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

***KP01*: 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 "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*

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

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

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

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

* Relaxations:

* Continuous (LP) Relaxation.

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

LP Relaxation of *KP01*

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

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

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

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

LP Relaxation of *KP01*: *Dantzig Upper Bound*

1) Assume:

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

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

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

3) Optimal LP solution:

$$x_j = 1 \;\text{ for }\; j = 1, \ldots, s - 1; \quad x_j = 0 \;\text{ for }\; j = s + 1, \ldots, n;$$

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

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

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:**

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

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

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

$UB_D = [100 + 90 + 60 + 30 * 40 / 40] = 280$

$z^* = 265$, $(x^*_j) = (1, 1, 1, 0, 1, 0, 0)$

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

1) For each $j \in N = \{ 1, \ldots, n \}$ 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, $J0$:

$r_j > r_s$ for $j \in J1$

$r_j = r_s$ for $j \in JC$

$r_j < r_s$ 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*

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

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

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

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

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

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

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 $v \geq 0$:

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

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

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

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

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**

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

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

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

if $P(v)_j > 0$ **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)

$$P(v)_j = P_j - v\, W_j \quad (j = 1, \ldots, n)$$

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

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

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

v must be decreased ($P(v)_j$ increases);

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

v must be increased ($P(v)_j$ decreases);

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

v (and $P(v)_j$) 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):**

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

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

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):**

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

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

*** 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):

$P(k) = \Sigma_{j=1,\, i-1} P_j\, x_j$ (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 + 1) ($x_i = 1$, generated only if $W_j \leq C(k)$):

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

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

* 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):**

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

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

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

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

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

*** 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:**

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

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: $N0 = \{\,j : U1(\,j\,) \le LB\}$; $N1 = \{\,j : U0(\,j\,) \le LB\}$;

$F = N \setminus N0 \setminus N1$

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:*

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

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

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

* 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, generate *a set of* random instances as follows:

1) *Uncorrelated (UCR) Instances:*

* W_j integer value randomly generated according to the uniform distribution in the interval [1, 1000] ($j = 1, \ldots, n$);

* P_j integer uniformly random in [1, 1000] ($j = 1, \ldots, n$).

2) *Weakly Correlated (WCR) Instances:*

* W_j integer uniformly random in [1, 1000] ($j = 1, \ldots, n$);

* P_j integer un. rand. in $[W_j, W_j + 100]$ ($j = 1, \ldots, n$).

3) *Strongly Correlated (SCR) Instances:*

* W_j integer uniformly random in [1, 1000] ($j = 1, \ldots, n$);

* $P_j = W_j + 100$ ($j = 1, \ldots, n$).

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

Computational Results for the Reduction Procedure for KP01

*** 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:

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

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

with $\quad j_1 < s < j_2$

* $(j_2 - j_1)$ 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: *j1 and *j2****

such that *j1 < *s* < *j2** and *j2** - *j1** ≥ *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 $J1^* = \{ j : P_j / W_j > P_{j1^*} / W_{j1^*} \}$,**

$z(ACP)$ = optimal value of *ACP*) 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 $\{1, \ldots, j1^* - 1\} \subseteq N1$ and $\{j2^* + 1, \ldots, n\} \subseteq N0$

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.

Difficult *KP01* Instances

*** *Algorithm PR* (Pandit and Ravi-Kumar, 1993) is a "specialized" exact algorithm for *KP01* designed to solve only *SCR instances*:**

*** it is able to solve to optimality *SCR* instances with up to 10,000 items in few CPU minutes,**

*** but it cannot solve *UCR* and *WCR* instances.**

Data Set	*n*	MT2	DPT	PR
UCR	50	0.01	0.02	-
	100	0.01	0.02	-
	500	0.05	0.09	-
	1000	0.09	0.24	-
	10000	0.50	3.73	-
WCR	50	0.01	0.06	-
	100	0.02	0.08	-
	500	0.08	0.49	-
	1000	0.12	1.66	-
	10000	0.31	5.86	-
SCR	50	0.13	0.51	0.01
	100	134.48 (9)	2.03	0.03
	500	642.62 (4)	45.01	0.50
	1000	-	124.78 (5)	1.94
	10000	-	-	207.48

* VAXstation 3100 seconds; Time limit = 2000 seconds;
* Average time over 10 instances (solved instances if < 10)

Data Set	*n*	MT2	DPT	PR	Cplex
UCR	50	0.01	0.02	-	0.12
	100	0.01	0.02	-	0.19
	500	0.05	0.09	-	0.72
	1000	0.09	0.24	-	1.51
	10000	0.50	3.73	-	27.84
WCR	50	0.01	0.06	-	0.14
	100	0.02	0.08	-	0.23
	500	0.08	0.49	-	1.29
	1000	0.12	1.66	-	2.54
	10000	0.31	5.86	-	21.81
SCR	50	0.13	0.51	0.01	4.22
	100	(9)	2.03	0.03	(8)
	500	(4)	45.01	0.50	(4)
	1000	-	(5)	1.94	(1)
	10000	-	-	207.48	-

* **VAXstation 3100 seconds; Time limit = 2000 seconds;**
* **Average time over 10 instances (solved instances if < 10)**

Additional Test Instances for KP01

* Given: n, generate *a set of* random instances as follows

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

4) *Almost Strongly Correlated (ASCR) Instances:*

* W_j integer uniformly random in [1, 1000] ($j = i, \ldots, n$);

* P_j integer un. rand. in [W_j + 99, W_j + 101] ($j = i, \ldots, n$).

5) *Uncorrelated with Large Weights (ULWR) Instances:*

* W_j integer un. rand. in [100,001; 101,000] ($j = i, \ldots, n$);

* P_j integer uniformly random in [1, 1000] ($j = i, \ldots, n$).

* *Algorithm PR* cannot solve *ASCR* and *ULWR* instances.

Data Set	*n*	MT2	DPT	PR
ASCR	50	0.09	0.51	-
	100	(9)	2.11	-
	500	(5)	44.47	-
	1000	-	118.78 (5)	-
	10000	-	-	-
ULWR	50	0.10	(8)	-
	100	0.02	(5)	-
	500	(7)	-	-
	1000	(8)	-	-
	10000	-	-	-

*** VAXstation 3100 seconds; Time limit = 2000 seconds;**

*** Average time over 10 instances (solved instances if < 10)**

- **Better Upper Bounds for *KP01* are needed:**

Data Set	*n*	MT2	DPT	PR
ASCR	50	0.09	0.51	-
	100	(9)	2.11	-
	500	(5)	44.47	-
	1000	-	118.78 (5)	-
	10000	-	-	-
ULWR	50	0.10	(8)	-
	100	0.02	(5)	-
	500	(7)	-	-
	1000	(8)	-	-
	10000	-	-	-

- **Better Upper Bounds for *KP01* are needed:**
  strengthen a relaxed problem *RP* by adding valid inequalities which are redundant for the original problem,
  but could be violated by the optimal solution of *RP*.

Stronger Upper Bound for *KP01 (M.-T. 1997)*

*** Determine:**

Kmax **= maximum number of items in a feasible solution**

*** Sort the items so that** $W_1 \le W_2 \le \ \dots \ \le W_n$

$$Kmax = \min \{ k : \Sigma_{j=1,k} \ W_j > C \} - 1$$

*** Example:**

$n = 6$; $C = 48$; $(P_j) = (15, 16, 19, 17, 19, 23)$;

$(W_j) = (10, 12, 15, 14, 17, 21)$.

* $s = 4$; $UB_D = 15 + 16 + 19 + [11 * 17 / 14] = 63$

* sorted (W_j): $(10, 12, 14, 15, 17, 21)$, $Kmax = 3$.

* **Optimal Solution** $(x_j) = (0, 1, 1, 0, 0, 1)$, $z(KP01) = 58$

Stronger Upper Bound for *KP01* (2)

* Determine:

KMAX = maximum number of items in a feasible solution

* Sort the items so that $W_1 \leq W_2 \leq \ldots \leq W_n$

$$KMAX = \min \{ k : \Sigma_{j=1,k} W_j > C \} - 1$$

* *Equivalent* ILP model for *KP01*:

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

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

$$\Sigma_{j=1,n} x_j \leq KMAX \qquad (**)$$

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

Stronger Upper Bound for *KP01* (3)

* Equivalent ILP model for *KP01*:

$$z(KP01) = \max \quad \Sigma_{j=1,n} \; P_j \, x_j \qquad (1)$$

$$\Sigma_{j=1,n} \; W_j \, x_j \le C \qquad (2)$$

$$\Sigma_{j=1,n} \; x_j \le KMAX \qquad (3)$$

$$x_j \in \{0, 1\} \qquad (j = 1, \ldots, n) \qquad (4)$$

* Lagrangian Relaxation of constraint (3) ($v \ge 0$), and LP Relaxation of the Lagrangian Relaxation:

$$UB(v) = \max \left(\Sigma_{j=1,n} \; P_j \, x_j + v \left(KMAX - \Sigma_{j=1,n} \; x_j\right)\right) \text{ s.t. } (2),(4')$$

$$0 \le x_j \le 1 \qquad (j = 1, \ldots, n) \qquad (4')$$

Stronger Upper Bound for *KP01* (4)

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

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

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

with $P(v)_j = P_j - v \qquad\qquad (\, j = 1, \ldots, n)$

* **The corresponding *Dantzig Upper Bound* is computed.**

* **The *best Lagrangian Multiplier v** (and the corresponding *UB*(*v**)) can be computed in *O(n*n)* time.**

Stronger Upper Bound for *KP01* (5)

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

$$\Sigma_{j=1,n}\; W_j\, x_j \;\le\; C\;; \quad 0 \;\le\; x_j \le\; 1 \qquad (j = 1, \ldots, n)$$

with $P(v)_j = P_j - v \qquad\qquad (j = 1, \ldots, n)$

*** Example:**

$n = 6$; $C = 48$; $(P_j) = (15, 16, 19, 17, 19, 23)$; $z(KP01) = 58$

$(W_j) = (10, 12, 15, 14, 17, 21)$; $Kmax = 3$.

* $v = 0$, $s(0) = 4$; $UB(0) = 15 + 16 + 19 + [11 * 17 / 14] = 63$

* $v = 5$, sorted items $\quad (P(5)_j) = (10, 14, 11, 12, 18, 14)$;

$(W'_j) = (10, \underline{15}, \underline{12}, 14, \underline{21}, \underline{17})$.

$s(5) = 4$; $UB(5) = 5 * 3 + 10 + 14 + 11 + [11 * 12 / 14] = 59$

Stronger Upper Bound for *KP01* (6)

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

$$\Sigma_{j=1,n} \ W_j \ x_j \ \le \ C \ ; \quad 0 \ \le \ x_j \le \ 1 \quad (j = 1, \ldots, n)$$

with $P(v)_j = P_j - v \qquad (j = 1, \ldots, n)$

*** Example:**

$n = 6;\ C = 48;\ (P_j) = (15, 16, 19, 17, 19, 23);\ z(KP01) = 58$

$(W_j) = (10, 12, 15, 14, 17, 21);\ Kmax = 3.$

* $v = 0,\ s(0) = 4,\ UB(0) = 63;$ * $v = 5,\ s(5) = 4,\ UB(5) = 59;$

* $v = 10$, **sorted items** $(P(10)_j) = (13, 9, 9, 5, 6, 7);$

$(W''_j) = (21, 15, 17, 10, 12, 14).$

$s(10) = 3;\ UB(10) = 10 * 3 + 13 + 9 + [12 * 9 / 17] = 58$

Alternative Stronger Upper Bound for *KP01*

* Determine: *KMIN* = minimum number of items in an optimal solution.

* Sort the items so that $P_1 \geq P_2 \geq \ldots \geq P_n$

$$KMIN = \min \{ k : \Sigma_{j=1,k} \; P_j \; > \; LB \}$$

where *LB* (Lower Bound) is the value of a feasible solution.

* Equivalent ILP model for *KP01*:

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

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

$$\Sigma_{j=1,n} \; x_j \geq KMIN$$

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

Algorithm MTH for *KP01* (M.-T., Oper. Res. 1997)

*** At each node of the branch-decision tree:**

a) compute the "stronger upper bound" (or the "alternative stronger upper bound") by using a parametric technique;

b) if the node is not fathomed, apply the Reduction Procedure, and try to fathom the node through Dominance Criteria;

c) try to fathom the node through a "Partial Dynamic Programming" list.

Data Set	*n*	MT2	DPT	PR	MTH
UCR	50	0.01	0.02	-	0.03
	100	0.01	0.02	-	0.04
	500	0.05	0.09	-	0.08
	1000	0.09	0.24	-	0.14
	10000	0.50	3.73	-	1.59
WCR	50	0.01	0.06	-	0.03
	100	0.02	0.08	-	0.04
	500	0.08	0.49	-	0.09
	1000	0.12	1.66	-	0.15
	10000	0.31	5.86	-	1.28
SCR	50	0.13	0.51	0.01	0.10
	100	(9)	2.03	0.03	0.15
	500	(4)	45.01	0.50	0.71
	1000	-	(5)	1.94	1.31
	10000	-	-	207.48	2.31

*** VAXstation 3100 seconds; Time limit = 2000 seconds;**

*** Average time over 10 instances (solved instances if < 10)**

Data Set	*n*	MT2	DPT	PR	MTH
ASCR	50	0.09	0.51	-	0.11
	100	(9)	2.11	-	0.64
	500	(5)	44.47	-	0.78
	1000	-	(5)	-	5.65
	10000	-	-	-	102.83
ULWR	50	0.10	0.02 (8)	-	0.07
	100	0.02	0.06 (5)	-	0.04
	500	(7)	-	-	3.30
	1000	(8)	-	-	0.76
	10000	-	-	-	327.58

* VAXstation 3100 seconds; Time limit = 2000 seconds;
* Average time over 10 instances (solved instances if < 10)