x_{ij} \le C \qquad (i = 1, \ldots, m)$

$\sum_{i=1,m} x_{ij} = 1 \qquad (j = 1, \ldots, n)$

$\sum_{j=1,n} x_{ij} \le M \, y_i \qquad (i = 1, \ldots, m) \quad M \ge n$

$y_i \in \{0, 1\} \qquad (i = 1, \ldots, m)$

$x_{ij} \in \{0, 1\} \qquad (i = 1, \ldots, m; \; j = 1, \ldots, n)$

$(2m + n)$ constraints

Alternative Models of BPP (2)

(M3) $\quad \min \sum_{i=1,m} y_i$

$$\sum_{j=1,n} W_j x_{ij} \le C\, y_i \qquad (i = 1, \ldots, m)$$

$$\sum_{i=1,m} x_{ij} = 1 \qquad (j = 1, \ldots, n)$$

$$y_i \in \{0, 1\} \qquad (i = 1, \ldots, m)$$

$$x_{ij} \in \{0, 1\} \qquad (i = 1, \ldots, m;\ j = 1, \ldots, n)$$

$(m + n)$ constraints

Alternative Models of BPP (3)

(M1) $\Sigma_{j=1,n} W_j\, x_{ij} \le C$ $(i = 1, \ldots, m)$

$x_{ij} \le y_i$ $(i = 1, \ldots, m;\ j = 1, \ldots, n)$

(M2) $\Sigma_{j=1,n} W_j\, x_{ij} \le C$ $(i = 1, \ldots, m)$

$\Sigma_{j=1,n}\ x_{ij} \le M\ y_i$ $(i = 1, \ldots, m)$ $M \ge n$

(M3) $\Sigma_{j=1,n} W_j\, x_{ij} \le C\ y_i$ $(i = 1, \ldots, m)$

- ***EXAMPLE:*** $C = 100$, $W_1 = 50$, $n = 1000, \ldots$
- ***"Linear Relaxation"*** **of the variables** y_i $(0 \le y_i \le 1)$,
- $x_{11} = 1$, $y_1 = 0.5$ $(x_{1j} = 0,\ j = 2, \ldots, n)$:

  (M2) **and** ***(M3)*****: all constraints are satisfied**

  (M1) $i = 1, j = 1$: **constraint** $x_{ij} \le y_i$ $(1 \le 0.5)$ **is not satisfied**

Linear Relaxation of Model (M1)

- ***Lower Bound LB*** **on the value of the optimal solution of *BPP*:**

$LB = \Sigma_{j=1,n} W_j / C \qquad (LB > 1); \qquad k = \lceil \mathrm{LB} \rceil$

* ***"Linear Relaxation"*** **of the variables x_{ij} and y_i:**

$0 \le x_{ij} \le 1, \quad 0 \le y_i \le 1 \qquad (i = 1, \ldots, m;\ j = 1, \ldots, n).$

- **Optimal solution of the *Linear Relaxation of BPP* (*Model M1*):**
- $y_i = 1 / LB = C / \Sigma_{j=1,n} W_j \quad (< 1) \qquad i = 1, \ldots, k - 1$
- $y_k = 1 - \Sigma_{i=1, k-1}\, y_i \qquad (0 < y_k \le y_1 < 1)$
- $y_h = 0 \qquad h = k + 1, \ldots, m$
- $x_{ij} = y_i \qquad (0 \le x_{ij} < 1) \qquad i = 1, \ldots, m;\ j = 1, \ldots, n$

Linear Relaxation of Model (M1)

- **Optimal solution of the *Linear Relaxation of BPP* (*Model M1*):**
- $y_i = 1 / LB = C / \Sigma_{j=1,n} W_j \quad (< 1) \qquad i = 1, \ldots, k-1$
- $y_k = 1 - \Sigma_{i=1,\,k-1}\, y_i \qquad (0 < y_k \le y_1 < 1)$
- $y_h = 0 \qquad h = k+1, \ldots, m$
- $x_{ij} = y_i \qquad (0 \le y_k < 1) \qquad i = 1, \ldots, m;\; j = 1, \ldots, n$
- ***Constraints:***

$\Sigma_{j=1,n} W_j\, x_{ij} \le C \qquad (i = 1, \ldots, m)$

$\Sigma_{j=1,n} W_j\, y_i = \Sigma_{j=1,n} W_j / LB = C \qquad (i = 1, \ldots, k-1);$

$\Sigma_{j=1,n} W_j\, y_k < \Sigma_{j=1,n} W_j y_1 = C \qquad (i = k)$

$\Sigma_{j=1,n} W_j\, y_i = 0 < C \qquad (i = k+1, \ldots, m)$

Linear Relaxation of Model (M1)

- **Optimal solution of the *Linear Relaxation of BPP* (*Model M1*):**
- $y_i = 1 / LB = C / \Sigma_{j=1,n} W_j \quad (< 1) \qquad i = 1, \ldots, k - 1$
- $y_k = 1 - \Sigma_{i=1, k-1} y_i \qquad (0 < y_k \leq y_1 < 1)$
- $y_h = 0 \qquad h = k + 1, \ldots, m$
- $x_{ij} = y_i \qquad (0 \leq y_k < 1) \qquad i = 1, \ldots, m; \; j = 1, \ldots, n$
- ***Constraints:***

* $\Sigma_{i=1,m} x_{ij} = 1 \qquad (j = 1, \ldots, n)$

$\Sigma_{i=1,m} y_i = 1 \qquad (j = 1, \ldots, n)$

* $x_{ij} \leq y_i \qquad (i = 1, \ldots, m; \; j = 1, \ldots, n)$

$x_{ij} = y_i \qquad (i = 1, \ldots, m; \; j = 1, \ldots, n)$

Linear Relaxation of Model (M1)

- **Optimal solution of the *Linear Relaxation of BPP* (*Model M1*):**
- $y_i = 1 / LB = C / \Sigma_{j=1,n} W_j \quad (< 1) \qquad i = 1, \ldots, k-1$
- $y_k = 1 - \Sigma_{i=1,\,k-1}\, y_i \qquad (0 < y_k \leq y_1 < 1)$
- $y_h = 0 \qquad h = k+1, \ldots, m$
- $x_{ij} = y_i \qquad (0 \leq y_k < 1) \qquad i = 1, \ldots, m; \; j = 1, \ldots, n$

All the constraints are satisfied: feasible solution!

- ***Objective Function of the LP Relaxation of M1:***

$$(M1) \quad z(LP) = \Sigma_{i=1,m}\, y_i = 1$$ **(useless Lower Bound!)**

Assignment Problem (AP)

Particular case of GAP:

***m* = *n*: *n* *machines* (persons) and *n* *jobs* (tasks):**

c_{ij} cost for assigning job j to machine i ($i = 1, \ldots, n$; $j = 1, \ldots, n$);

$r_{ij} = 1$ *amount of resource utilized for* assigning job j to machine i ($i = 1, \ldots, n$; $j = 1, \ldots, n$);

$b_i = 1$ amount of resource available for machine i ($i = 1, \ldots, n$).

***AP* is a Polynomial Problem solvable in $O(n^3)$ time.**

Mathematical Model of *GAP*

- **Objective function (minimum cost)**

$$\min \quad \Sigma_{i=1,m} \Sigma_{j=1,n} c_{ij} x_{ij}$$

- **One machine assigned to each job:**

$$\Sigma_{i=1,m} x_{ij} = 1 \qquad (j = 1, \ldots, n)$$

- **Resource utilized for each machine:**

$$\Sigma_{j=1,n} r_{ij} x_{ij} \le b_i \qquad (i = 1, \ldots, m)$$

$$x_{ij} \in \{0, 1\} \qquad (i = 1, \ldots, m, \; j = 1, \ldots, n)$$

BLP Model

Mathematical Model of *AP*

- **Objective function (minimum cost)**

  $$\min \quad \Sigma_{i=1,n}\,\Sigma_{j=1,n}\; c_{ij}\; x_{ij}$$

- **One machine assigned to each job:**

  $$\Sigma_{\,i=1,n}\;\; x_{ij} \;=\; 1 \qquad\qquad (\,j = 1, \ldots, n)$$

- **Resource utilized for each machine:**

  $$\Sigma_{\,j=1,n}\;\; x_{ij} \;\leq\; 1 \qquad\qquad (\,i = 1, \ldots, \mathrm{n}) \qquad ***$$

$$x_{ij} \in \{0, 1\} \qquad\qquad (i = 1, \ldots, n,\; j = 1, \ldots, n)$$

BLP Model

Mathematical Model of *AP*

- **Objective function (minimum cost)**

  $$\min \quad \Sigma_{i=1,n} \Sigma_{j=1,n} c_{ij} x_{ij}$$

- **One machine assigned to each job:**

  $$\Sigma_{i=1,n} \; x_{ij} \; = \; 1 \qquad (j = 1, \ldots, n)$$

- **Resource utilized for each machine:**

$$\Sigma_{j=1,n} \; x_{ij} \; \leq \; 1: \quad \Sigma_{j=1,n} \; x_{ij} \; = \; 1 \quad (i = 1, \ldots, \mathrm{n})$$

$$0 \leq \; x_{ij} \leq 1 \qquad (i = 1, \ldots, n, \; j = 1, \ldots, n) \qquad ***$$

Mathematical Model of *AP*

- **Objective function (minimum cost)**

  $$\min \quad \Sigma_{i=1,n} \Sigma_{j=1,n} c_{ij}\, x_{ij}$$

- **One machine assigned to each job:**

  $$\Sigma_{i=1,n}\ x_{ij} = 1 \qquad (j = 1, \ldots, n)$$

- **Resource utilized for each machine:**

  $$\Sigma_{j=1,n}\ x_{ij} = 1 \qquad (i = 1, \ldots, n)$$

  $$x_{ij} \geq 0 \qquad (i = 1, \ldots, n,\ j = 1, \ldots, n)$$

LP Model

The optimal solution of the LP model is INTEGER

(the Coefficient Matrix is "Totally Unimodular")

Maximization Version of *AP* (*Max-AP*)

- **Objective function (maximum "cost")**

  $$\max \quad \Sigma_{i=1,n} \Sigma_{j=1,n} c_{ij} x_{ij}$$

- **One machine assigned to each job:**

  $$\Sigma_{i=1,n} x_{ij} = 1 \qquad (j = 1, \ldots, n)$$

- **Resource utilized for each machine:**

  $$\Sigma_{j=1,n} x_{ij} = 1 \quad (i = 1, \ldots, n)$$

$$x_{ij} \geq 0 \quad (i = 1, \ldots, n, \; j = 1, \ldots, n)$$

LP model

Min-Max Version of *AP* (*Bottleneck AP*)

- **Assume** $c_{ij} \geq 0 \ (i = 1, \ldots, n; \ \ j = 1, \ldots, n);$
- **Objective function (minimum cost of an assignment)**

$$\min \quad z = \mathrm{Max}\{ c_{ij}\, x_{ij} : i = 1, \ldots, n; \ \ j = 1, \ldots, n\}$$

$$\Sigma_{\,i=1,n} \ x_{ij} = 1 \qquad\qquad (j = 1, \ldots, n)$$

$$\Sigma_{\,j=1,n} \ x_{ij} = 1 \qquad\qquad (i = 1, \ldots, n)$$

$$x_{ij} \geq 0 \qquad\qquad (i = 1, \ldots, n, \ j = 1, \ldots, n)$$

$$\min \ z$$

$$z \geq c_{ij}\, x_{ij} \qquad\qquad (i = 1, \ldots, n; \ \ j = 1, \ldots, n)$$

LP Model

Min-Max Version of GAP (*Bottleneck GAP*)

- **Assume** $c_{ij} \geq 0 \ (i = 1, \ldots, m; \ \ j = 1, \ldots, n);$
- **Objective function (minimum cost of an assignment)**

$$\min \quad z = \mathrm{Max}\{c_{ij}\, x_{ij} : i = 1, \ldots, m; \ \ j = 1, \ldots, n\}$$

$$\Sigma_{\, i=1,m} \ x_{ij} = 1 \qquad (j = 1, \ldots, n)$$

$$\Sigma_{\, j=1,n} \ r_{ij}\, x_{ij} \leq b_i \qquad (i = 1, \ldots, m)$$

$$x_{ij} \in \{0, 1\} \qquad (i = 1, \ldots, m, \ j = 1, \ldots, n)$$

$$\min \ z$$

$$z \geq c_{ij}\, x_{ij} \qquad (i = 1, \ldots, m; \ \ j = 1, \ldots, n)$$

BLP Model

Transportation Problem (TP)

Given: ***m origins*** **and** ***n destinations*****:**

a_i **amount of product to be transported from origin** i

($i = 1, \ldots, m$**),** $a_i \geq 0$**;**

b_j **amount of product to be transported to destination** j

($j = 1, \ldots, n$**),** $b_j \geq 0$**;**

c_{ij} **cost for transporting one unit of product from origin** i **to destination** j **(**$i = 1, \ldots, m$**;** $j = 1, \ldots, n$**):**

Determine the amount of product **(**x_{ij}**) to be transported from** ***each origin i*** **(**$i = 1, \ldots, m$**)** ***to each destination j*** **(**$j = 1, \ldots, n$**) so as to** ***minimize the global cost.***

$$\Sigma_{j=1,n}\ b_j = \Sigma_{i=1,m}\ a_i$$

Mathematical Model of *TP*

$$\min \quad \sum_{i=1,m} \sum_{j=1,n} c_{ij}\, x_{ij}$$

$$\sum_{j=1,n} x_{ij} = a_i \qquad (i = 1, \ldots, m)$$

$$\sum_{i=1,m} x_{ij} = b_j \qquad (j = 1, \ldots, n)$$

$$x_{ij} \geq 0 \qquad (i = 1, \ldots, m,\; j = 1, \ldots, n)$$

LP model

***TP* is a polynomial problem**

If a_i and b_j are integer: x_{ij} integer

AP* is a special case of *TP

Set Covering Problem (SCP)

- **Given: a "Binary Matrix" A with m rows e n columns;**

  C_j "cost" of column j ($j = 1, \ldots, n$) ($C_j > 0$)

If $A_{ij} = 1$ ($i = 1, \ldots, m; j = 1, \ldots, n$):

column j "*covers*" row i

row i "*is covered*" by column j

Select a subset of the n columns of A_{ij} so that:

- **the sum of the costs of the selected columns is minimum,**
- **all the m rows are covered at least once by the selected columns**

Set Covering Problem (SCP)

- **Given: a "Binary Matrix" A with m rows e n columns;**

  C_j "cost" of column j ($j = 1, \ldots, n$) ($C_j > 0$) ??

  If $A_{ij} = 1$ ($i = 1, \ldots, m$; $j = 1, \ldots, n$):

  column j "*covers*" row i

  row i "*is covered*" by column j

Select a subset of the n columns of A_{ij} so that:

- **the sum of the costs of the selected columns is minimum,**
- **all the m rows are covered at least once by the selected columns**

Example of *SCP*

n= 8, m= 5;

binary matrix:

i \ j	1	2	3	4	5	6	7	8
1	0	1	0	1	1	0	0	1
2	1	0	0	0	0	0	0	1
3	0	0	1	1	0	1	0	1
4	1	1	0	1	0	1	1	0
5	0	0	1	0	1	0	1	0
costs	5	5	3	6	2	4	5	9

feasible solution: Cost = 5 + 5 + 3 = 13

Example of *SCP*

$i \backslash j$	1	2	3	4	5	6	7	8
1	0	1	0	1	1	0	0	1
2	1	0	0	0	0	0	0	1
3	0	0	1	1	0	1	0	1
4	1	1	0	1	0	1	1	0
5	0	0	1	0	1	0	1	0
costs	5	5	3	6	2	4	5	9

optimal solution: **Cost = 5 +3 + 2 = 10**

Mathematical Model of *SCP*

$$x_j = \begin{cases} 1 & \text{if column } j \text{ is selected} \\ 0 & \text{otherwise} \end{cases} \qquad (j = 1, \ldots, n)$$

$$\min \; \Sigma_{j=1,n} \, C_j \, x_j$$

$$\Sigma_{j=1,n} \, A_{ij} \, x_j \geq 1 \qquad (i = 1, \ldots, m)$$

$$x_j \in \{0, 1\} \qquad (j = 1, \ldots, n)$$

BLP Model

The Feasibility Problem of *SCP* is polynomial.

SCP $\in$ ***Class NP*** **(Binary Decision Tree with *n* levels)**

SCP **is known to be NP-Hard**

Variant: *Set Partitioning* (*SPP*)

Select a subset of the n columns of matrix A_{ij} so that:

- **the sum of the costs of the selected columns is minimum,**
- **all the m rows are covered exactly once by the selected columns.**

$$\min \textstyle\sum_{j=1,n} C_j\, x_j$$

$$\textstyle\sum_{j=1,n} A_{ij}\, x_j = 1 \qquad (i = 1, \ldots, m)$$

$$x_j \in \{0, 1\} \qquad (j = 1, \ldots, n) \qquad \text{BLP model}$$

The *Feasibility Problem of SPP* is known to be NP-Hard

***SPP* is NP-Hard (*SPP* $\in$ *Class NP*: Binary Decision Tree with n levels)**

Variant: *Set Partitioning* (*SPP*)

Select a subset of the n columns of matrix A_{ij} so that:

- **the sum of the costs of the selected columns is minimum,**
- **all the m rows are covered exactly once by the selected columns.**

$$\min \; \Sigma_{j=1,n} \; C_j \; x_j$$

$$\Sigma_{j=1,n} A_{ij} \; x_j \; = \; 1 \qquad (\, i = 1, \ldots, m)$$

$$x_j \; \in \{0, 1\} \qquad\qquad (\, j = 1, \ldots, n) \qquad \text{BLP model}$$

The Feasibility Problem of *SPP* is known to be NP-Hard

***SPP* is NP-Hard**

C_j > 0 ???

Example of *SPP*

optimal solution of *SCP* **Cost = 5 +3 + 2 = 10**

i \ j	1	2	3	4	5	6	7	8	Number of covering columns
1	0	1	0	1	1	0	0	1	1
2	1	0	0	0	0	0	0	1	1
3	0	0	1	1	0	1	0	1	1
4	1	1	0	1	0	1	1	0	1
5	0	0	1	0	1	0	1	0	2
costs	5	5	3	6	2	4	5	9	

infeasible solution of *SPP*

Example of *SPP*

optimal solution of *SCP*, cost = 10

cost(*SPP*) $\geq$ cost(*SCP*)

i \ j	1	2	3	4	5	6	7	8	Number of covering columns
1	0	1	0	1	1	0	0	1	1
2	1	0	0	0	0	0	0	1	1
3	0	0	1	1	0	1	0	1	1
4	1	1	0	1	0	1	1	0	1
5	0	0	1	0	1	0	1	0	1
costs	5	5	3	6	2	4	5	9	

optimal solution of *SPP*: **Cost = 5 + 9 = 14**

Strong Formulation of the *BPP*

Let S = subset of the *n* items corresponding to a *feasible loading* of a bin:

S contained in $\{1, 2, \ldots, n\}$ and such that $\Sigma_{j \in S} W_j \leq C$

P = family of all the feasible subsets $S = \{S_1, S_2, \ldots, S_k\}$

(*k* can grow exponentially with *n*).

A subset S_h ($h = 1, \ldots, k$) is *maximal* if the addition of an item generates an infeasible subset. ***

For $j = 1, 2, \ldots, n$; $h = 1, 2, \ldots, k$:

$A_{jh} = 1$ if item j belongs to feasible subset S_h

$A_{jh} = 0$ otherwise

Strong Formulation of the *BPP* (2)

$$x_h = \begin{cases} 1 & \text{if subset } S_h \text{ is selected} \\ 0 & \text{otherwise} \end{cases} \qquad (h = 1, \ldots, k)$$

$$\min \ \Sigma_{h=1,k} \ x_h$$

$$\Sigma_{h=1,k} \ A_{j\,h} \, x_j = 1 \qquad (j = 1, \ldots, n)$$

$$x_h \ \in \{0, 1\} \qquad (h = 1, \ldots, k)$$

Set Partitioning Formulation (BLP Model)

$$\Sigma_{h=1,k} \ A_{j\,h} \, x_j \geq 1 \qquad (j = 1, \ldots, n)$$

Set Covering Formulation (BLP Model)

Strong Formulation of the *BPP* (2)

$$x_h = \begin{cases} 1 & \text{if subset } S_h \text{ is selected} \\ 0 & \text{otherwise} \end{cases} \qquad (h = 1, \ldots, k)$$

$$\min \ \Sigma_{h=1,k} \ x_j$$

$$\Sigma_{h=1,k} A_{j\,h} x_j \geq 1 \qquad (j = 1, \ldots, n)$$

$$x_h \ \in \{0, 1\} \qquad (h = 1, \ldots, k)$$

Set Covering Formulation (BLP Model)

- **If an item is inserted in more than one bin, a feasible solution of the same value can be obtained by using any of these bins for the item.**

Consider only "maximal" feasible subsets S_h

Sequencing of Jobs

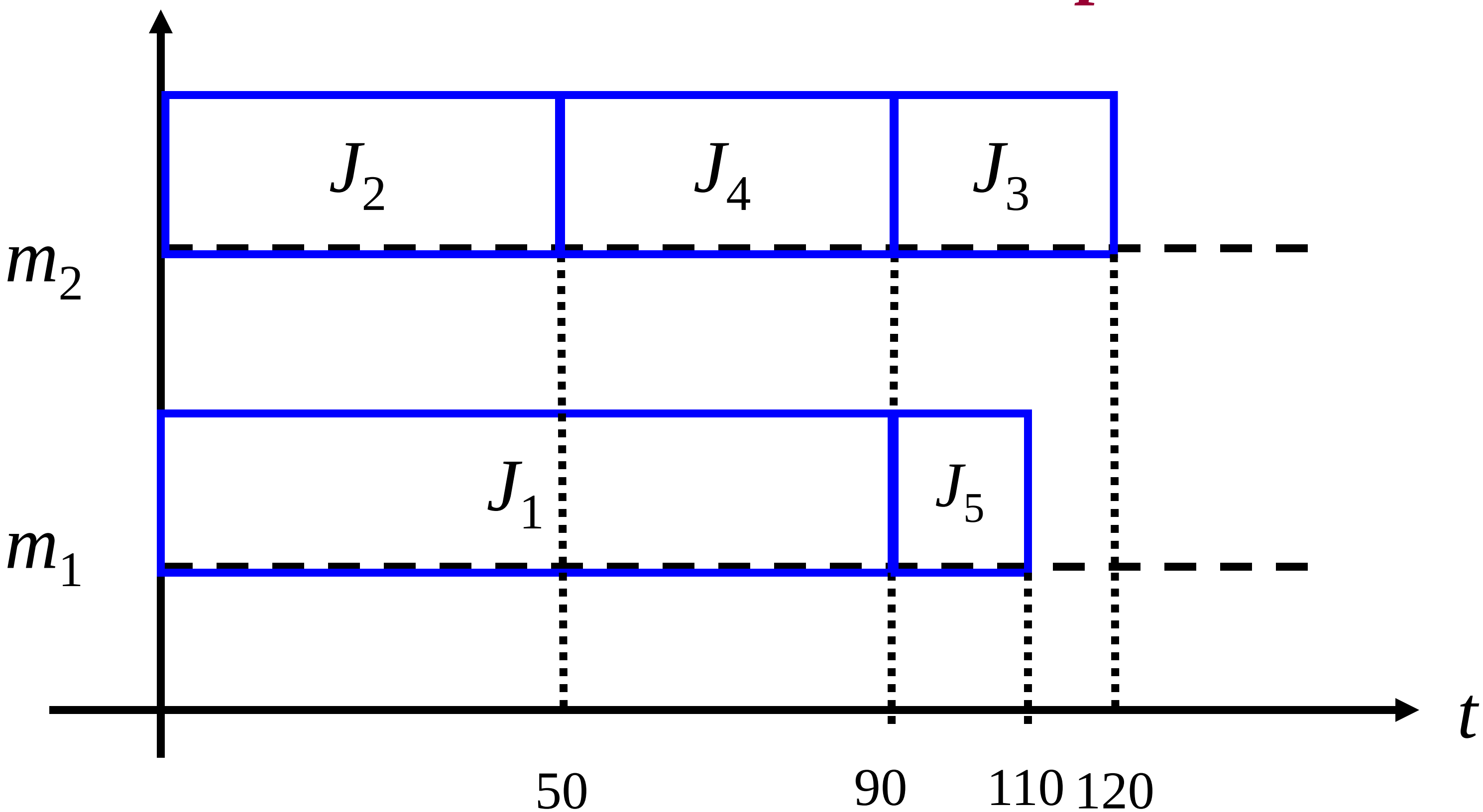
Given:

- ***n jobs***
- ***m identical machines***
- **p_j *processing time* of job j (j = 1, 2, …, n), $p_j > 0$**
- **"*no preemption*" = the processing of a job cannot be interrupted;**
- **a machine cannot process more than one job at the same time;**
- **assign the jobs to the machines so as to minimize the time at which all the machines have finished to process the assigned jobs ("*makespan*").**

Sequencing of Jobs (2)

- ***Example:*** $n = 5$, $m = 2$, $p_j = (90, 50, 30, 40, 20)$

makespan = 120



Sequencing of Jobs (3)

- ***n* = 5**, ***m* = 2**, **p_j = {90, 50, 30, 40, 20}**

- **Z = makespan = 120**

- **LB = Lower Bound =** $\sum_{j=1,n} P_j / m = 230/2 = 115$

- ***Z = value of the optimal solution***
- ***optimal solution : machine 1: jobs 1 and 5***
  machine 2: jobs 2, 3 and 4

Mathematical Model

$$x_{ij} = \begin{cases} 1 & \text{if machine } i \text{ processes job } j \\ 0 & \text{otherwise} \quad (i = 1, \ldots, m;\ j = 1, \ldots, n) \end{cases}$$

$$\min \quad z$$

$$\Sigma_{j=1,n}\ p_j\, x_{ij} \ \le \quad z \qquad (i = 1, \ldots, m)$$

$$\Sigma_{i=1,m}\ x_{ij} \quad = \quad 1 \qquad (j = 1, \ldots, n)$$

$$x_{ij} \in \{0, 1\} \quad (i = 1, \ldots, m;\ j = 1, \ldots, n)$$

$$z \ \ge \ 0$$

MLP model

The Job Sequencing Problem is NP-Hard