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: m, C, k, (P_j), (W_j) ($j = 1, \ldots, n$), N_h ($h = 1, \ldots, k$)**
- **Size: $3 + 2\,n + k * n$ (matrix A_{hj}), with $k \le n$: $n * n$**
- **Size: $3 + 2\,n + n$ (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*:

Multiple Knapsack Problem (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, …, n),

C_i capacity of container i (i = 1, …, 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, …, n), (C_i) (i = 1, …, m)

MKP01 is NP-Hard

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

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

- **Size:** $2 + 2n + m$: $n + m$, ($m \leq n$: **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 $\propto$ ***MCKP***

Bin Packing Problem (*BPP*) 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 bins in order to minimize the number of used bins, and in such a way that the global weight of the items inserted in a bin is not greater than the capacity C.*

- **Input:** $n, m, C, (W_j)$ ($j = 1, \ldots, n$); **Size:** $3 + n$: n
- $m \leq n$

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*** **∝** ***F-BPP*** **:**
- **Given any instance of *PP*:** t, (a_j), b **(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 **) of *F-BPP*.**

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

Otherwise: *PP* has no feasible solution.

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

Generalized Assignment Problem (*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: $2 + 2\, m * n + m$: $m * n$

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* ($i = 1, \dots, m$; $j = 1, \dots, n$);

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

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.

PP ∝ *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*:

*** $n := t$**

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

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

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 (x_{1j} , x_{2j})

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

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)

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

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) $\Sigma_{j=1,\,n}\ W_j\, y_j \;=\; \Sigma_{j=1,\,n}\ W_j\ (\,1 - x_j\,) \;=$

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

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

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

KP01-Min is “equivalent” to *KP01* (3).

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

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

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

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 weight is equal to the knapsack capacity C.**

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

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

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

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

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

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

- **Same input data as for the *KP01*: 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 KP-01:***

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

F-E-KP01 is NP-Hard

- **Same input data as for the *KP01*: $n, C, (W_j)$**
- **Determine a subset of the n items so as to that the global weight is equal to the knapsack capacity C.**
- ***PP* ∝ *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-KP-01.*

3) If a feasible solution (x_j) of *F-E-KP-01* 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 planks with minimization of the waste.

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

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

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

***SSP* is NP-Hard**

SSP is 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 KP-01: 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.

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

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

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

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

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

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

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

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

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

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

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

(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 KP-01:***

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

*** *2C-KP is a generalization of KP-01:***

($KP\text{-}01 \propto 2C\text{-}KP$)

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 ($j = 1, \ldots, n$)

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

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

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

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

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 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 possible number of inserted items of item-type j) *and constant time for each node*:

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

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?**

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

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

* 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 $h = 1, \ldots, k$

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

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

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

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 ($j = 1, \ldots, n$).

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

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

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

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

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 ($j = 1, \ldots, n$)**

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

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

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

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

Set Covering Problem (SCP)

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

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

If $A_{ij} = 1$ (i = 1, …, m; j = 1, …, 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 \ 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 **∈ *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.**

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

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

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

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

***SPP* is NP-Hard**

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*) ≥ cost(*SCP*)

Number of covering columns

i \ j	1	2	3	4	5	6	7	8	
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	

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, …, 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 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)

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

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

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

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

Set Partitioning Formulation (BLP Model)

Consider only "maximal" feasible subsets S_h

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

Set Covering Formulation (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.***

Mathematical Model of *TP*

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

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

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

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

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

Sequencing of Jobs

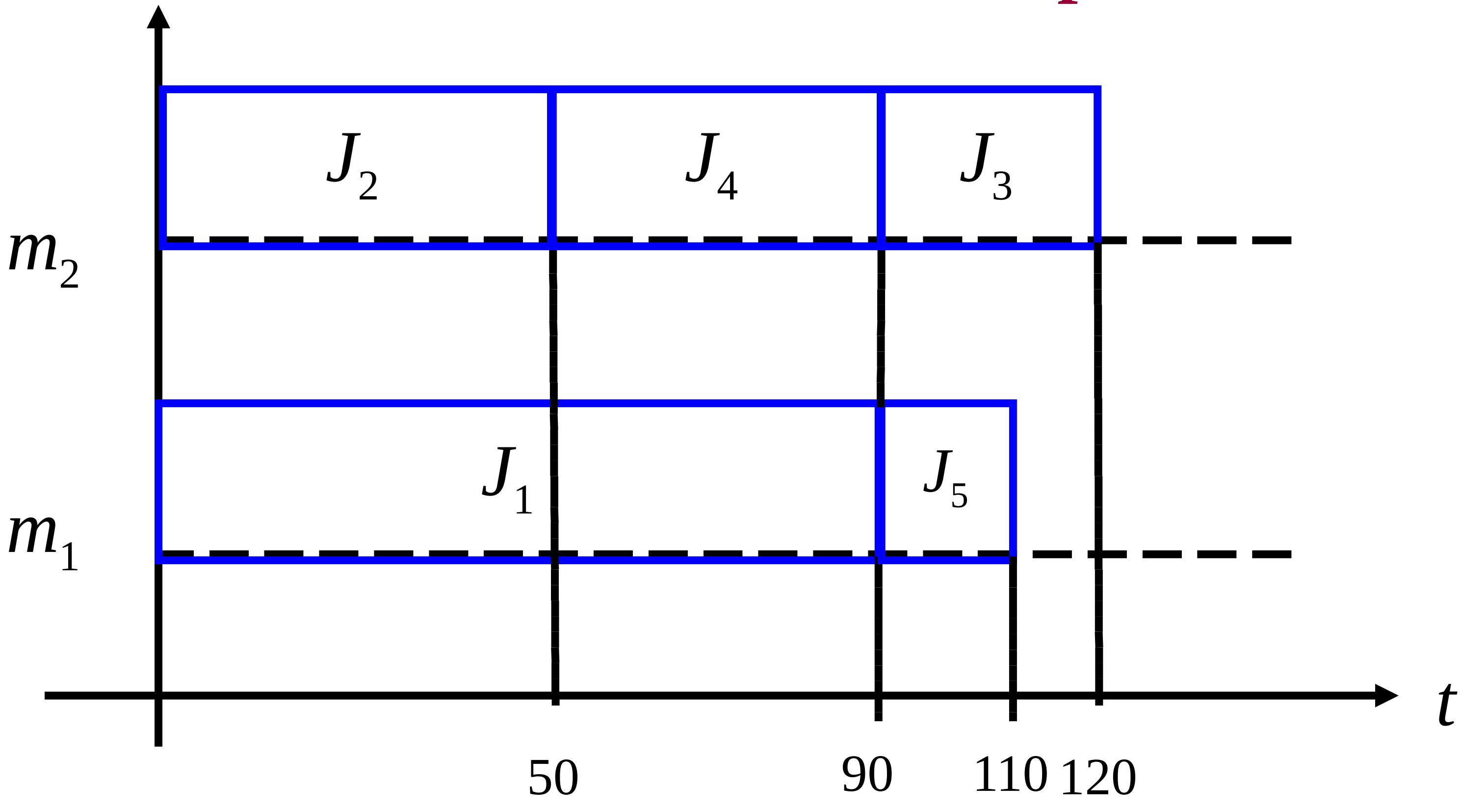
Given:

- ***n*** ***jobs***
- ***m*** ***identical machines***
- p_j ***processing time*** **of job** j **(** $j = 1, 2, \ldots, n$ **),** $p_j > 0$
- **"*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 $\quad z$

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

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

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

$$z \ge 0$$

MLP model

The Job Sequencing Problem is NP-Hard