Algorithms for *SCP* (*Set Covering Problem*)

- **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 matrix A so that:

1) the sum of the costs of the selected columns is minimum,

2) all the m rows are covered at least once by the selected columns.

*** *SCP* is NP-Hard**

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**

Example of *SCP*

Define: $I(j) = \{ i: A_{ij} = 1, i = 1, \ldots, m\}$ (for $j = 1, \ldots, n$): subset of rows covered by column j

$I(1) = \{2, 4\}$; $I(2) = \{1, 4\}$; …; $I(8) = \{1, 2, 3\}$.

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

Example of *SCP*

Define: $J(i) = \{ j : A_{ij} = 1 , j = 1, \ldots, n\}$ (for $i = 1, \ldots, m$):
subset of columns covering row i

$J(1) = \{2, 4, 5, 8\}$; $J(2) = \{1, 8\}$; …; $J(5) = \{3, 5, 7\}$.

$$\begin{array}{c|cccccccc} i \backslash j & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 3 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 4 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 5 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{array}$$

$q = \Sigma_{i=1,m} \Sigma_{j=1,n} A_{ij} = \Sigma_{i=1,m} | J(i) | = \Sigma_{j=1,n} | I(j) |$ ($q << m * n$)

number of elements equal to "1" in matrix (A_{ij})

Mathematical BLP Model of *SCP*

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

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

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

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

The Feasibility Problem of *SCP* is polynomial.

Mathematical BLP Model of *SCP*

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

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

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

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

Constraints (**) can be replaced by constraints:

$$\Sigma_{j \, \in \, J(i)} \, x_j \geq 1 \qquad (i = 1, \ldots, m)$$

Applications of *SCP*

* Crew Scheduling

* Location of Emergency Units

* Vehicle Routing

* Ship Scheduling

* Assembly Line Balancing

* Simplification of Boolean Expressions

* Calculation of Bounds in ILP Models

* Information Retrieval

* Political Districting

* Loading Problems

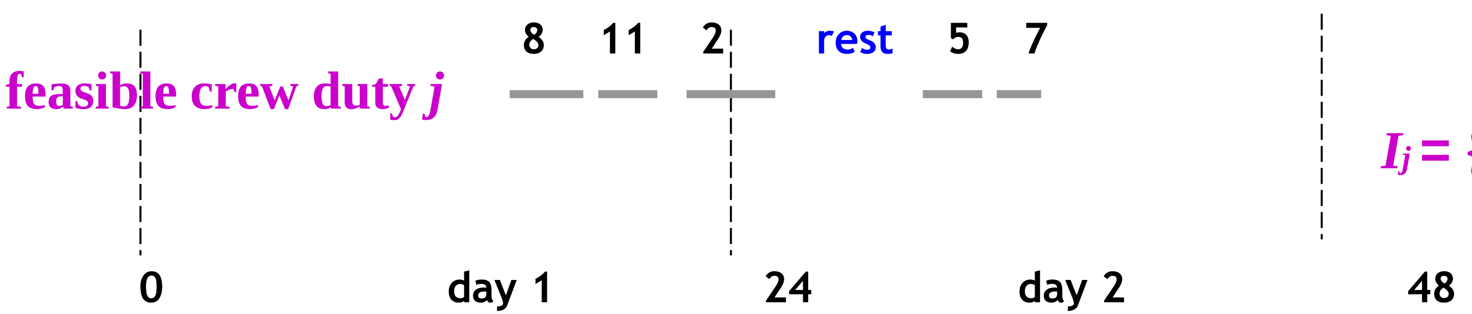
* Vertex Coloring

• • •

Railway Crew Scheduling Application of *SCP*

Given a set of timetabled train trips, find a min-cost set of crew duties so as to cover all the train trips.

- **column j of matrx A** ⟷ **feasible crew duty j**
- **cost C_j** ⟷ **cost of duty j**
- **row of matrix A** ⟷ **trip to be "covered"**



$I_j = \{2, 5, 7, 8, 11\}$

feasible duty j: $I_j = \{ 2, 5, 7, 8, 11\}$

column j of matrix A

$$
\begin{array}{r|c}
1 & 0 \\
2 & 1 \\
3 & 0 \\
4 & 0 \\
5 & 1 \\
6 & 0 \\
7 & 1 \\
8 & 1 \\
9 & 0 \\
10 & 0 \\
11 & 1
\end{array}
$$

VERY LARGE-SCALE INSTANCES

More than 5,000 rows (trips)

and 1,000,000 columns (crew duties)

(Trenitalia: Italian Railway Company).

*** Very sparse A matrices:**

at most 10 trips in each feasible duty

- **In Railway Applications, a crew can travel as a passenger on some trips at no extra-cost (main difference with respect to Airline Applications).**

- **Crew Scheduling Problem solved as a SET COVERING PROBLEM ("overcovered" trips).**

- **Only inclusion-maximal feasible duties, among those with the same cost, have to be generated.**

Exact Algorithms for *SCP*

1) Implicit Enumeration Algorithms

* **Pierce (*Management Science* 1968)**
* **Bellmore-Ratliff (*Management Science* 1971)**
* **Pierce-Lasky (*Management Science* 1973)**

2) Branch-and-Bound Algorithms

* **Lemke-Salkin-Spielberg (*Operations Research* 1971)**
* **Glover (*Operations Research* 1971)**
* **Christofides- Korman (*Management Science* 1975)**
* **Etcheberry (*Operations Research* 1977)**
* **Balas-Ho (*Mathematical Programming* 1980)**
* **Beasley-Jornsten (*European Journal of Operational Research* 1982)**
* **Fisher-Kedia (*Management Science* 1990)**
* **Balas-Carrera (*Operations Research* 1996)**

…

Lower Bounds for *SCP : LP Relaxation*

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

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

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

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

Constraints (**) can be replaced by constraints:

$$\Sigma_{j \in J(i)} \, x_j \geq 1 \qquad (i = 1, \ldots, m)$$

*** No specialized algorithm exists for finding the optimal solution of the *LP Relaxation* of *SCP*.**

Lagrangian Relaxation of SCP

* *Lagrangian Multipliers* (u_i) $(i = 1, \ldots, m)$ with $u_i \geq 0$

$$LB(u) = \min \left(\sum_{j=1,n} C_j x_j + \sum_{i=1,m} u_i \left(1 - \sum_{j=1,n} A_{ij} x_j\right) \right) \leq z(SCP)$$

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

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

* $LB(u) = \sum_{i=1,m} u_i + \min \sum_{j=1,n} C(u)_j x_j$
* with $C(u)_j = C_j - \sum_{i=1,m} u_i A_{ij} \qquad (j = 1, \ldots, n)$

$C(u)_j$: *Lagrangian Cost* of column j

* $O(n * m)$ time for computing all the Lagrangian Costs.

Lagrangian Relaxation of *SCP* (2)

* *Lagrangian Multipliers* (u_i) ($i = 1, \ldots, m$) with $u_i \geq 0$

* $LB(u) = \Sigma_{i=1,m} u_i + \min \Sigma_{j=1,n} C(u)_j \, x_j$

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

* with $C(u)_j = C_j - \Sigma_{i=1,m} u_i A_{ij} = C_j - \Sigma_{i \in I(j)} u_i$

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

* $O(q)$ time for computing all the Lagrangian Costs.

* Optimal Solution (x_j) of the *Lagrangian Relaxation*:

$x_j = 1$ if $C(u)_j \leq 0$, $x_j = 0$ otherwise $(j = 1, \ldots, n)$

$O(n)$ time

* $O(q)$ time for computing $LB(u)$.

Lagrangian Relaxation of *SCP* (3)

* *Lagrangian Multipliers* (u_i) $(i = 1, \ldots, m)$ with $u_i \geq 0$

* $LB(u) = \Sigma_{i=1,m} u_i + \min \Sigma_{j=1,n} C(u)_j x_j$

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

* with $C(u)_j = C_j - \Sigma_{i \in I(j)} u_i$ $(j = 1, \ldots, n)$

* $x_j = 1$ if $C(u)_j \leq 0$, $x_j = 0$ otherwise $(j = 1, \ldots, n)$

* $O(q)$ time for computing $LB(u)$.

* *Lagrangian Dual Problem*:

determine the optimal array of Lagrangian Multipliers (u^*_i) so that: $LB(u^*) = \max \{LB(u): u \geq 0\}$

* It can be proved that $LB(u^*) \leq LBC$

Subgradient Optimization Procedure for the Lagrangian Relaxation of *SCP*

* **Etcheberry (*Operations Research* 1977)**

* ***Subgradient Vector*** $S(u)$ **:**

$$S_i(u) = 1 - \Sigma_{j \in J(i)} x(u)_j \qquad (i = 1, \ldots, m)$$

with $u_i \geq 0$

* $LB(u) = \Sigma_{i=1,m} u_i + \min \Sigma_{j=1,n} C(u)_j x_j$

* ***Input parameters*: *n, m, C, A; LB, UB,*** (u_i)**; *r, t, e, d, Kmax***

* ***Output parameters*: *LB improved,*** (u_i) ***improved;***

Subgradient Optimization Procedure SCP

* $S_i(u) = 1 - \Sigma_{j \in J(i)} x(u)_j \quad (i = 1, \ldots, m)$ with $u_i \geq 0$

* $C(u)_j = C_j - \Sigma_{i \in I(j)} u_i \quad (j = 1, \ldots, n)$; $LB(u) = \Sigma_{i=1,m} u_i + \min \Sigma_{j=1,n} C(u)_j x_j$

* $x_j = 1$ if $C(u)_j \leq 0$, $x_j = 0$ otherwise $(j = 1, \ldots, n)$

$k := 1$

while $UB > LB$ ***do***

$LB(u) := \Sigma_{i=1,m} u_i$; ***for*** i ***to*** m ***do*** $S_i(u) = 1$;

for $j := 1$ ***to*** n ***do***

$C(u)_j = C_j - \Sigma_{i \in I(j)} u_i$;

if $C(u)_j \leq 0$ ***then*** $x(u)_j := 1$; $LB(u) := LB(u) + C(u)_j$;

for $i \in I(j)$ ***do*** $S_i(u) := S_i(u) - 1$

else $x(u)_j := 0$;

$LB := \max \{LB, LB(u)\}$; $k := k + 1$; ***if*** $k > Kmax$ ***then*** *STOP*;

if LB unchanged for t iterations ***then*** $r := r / 2$;

$h := r * (UB - LB) / \| S(u) \|^2$ (step length h);

if $(h < e)$ ***or*** $\| S(u) \|^2 < d$ ***then*** *STOP*;

for i ***to*** m ***do*** $u_i := \max \{0, u_i + h * S_i(u)\}$

endwhile ($O(q)$ time for each iteration of the ***while*** loop)

Surrogate Relaxation for SCP

* Lorena, Belo-Lopez (*Eur. J. Operational Research* 1994)

* *Surrogate Multipliers* (v_i) $(i = 1, \ldots, m)$ with $v_i \geq 0$

$$LBS(v) = \min \Sigma_{j=1,n} C_j x_j$$

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

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

* $$\Sigma_{j=1,n} W(v)_j \, x_j \geq B \qquad (**)$$

* with $W(v)_j = \Sigma_{i=1,m} v_i A_{ij} = \Sigma_{i \in I(j)} v_i \qquad (j = 1, \ldots, n)$

$$B = \Sigma_{i=1,m} v_i$$

* $O(q)$ time for computing all the Surrogate Weights $W(v)_j$.

Surrogate Relaxation for SCP (2)

* *Surrogate Multipliers* (v_i) ($i = 1, \ldots, m$) with $v_i \geq 0$

$$LBS(v) = \min \Sigma_{j=1,n} C_j x_j$$

$$\Sigma_{j=1,n} W(v)_j \; x_j \geq B \qquad (**)$$

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

* with $W(v)_j = \Sigma_{i \in I(j)} v_i \qquad (j = 1, \ldots, n)$; $B = \Sigma_{i=1,m} v_i$

* $O(q)$ time for computing all the Surrogate Weights $W(v)_j$

* The computation of $LBS(v)$ requires the solution of a *Min-KP01* (*NP-Hard* problem).

* The corresponding *LP Relaxation* can be solved in $O(log(n))$ time (Dantzig), or in $O(n)$ time (Balas-Zemel)

Reduction Procedure for SCP

*** Try to fix at their optimal value (0 or 1) as many variables (x_j) as possible.**

*** Partition the column set $N = \{1, 2, \ldots, n\}$ into three subsets $N0$, $N1$ and F, so that any feasible solution (x^*_j) of value smaller than a given Upper Bound UB (corresponding to a feasible solution (x'_j)) must have:**

*** $x^*_j = 0$ for $j \in N0$, $x^*_j = 1$ for $j \in N1$**

1) For $j = 1, \ldots, n$ compute:

$L0(j)$ = *Lower Bound* on $z(SCP)$ by imposing $x_j = 0$;

2) For $j = 1, \ldots, n$ compute:

$L1(j)$ = *Lower Bound* on $z(SCP)$ by imposing $x_j = 1$.

3) Define: $N0 = \{ j : L1(j) \geq UB\}$; $N1 = \{ j : L0(j) \geq UB\}$;

$F = N \setminus N0 \setminus N1$