Algorithms for the 0-1 Knapsack Problem (*KP01*)

***KP01*: given:**

n items,

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

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

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

"Determine a subset of the n items so as to maximize the global profit, and such that the global weight is not larger than the knapsack capacity C."

- ***KP01* is NP-Hard.**

*** Assume (P_j) and (W_j) positive integers.**

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

Branch-and-Bound Algorithms for *KP01*

* Horowitz-Sahni (*Journal of ACM*, 1974).
* Ahrens-Finke (*Operations Research*, 1974).
* Nauss (*Management Science*, 1976).
* Martello-T. (*European Journal of Operational Research*, 1977).
* Balas-Zemel (*Operations Research*, 1980).
* Fayard-Plateau (*Computing*, 1982).
* Martello-T. (*Management Science*, 1988, *Operations Res.* 1997).
* Pandit – Ravi Kumar (*Opsearch*, 1993).
* Pisinger (*Operations Research*, 1997).
* Martello-Pisinger-T. (*Management Science*, 1999).

Dynamic Programming Algorithms for *KP01*

* Bellman (*Dynamic Programming Book*, 1957).
* Horowitz-Sahni (*Journal of ACM*, 1974).
* T. (*Computing*, 1980).

ILP Model *KP01*

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

$$z(KP01) = \max \sum_{j=1,n} P_j \, x_j$$

$$\sum_{j=1,n} W_j \, x_j \le C \qquad (**)$$

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

*** Relaxations:**

*** Continuous (LP) Relaxation.**

*** Lagrangian Relaxation of the “Capacity Constraint (**)**

LP Relaxation of *KP01*

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

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

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

$$0 \leq x_j \leq 1 \qquad (j = 1, \ldots, n)$$

LP Relaxation of *KP01*: Dantzig Upper Bound

1) Assume:

$P_j / W_j \geq P_{j+1} / W_{j+1}$ for $j = 1, \ldots, n - 1$

2) Define the "*critical item*" *s* such that:

$s = \min \{ k : \sum_{j=1,k} W_j > C \}$

3) Optimal LP solution:

$x_j = 1$ for $j = 1, \ldots, s - 1$; $x_j = 0$ for $j = s + 1, \ldots, n$;

$x_s = (C - \sum_{j=1, s-1} W_j) / W_s \quad (0 \leq x_s < 1)$

$UB_D = [\sum_{j=1, s-1} P_j + (C - \sum_{j=1, s-1} W_j) P_s / W_s]$

Dantzig Upper Bound (2)

1) $P_j / W_j \geq P_{j+1} / W_{j+1}$ for $j = 1, \ldots, n - 1$

2) $s = \min \{ j : \sum_{i=1,j} W_j > C \}$

3) $x_j = 1$ for $j = 1, \ldots, s - 1$; $x_j = 0$ for $j = s + 1, \ldots, n$;

$$x_s = (C - \sum_{j=1,s-1} W_j) / W_s$$

$$UB_D = \left[\sum_{j=1,s-1} P_j + (C - \sum_{j=1,s-1} W_j) P_s / W_s \right]$$

- **At most one non-integer variable (x_s).**
- **Computation of UB_D in $O(n)$ time, once s is known;**
- **Computation of s in $O(n \log(n))$ time (Sorting Proc.),**

  in $O(n)$ time through the "partitioning" procedure proposed by Balas-Zemel (*Operations Research*, 1980)

Dantzig Upper Bound (3)

1) $P_j / W_j \geq P_{j+1} / W_{j+1}$ for $j = 1, \ldots, n - 1$

2) $s = \min \{ j : \Sigma_{i=1,j} W_j > C \}$

3) $x_j = 1$ for $j = 1, \ldots, s - 1$; $x_j = 0$ for $j = s + 1, \ldots, n$;

$$x_s = (C - \Sigma_{j=1, s-1} W_j) / W_s$$

$$UB_D = [\Sigma_{j=1, s-1} P_j + (C - \Sigma_{j=1, s-1} W_j) P_s / W_s]$$

*Example:

$n = 7$; $C = 100$; $(P_j) = (100, 90, 60, 40, 15, 10, 10)$;

$(W_j) = (20, 20, 30, 40, 30, 60, 70)$.

$s = 4$; $x_1 = x_2 = x_3 = 1$; $x_4 = 30/40$; $x_5 = x_6 = x_7 = 0$.

$UB_D = [100 + 90 + 60 + 30 * 40 / 40] = 280$ ($z^* = 265$).

Balas-Zemel Procedure (*O.R.*, 1980): Finding the Critical Item in *O(n)* time

1) For each j ∈ $N = \{ 1, \ldots, n \}$ define $r_j = P_j / W_j$.

2) The "critical ratio" r_s can be identified by determining a "partition" of N into subsets $J1, JC, J0$:

$$r_j > r_s \text{ for } j \in J1$$

$$r_j = r_s \text{ for } j \in JC$$

$$r_j < r_s \text{ for } j \in J0$$

with $\Sigma_{j \text{ in } J1} W_j \leq C < \Sigma_{j \text{ in } J1 \text{ union } JC} W_j$

* Progressively determine *J1* and *J0* using, at each iteration, a tentative value *U* for r_s to partition the subset of the currently "free" items in $N \setminus (J1$ *union* $J0)$: U = "*median*" of (r_j) (with j in $N \setminus \{J1$ *union* $J0\}$).

* Given the subsets *J1*, *JC* and *J0*, the critical item *s* is determined by filling, in any order, the "residual capacity" ($C - \Sigma_{j \text{ in } J1} W_j$) with items in subset *JC*.

Lagrangian Relaxation of *KP01*

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

$$z(KP01) = \max \sum_{j=1,n} P_j x_j$$

$$\sum_{j=1,n} W_j x_j \leq C \qquad (**)$$

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

Lagrangian Relaxation of inequality (**), with $v \geq 0$:

$$UB(v) = \max \left(\sum_{j=1,n} P_j x_j + v \left(C - \sum_{j=1,n} W_j x_j \right) \right)$$

Lagrangian Relaxation of *KP01*

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

Lagrangian Relaxation of inequality (**), with $v \geq 0$:

$$UB(v) = (\max \Sigma_{j=1,n} P_j x_j + v\,(C - \Sigma_{j=1,n} W_j x_j))$$

$$UB(v) = v\,C + \max \Sigma_{j=1,n} P(v)_j x_j$$

(where $P(v)_j = P_j - v\,W_j$)

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

Lagrangian Relaxation of *KP01*

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

Lagrangian Relaxation of inequality (**), with $v \geq 0$:

$$UB(v) = v\,C + \max \sum_{j=1,n} P(v)_j \, x_j$$

(where $P(v)_j = P_j - v\,W_j$)

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

*** *Optimal Solution* (*O(n)* time):**

- $x_j = 1$ if $P(v)_j > 0$; $x_j = 0$ if $P(v)_j \leq 0$ $(j = 1, \ldots, n)$
- It can be proved that: $UB(v^*) = UB_D$

and that: $v^* = P_s / W_s$ (where *s = critical item*)

Determination of "*good*" Lagrangian multipliers: *Subgradient Optimization Procedure for KP01*

* $UB(v) = v\,C + \max \sum_{j=1,n} P(v)_j\, x_j$ $\quad (P(v)_j = P_j - v\,W_j)$

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

* $x_j = 1$ if $P(v)_j > 0$; $x_j = 0$ if $P(v)_j \leq 0$ $\quad (j = 1, \ldots, n)$

Define: $S(v) = C - \sum_{j=1,n} W_j\, x_j$ ("subgradient element")

Input parameters:

LB = Lower Bound (value of a feasible solution);

$v_0 > 0$; $Kmax$ = max number of iterations; h = "step length" ($h > 0$);

Subgradient Optimization Procedure for KP01 (2)

$k := 1$; $v := v_0$; $UB = \infty$;

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

$UB(v) := v * C$; $S(v) := C$;

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

$P(v)_j = P_j - v * W_j$;

if $P(v)_j \geq 0$ ***then*** $x(v)_j := 1$; $UB(v) := UB(v) + P(v)_j$; $S(v) := S(v) – W_j$

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

$UB := \min \{UB, UB(v)\}$; $k := k + 1$;

if $k > Kmax$ ***then*** $STOP$;

$v := \max \{0, v – h * S(v)\}$

endwhile

Subgradient Optimization Procedure for KP01 (3)

$S(v) = C - \sum_{j=1,n} W_j\, x_j$ **(“subgradient element”)**

Multiplier Updating Formula:

$v := \max\{0,\ v - h * S(v)\}$

*** If $S(v) > 0$ the relaxed constraint is “too satisfied”:**

v must be decreased;

*** If $S(v) < 0$ the relaxed constraint is violated:**

v must be increased;

*** If $S(v) = 0$ the relaxed constraint is exactly satisfied:**

v must not be changed.

Branching Scheme for KP01

* **Assume:** $P_j / W_j \geq P_{j+1} / W_{j+1}$ **for** $j = 1, \ldots, n - 1$

* **At each level** i ($i = 1, \ldots, n$) **consider item** i **and generate two descendent nodes by setting first** $x_i = 1$, **and then** $x_i = 0$.

* **Depth-first branching strategy.**

* **At each node** k, **corresponding to subproblem** P^k **generated at level** (i -1):

$$P(k) = \sum_{j=1,\, i-1} P_j x_j \qquad \text{(profit at node } k\text{)}$$

$$C(k) = C - \sum_{j=1,\, i-1} W_j x_j \qquad \text{(residual capacity at node } k\text{)}$$

Upper Bound for *KP01 at node k*

* At each node *k*, corresponding to subproblem P^k generated at level (*i* -1):

$P(k) = \Sigma_{j=1, i-1} P_j x_j$ (profit at node *k*)

$C(k) = C - \Sigma_{j=1, i-1} W_j x_j$ (residual capacity at node *k*, $C(k) \geq 0$)

* $UB(P^k) = P(k) + UB_D(P^k)$, where:

$$UB_D(P^k) = \max \sum_{j=i,n} P_j y_j$$

$$\sum_{j=i,n} W_j y_j \leq C(k)$$

$$0 \leq y_j \leq 1 \qquad (j = i, \ldots, n)$$

Dantzig Upper Bound **(LP Relaxation of** P^k**)**

Branching Scheme for KP01 (2)

*** At each node k, corresponding to subproblem P^k generated at level (i -1):**

$$P(k) = \Sigma_{j=1,\, i-1} P_j\, x_j \qquad \text{(profit at node } k\text{)}$$

$$C(k) = C - \Sigma_{j=1,\, i-1} W_j\, x_j \qquad \text{(residual capacity at node } k,\ C(k) \geq 0\text{)}$$

*** At the first descendent node (k + 1) (x_i = 1, generated only if $W_j \leq C(k)$):**

$$P^* = P(k) + P_i\,; \quad C^* = C(k) - W_i \ \text{(with } C^* \geq 0\text{)}$$

*** At the second descendent node (k + b) (x_i = 0, always generated):**

$$P^* = P(k)\,; \quad C^* = C(k)$$

Upper Bounds at the descendent nodes

* **At each node k, corresponding to subproblem P^k generated at level (i -1):**

$$P(k) = \Sigma_{j=1,\, i-1} P_j\, x_j \qquad \text{(profit at node } k\text{)}$$

$$C(k) = C - \Sigma_{j=1,\, i-1} W_j\, x_j \qquad \text{(residual capacity at node } k,\ C(k) \geq 0\text{)}$$

* **At node (k + 1) (x_i = 1, generated only if $W_j \leq C(k)$):**

$$P^* = P(k) + P_i; \quad C^* = C(k) - W_i \text{ (with } C^* \geq 0\text{) :}$$

* $UB(P^{k+1}) = UB(P^k)$

* **the new imposed constraint (x_i = 1) is satisfied by the optimal solution of the LP Relaxation determined at node k (*parametric technique:* the *critical item* at node (k + 1) is equal to the *critical item* at node k).**

Upper Bounds at the descendent nodes (2)

* At each node *k*, corresponding to subproblem P^k generated at level (*i* -1):

$P(k) = \Sigma_{j=1,\, i-1} P_j\, x_j$ (profit at node *k*)

$C(k) = C - \Sigma_{j=1,\, i-1} W_j\, x_j$ (residual capacity at node *k*, $C(k) \geq 0$)

* At node ($k + b$) ($x_i = 0$):

$P^* = P(k)$; $C^* = C(k)$ (with $C^* \geq 0$) :

* $UB(P^{k+1}) \leq UB(P^k)$

* the new imposed constraint ($x_i = 0$) is violated by the optimal solution of the LP Relaxation determined at node *k* (*parametric technique:* the *critical item* at node ($k + b$) is greater than or equal to the *critical item* at node *k*).

Reduction Procedure for KP01

*** Partition the item set $N = \{1, 2, \ldots, n\}$ into three subsets $N0$, $N1$ and F, so that any feasible solution (x^*_j) of value greater than a given Lower Bound LB (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 $j = 1, \ldots, s$ **compute:**

$U0(j)$ = ***Upper Bound*** **on** $z(KP01)$ **by imposing** $x_j = 0$;

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

$U1(j)$ = ***Upper Bound*** **on** $z(KP01)$ **by imposing** $x_j = 1$.

3) Define: $N0 = \{j : U1(j) \leq LB\}$; $N1 = \{j : U0(j) \leq LB\}$;

Reduction Procedure for KP01 (2)

*** Partition the item set $N = \{1, 2, \ldots, n\}$ into three subsets $N0$, $N1$ and F, so that any feasible solution (x^*_j) of value greater than a given Lower Bound LB (corresponding to a feasible solution (x'_j)) must have: * $x^*_j = 0$ for $j \in N0$, $x^*_j = 1$ for $j \in N1$**

* *Reduced Problem RD:*

$$z(RD) = \sum_{j \text{ in } N1} P_j + \max \sum_{j \text{ in } F} P_j x_j$$

$$\sum_{j \text{ in } F} W_j x_j \leq C - \sum_{j \text{ in } N1} W_j$$

$$x_j \in \{0, 1\} \qquad j \in F$$

*** The *Reduction Procedure* can be implemented to run in $O(n \log(n))$ time (a “weaker” version in $O(n)$ time).**

Test Instances for KP01

*** Given: *n* and *R*, generate *k* random instances as follows:**

1) *Uncorrelated (UCR) Instances:*

*** W_j integer value randomly generated according the uniform distribution in the interval [1, *R*] ($j = i, \ldots, n$);**

*** P_j integer uniformly random in [1, *R*] ($j = i, \ldots, n$).** *****

$$C = 0.5 \sum_{j=1, n} W_j$$

2) *Weakly Correlated (WCR) Instances:*

*** W_j integer uniformly random in [1, *R*] ($j = i, \ldots, n$);**

*** P_j integer un. rand. in [W_j, W_j + *R/10*] ($j = i, \ldots, n$).**

3) *Strongly Correlated (SCR) Instances:*

*** W_j integer uniformly random in [1, *R*] ($j = i, \ldots, n$);**

Computational Results for the Reduction Procedure for KP01 (3) with *R* = 1000

* Partition the item set *N* into three subsets *N0*, *N1* and *F*

* The global computing times of the Reduction Procedure are about 1.5 times the corresponding sorting times.

* For the UCR instances, the average number of items left in the Reduced Problem (i.e., | *F* |) is about 25 if n = 100, and about 80 if n = 500.

* For the WCR instances, average | *F* | is about 55 if n = 100, and about 180 if n = 500.

* For the SCR instances, average | *F* | is about 90 if n = 100, and about 450 if n = 500.

"Core Problem" Approach for KP01

*** In Large-Size "easy" *KP01* instances: most of the computing time is spent for preliminary sorting of the items according to non-increasing P_j / W_j ratios**

*** If the items are sorted, the Optimal Solution (x^*_j) to a Large-Size *KP01* instance is defined by:**

$$x^*_j = 1 \text{ for } j = 1, \ldots, j_1 - 1 \; ; \; x^*_j = 0 \text{ for } j = j_2 + 1, \ldots, n \, ;$$

$$x^*_j \in \{0, 1\} \text{ for } j = j_1, \ldots, j_2$$ **("Core Problem" *CP*)**

with $j_1 < s < j_2$

*** ($j_2 - j_1$) very small fraction of n (30 to 40 for n = 1000) and slowly increasing with n**

Algorithm MT2 for KP01 (M. - T., Man. Sc. 1988)

1) Find $J1$, JC, $J0$, s without sorting (Balas-Zemel, 1980).

2) Define: j^*_1 and j^*_2

such that $j^*_1 < s < j^*_2$ and $j^*_2 - j^*_1 \geq u$ (u given)

"Approximate Core Problem" *ACP* (*O(n)* time).

3) Sort the items in *ACP* according to non-increasing P_j / W_j ratios.

4) Solve *ACP* through a Branch-and-Bound Algorithm:

$LB = \Sigma_{j \text{ in } J1} P_j + z(ACP)$ (where $z(ACP)$ is the optimal value)

is a valid Lower Bound for *KP01*.

UB = Upper Bound for *KP01 (Improved Dantzig Upper Bound)*.

5) If $LB = UB$ then *STOP* (optimal solution found).

Algorithm MT2 for KP01 (2)

6) Apply the *Reduction Procedure* (version without "*sorting*", *O(n)* time) to *KP01*, and determine subsets *N0*, *N1* and *F*.

7) If $\{1, \ldots, j^*_1 - 1\} \subseteq N1$ and $\{j^*_2 + 1, \ldots, n\} \subseteq N0$

then *STOP* (optimal solution found).

8) Sort the items in *F* according to non-increasing P_j / W_j ratios.

9) Solve the *KP01* corresponding to *F* through a Branch-and-Bound Algorithm.

Computational Results for *KP01*

*** *Algorithm MT2* is able to solve to optimality *UCR* and *WCR* instances with up to 100,000 items in few CPU seconds,**

• but it can fail to determine, within 5-10 minutes, the optimal solution for SCR instances with 100 items.

*** *Dynamic Programming Algorithm DPT* (T. 1980) is able to solve to optimality *UCR* and *WCR* instances with up to 10000 items in 5-10 CPU seconds,**

• but it can fail to determine, within 5-10 CPU minutes, the optimal solution for SCR instances with 1000 items.