Computing 28, 269—287 (1982)

Computing
© by Springer-Verlag 1982

Algorithm / Algorithmus 47

An Algorithm for the Solution of the 0-1 Knapsack Problem

D. Fayard, Orsay, and **G. Plateau**, Villeneuve D'Ascq

Received November 20, 1980; revised August 5, 1981

Abstract — Zusammenfassung

Algorithm 47. An Algorithm for the Solution of the 0-1 Knapsack Problem. A new implicit enumeration algorithm for the solution of the 0-1 knapsack problem – denoted by FPK 79 – is proposed. The implementation of the associated FORTRAN IV subroutine is then described. Computational results prove the efficiency of this algorithm (practically linear time complexity including the initial arrangement of the data) whose performance is generally better than that of algorithm 37 and thus superior to that of the best known algorithms.

AMS Subject Classification: 90—04, 90 C 08, 90 C 09.
Key words: Binary knapsack, integer programming.

Algorithmus 47. Ein Algorithmus für die Lösung des 0-1 Knapsack Problems. Wir stellen einen neuen Enumerationsalgorithmus – FPK 79 genannt – für die Lösung des 0-1 Knapsack Problems vor. Dann beschreiben wir die zugehörige Fortran IV Subroutine. Die durchgeführten numerischen Versuche zeigen experimentell, daß der Algorithmus einschließlich des Sortierens der Eingangsdaten lineares Zeitverhalten aufweist. Er ist damit leistungsfähiger als der Algorithmus 37 und somit besser als die besten bekannten Algorithmen.

1. Introduction

The improvement of the three phases of the MSB implicit enumeration method described in [2]:

Phase 1: Solving the relaxed problem
Phase 2: Reduction of the size
Phase 3: Implicit enumeration (including a reduction scheme)

explains the greater efficiency of the proposed algorithm – denoted by FPK 79 – for the solution of the 0-1 knapsack problem.

Many options have been analyzed and proved in [6]; the aim of this paper is the description of the algorithm which has been actually implemented (section 3). Directions for use of the FORTRAN subroutine (section 6) are to be found in section 4. Computational results (section 5) show that algorithm FPK 79 is faster than the one of Martello and Toth (see [9]). (Algorithms have been tested both with randomly generated and concrete problems.)

0010-485X/82/0028/0269/$ 03.80

2. Notations

$\lfloor\lambda\rfloor$:	integer part of a real number λ

Given a set J:

$[J]$:	convex hull of J
$\|J\|$:	cardinality of J
$J \backslash U$:	complement of a given subset U of J

Given an optimization problem (P):

$v(P)$:	optimal value of (P)
$\bar{v}(P)$:	(resp. $\underline{v}(P)$) upper (resp. lower) bound on $v(P)$
$F(P)$:	set of feasible solutions for (P)
$(P \| x \in X)$:	(P) with the added constraint $x \in X$ [i.e. $(\exists j : x_j = \varepsilon)$ or $(\exists j_1, j_2 : x_{j_1} = \varepsilon_1 ; x_{j_2} = \varepsilon_2)$ with $\varepsilon, \varepsilon_1, \varepsilon_2 \in \{0,1\}$]

When (P) is a 0-1 problem in n variables

x^*:	optimal solution for (P)
V	$= \{x \in \mathbb{R}^n \| x_j = 0 \text{ or } 1, j = 1, \ldots, n\}$
$(\bar{P})$:	problem (P) when $[V]$ is substituted for V
$\bar{x}$:	optimal solution for $(\bar{P})$

3. Description of the Algorithm

Algorithm FPK 79 solves the following problem

$$(B) \left[\begin{array}{ll} \text{maximize} & cx \\ \text{subject to} & ax \le b \\ & x \in V \end{array} \right.$$

whose data are such that:

$$\left| \begin{array}{l} c, a \in \mathbb{N}_*^n, b \in \mathbb{N}_* \\ \max\limits_{1 \le j \le n} a_j \le b < \sum\limits_{j=1}^{n} a_j \end{array} \right.$$

(These latest assumptions eliminate trivial solutions.)

Note: Only few remarks follow the algorithm; for detailed proofs, see the references included in this description; a simplified flowchart of the algorithm is presented in Fig. 1.

Algorithm FPK 79:

Phase 1: *Solving $(\bar{B})$: Search $U \subset I = \{1, \ldots, n\}$ such that*

$$\left[\begin{array}{l} \sum\limits_{j \in U} a_j \le b < \sum\limits_{j \in U \cup \{i\}} a_j \\ \forall j \in U \; c_j/a_j \ge c_p/a_p \; \forall p \notin U ; \; c_i/a_i = \max \{c_p/a_p \mid p \notin U\} \end{array} \right.$$

in order to define the optimal solution $\bar{x}$ *of* $(\bar{B})$: $\bar{x}_j = 1\ \forall j \in U$; $\bar{x}_i = (b - \sum_{j \in U} a_j)/a_i$; $\bar{x}_j = 0\ \forall j \in L = I \backslash (U \cup \{i\})$.

The following algorithm NKR (expected linear time complexity) is analyzed in [3,4,6] (see also [5]).

```
0  $I \equiv J \equiv \{1, \ldots, n\}$, $U \equiv \emptyset$
1  if $|J| \leq 10$ then
     Apply the following algorithm CKR (see [3, 4]):
     1.1  Use the sorting method called "Insertion Sort" in [10] in order to sort in
          decreasing order the elements of
          $R = \bigcup_{j \in J} \{c_j/a_j\}$.
          It is assumed that the data are renumbered so that
          $c_{j_1}/a_{j_1} \geq c_{j_2}/a_{j_2} \geq \ldots \geq c_{j_q}/a_{j_q}$ where $q = |J|$
     1.2  $k^* \leftarrow \min \left\{ k \mid \sum_{p=1}^{k} a_{j_p} > b - \sum_{j \in U} a_j \right\}$
     1.3  $U \equiv U \cup \{j_1, j_2, \ldots, j_{k^*-1}\}$
          if $\sum_{j \in U} a_j = b$ then $L \equiv I \backslash U$
                              else $i \leftarrow j_{k^*}$; $L \equiv I \backslash (U \cup \{i\})$;
                                   $\bar{x}_i \leftarrow (b - \sum_{j \in U} a_j)/a_i$;
          goto 6
2  $R \equiv \bigcup_{j \in J} \{c_j/a_j\}$
     2.1  Find $k \in J$ such that $c_k/a_k$ is the median element of the three elements located
          at the beginning, the middle and the end of $R$.
     2.2  Construct the following $(R, k)$-partition of $J$:
          $U(J, R, k) \equiv \{j \in J \backslash \{k\} \mid c_j/a_j \geq c_k/a_k\}$;
          $L(J, R, k) \equiv J \backslash (U(J, R, k) \cup \{k\})$
          [elements of $\{j \in J \backslash \{k\} \mid c_j/a_j = c_k/a_k\}$ are distributed in $U(J, R, k)$ and
          $L(J, R, k)$ in order to minimize the number of permutations]
3  if $\sum_{j \in U \cup U(J,R,k)} a_j > b$ then $J \equiv U(J, R, k)$; goto 1
                                  else $U \equiv U \cup U(J, R, k)$
4  if $\sum_{j \in U} a_j = b$ then $L \equiv I \backslash U$; goto 6
```

```
5 if $\sum_{j\in U} a_j + a_k > b$ then $i \leftarrow k$; $L \equiv I\setminus(U \cup \{i\})$;
                  $\bar{x}_i \leftarrow (b - \sum_{j\in U} a_j)/a_i$;
                  goto 6
          else $U \equiv U \cup \{k\}$
  if $\sum_{j\in U} a_j = b$ then $L \equiv I\setminus U$; goto 6
          else $J \equiv L(J, R, k)$; goto 1
6 $\bar{x}_j \leftarrow 1\ \forall j \in U$; $\bar{x}_j \leftarrow 0\ \forall j \in L$; $v(\bar{B}) \leftarrow c\,\bar{x}$
```

——————————— *End of Phase* 1 ———————————

Is the solution of $(\bar{B})$ *integral?*

```
7 if $\bar{x} \in V$ then $x^* \leftarrow \bar{x}$; $v(B) \leftarrow v(\bar{B})$; stop
```

Find a lower bound on $v(B)$ *by the classical greedy algorithm of* [7]

```
8 Construct $\underline{x} \in V$ such that
    $\underline{x}_j \leftarrow 1\ \forall j \in U$; $\underline{x}_i \leftarrow 0$; by denoting $L = \{l_1, l_2, \ldots, l_{|L|}\}$
    $\underline{x}_{l_j} \leftarrow \begin{cases} 1 \text{ if } x_{l_j} \le b - \sum_{k\in U} a_k - \sum_{k=1}^{j-1} a_{l_k} x_{l_k} \\ 0 \text{ otherwise} \end{cases} \quad j = 1, 2, \ldots, |L|.$
  $\underline{v}(B) \leftarrow c\,\underline{x}$
9 $\bar{v}(B) \leftarrow \lfloor v(\bar{B}) \rfloor$
  if $\underline{v}(B) = \bar{v}(B)$ then $x^* \leftarrow \underline{x}$; $v(B) \leftarrow \underline{v}(B)$; stop
```

Phase 2: *Reduction of the size of* (B):
The upper bounds on the optimal values of the following subproblems are values of the lagrangean relaxation of (B) *associated with* c_i/a_i *(optimal multiplier associated with* $ax \le b$ *for* $(\bar{B})$*)*; i.e. $\forall j \in \{1, \ldots, n\}$ $\forall \varepsilon \in \{0, 1\}$

$$\bar{v}(B \mid x_j = \varepsilon) = \lfloor c_i b/a_i + \max\{(c - (c_i/a_i)a)x \mid x \in V;\ x_j = \varepsilon\} \rfloor:$$

Notations:

$x_j \overset{*}{\leftarrow} \varepsilon : x_j$ *must be fixed at* $\varepsilon \in \{0, 1\}$ *in order to improve the current value of* $\underline{v}(B)$

$$X_0 = \{j \in I \mid x_j \overset{*}{\leftarrow} 0\} \quad X_1 = \{j \in I \mid x_j \overset{*}{\leftarrow} 1\}$$

```
10 $X_0 \equiv X_1 \equiv \emptyset$   $X_2 \equiv \{1, \ldots, n\}$
   $\forall j \in U \cup L$, given $\varepsilon_j = \begin{cases} 0 \text{ if } j \in U \\ 1 \text{ if } j \in L, \end{cases}$
   if $\bar{v}(B \mid x_j = \varepsilon_j) \le \underline{v}(B)$ then $x_j \overset{*}{\leftarrow} 1 - \varepsilon_j$
                          $X_{1-\varepsilon_j} \equiv X_{1-\varepsilon_j} \cup \{j\}$; $X_2 \equiv X_2 \setminus \{j\}$
   $\forall j \in X_2$, if $a_j > b - \sum_{k\in X_1} a_k$ then $x_j \overset{*}{\leftarrow} 0$; $X_0 \equiv X_0 \cup \{j\}$; $X_2 \equiv X_2 \setminus \{j\}$
```

11 **if** $X_2=\emptyset$ **then** $x^*\leftarrow \underline{x}$; $v(B)\leftarrow \underline{v}(B)$; stop

---

12 *Hierarchy of variables: Let the variables of the reduced problem be reindexed in the following manner (see [6]):*

- * *when $n<1000$: use the "Quicksort" method (see [10]) in order to arrange in increasing order the absolute values of the optimal relative costs of $(\bar{B})$, that is*

$$\left|c_j-\frac{c_i}{a_i}a_j\right| \qquad \forall j\in X_2$$

(i always denotes the basic variable index) (see [2]).

- * *when $n\geq 1000$: the arrangement based on the relative costs of $(\bar{B})$ is realized by the Quicksort method with a threshold value – denoted by t – which leads to unsorted files of length t or less.*

Note: The parameter t is an increasing function of the size $m=|X_2|$ of the reduced problem, and could be fitted at the end of phase 2. But for a practical point of view, its values are in fact chosen as a function of the size n of the given problem; for example, for the randomly generated problems of section 5, the different values of pairs (n, t) are:

n	1000	2000	5000	7500
t	8	10	35	60

Phase 3: *Implicit enumeration for the reduced problem (including a reduction scheme)*:

(*i*) *After renumbering the variables from 1 to $m=|X_2|$ (from the smallest $\left|c_j-\frac{c_i}{a_i}a_j\right|$ to the highest ones), explicit enumeration of the set of the unit cube vertices of $\mathbb{R}^m$ is realized*:

– *by applying a lexicographical search for the unit cube of $\mathbb{R}^{m-2}$ (associated with the subset of variables whose indices are in $I=\{3,4,\ldots,m\}$): starting from x_I^1 defined as $x_j^1=\lfloor\bar{x}_j\rfloor$ $\forall j\in I$,*
the 2^{m-2} other unit cube vertices x_I^j $(j=2,\ldots,2^{m-2})$ are searched in the following order:

$$\begin{array}{ll}
x_I^2: & (1-x_3^1,\quad x_4^1,\quad x_5^1,\ldots,\quad x_m^1)\\
x_I^3: & (\quad x_3^1,\quad 1-x_4^1,\quad x_5^1,\ldots,\quad x_m^1)\\
x_I^4: & (1-x_3^1,\quad 1-x_4^1,\quad x_5^1,\ldots,\quad x_m^1)\\
x_I^5: & (\quad x_3^1,\quad x_4^1,\quad 1-x_5^1,\ldots,\quad x_m^1)\\
\vdots & \\
x^{2^{m-2}}: & (1-x_3^1,\quad 1-x_4^1,\quad 1-x_5^1,\ldots,\quad 1-x_m^1)
\end{array}$$

– by solving the following auxiliary problem for each 0-1 vector $x_I^k (k=1,\dots,2^{m-2})$ of $\mathbb{R}^{m-2}$ considered as parameters:

$$(PA)\quad \left[\begin{array}{ll} c_I x_I^k + \max c_1 x_1 + c_2 x_2 & \\ \quad s.t. & a_1 x_1 + a_2 x_2 \le b - a_I x_I^k \\ & x_1, x_2 \in \{0,1\} \end{array}\right.$$

(*ii*) *The arborescence associated with the explicit enumeration of the set of solutions is implicitly searched as follows:*

Given a node of this directed tree, that is a partition $\{S_0, S_1, S_2\}$ of X_2

$$where \quad \left[\begin{array}{l} S_0 = \{j \mid x_j = 0\} \\ S_1 = \{j \mid x_j = 1\} \end{array}\right.$$

the following tests are applied to the subproblem

$$(P) \equiv (B \mid x_j = 0\ \forall j \in X_0 \cup S_0;\ x_j = 1\ \forall j \in X_1 \cup S_1)$$

i.e.

$$\left[\begin{array}{ll} \sum_{j\in X_1 \cup S_1} c_j + \max \sum_{j \in S_2} c_j x_j & \\ \quad s.t. & \sum_{j\in S_2} a_j x_j \le b(P) = b - \sum_{j \in X_1 \cup S_1} a_j \\ & x_j = 0 \text{ or } 1\ \forall j \in S_2 \end{array}\right.$$

(*The upper bound used is the value of the lagrangean relaxation of* (*B*) *associated with c_i/a_i*

$$\text{i.e. } \bar{v}(P) = c_i b/a_i + \max\{(c - (c_i/a_i)a)x \mid x_j = 0\ \forall j \in X_0 \cup S_0;$$
$$x_j = 1\ \forall j \in X_1 \cup S_1;\ x_j = 0 \text{ or } 1\ \forall j \in S_2\}):$$

```
13.0  $z \leftarrow \sum_{j \in X_1 \cup S_1} c_j$

13.1  if $\bar{v}(P) \le \underline{v}(B)$ then $\nexists x \in F(P): cx > \underline{v}(B)$
                                   goto 13.13

13.2  if $b(P) < 0$ then $F(P) \equiv \emptyset$; goto 13.13

13.3  if $b(P) = 0$ then $x_j \leftarrow 0\ \forall j \in S_2$
                         $S_0 \equiv S_0 \cup S_2$; $S_2 \equiv \emptyset$;
                         goto 13.10

13.4  $\forall j \in S_2$ : if $a_j > b(P)$ then $x_j \leftarrow 0$; $S_0 \equiv S_0 \cup \{j\}$
                                       $S_2 \equiv S_2 \backslash \{j\}$

13.5  if $S_2 = \emptyset$ then goto 13.10
```

```
13.6  if $\sum_{j\in S_2} a_j \le b(P)$ then $x_j \leftarrow 1\ \forall j \in S_2$
                             $S_1 \equiv S_1 \cup S_2;\ S_2 \equiv \emptyset$
                             $z \leftarrow v(P) = \sum_{j\in X_1\cup S_1} c_j$
                             goto 13.10

13.7  if $\bar{v}(P) \le \underline{v}(B)$ then $\nexists x \in F(P): cx > \underline{v}(B)$
                             goto 13.13

13.8  Let $x(P)$ be the optimal solution of a relaxation of $(P)$
      if $x(P) \in F(P)$ then $z \leftarrow v(P) = cx(P)$
                             goto 13.10

13.9  Find $\underline{v}(P)$ by any heuristic method; $z \leftarrow \underline{v}(P)$

13.10 if $z > \underline{v}(B)$ then $\underline{v}(B) \leftarrow z$
              if $\underline{v}(B) = \bar{v}(B)$ then $v(B) \leftarrow \underline{v}(B)$; stop.
      reduction of the size of (B):
      $\forall j \in (U\backslash X_1) \cup (L\backslash X_0)$, given $\varepsilon_j = \begin{cases} 0 & \text{if } j \in U\backslash X_1 \\ 1 & \text{if } j \in L\backslash X_0 \end{cases}$
      if $\bar{v}(B \mid x_j = \varepsilon_j) \le \underline{v}(B)$ then $x_j \overset{*}{\leftarrow} 1 - \varepsilon_j$
                             $X_{1-\varepsilon_j} \equiv X_{1-\varepsilon_j} \cup \{j\}$
                             $X_2 \equiv X_2 \backslash \{j\}$
      if $X_2 = \emptyset$ or $X_1 \cap S_0 \ne \emptyset$ or $X_0 \cap S_1 \ne \emptyset$ then $x^* \leftarrow \underline{x}$; $v(B) \leftarrow \underline{v}(B)$; stop
      if $z = v(P)$ then goto 13.13

13.11 reduction of the size of (P)
      $\forall j \in S_2$, given $\varepsilon_j = \begin{cases} 0 & \text{if } j \in U \cap S_2 \\ 1 & \text{if } j \in L \cap S_2 \end{cases}$
      if $\bar{v}(P \mid x_j = \varepsilon_j) \le \underline{v}(B)$ then $x_j \leftarrow 1 - \varepsilon_j$
                             $S_{1-\varepsilon_j} \equiv S_{1-\varepsilon_j} \cup \{j\}$
                             $S_2 \equiv S_2 \backslash \{j\}$

13.12 if $S_2 = \emptyset$ then
          if $x \in F(P)$ and $cx > \underline{v}(B)$ then
              $\underline{v}(B) \leftarrow cx$
              if $\underline{v}(B) = \bar{v}(B)$ then $x^* \leftarrow x$; $v(B) \leftarrow \underline{v}(B)$; stop
              repeat the phase of reduction of (B) (step 13.10)
          goto 13.13
        else Branching step in the lexicographical search framework

13.13 Backtracking step in the lexicographical search framework
```

Note:

(i) Two other algorithms with a linear time complexity for solving the knapsack relaxation are also proposed in [1, 8].

(ii) *Reduction phases*: $x_j \overset{*}{\leftarrow} \varepsilon \in \{0,1\}$ does not imply that $x_j^* = \varepsilon$ (when $v(B) = \underline{v}(B)$, for example). The dominance relations between variables described in [4, 6, 11, 12] are not implemented in this code.

(iii) Steps 13.8 and 13.9 are not performed for problems with randomly generated data.

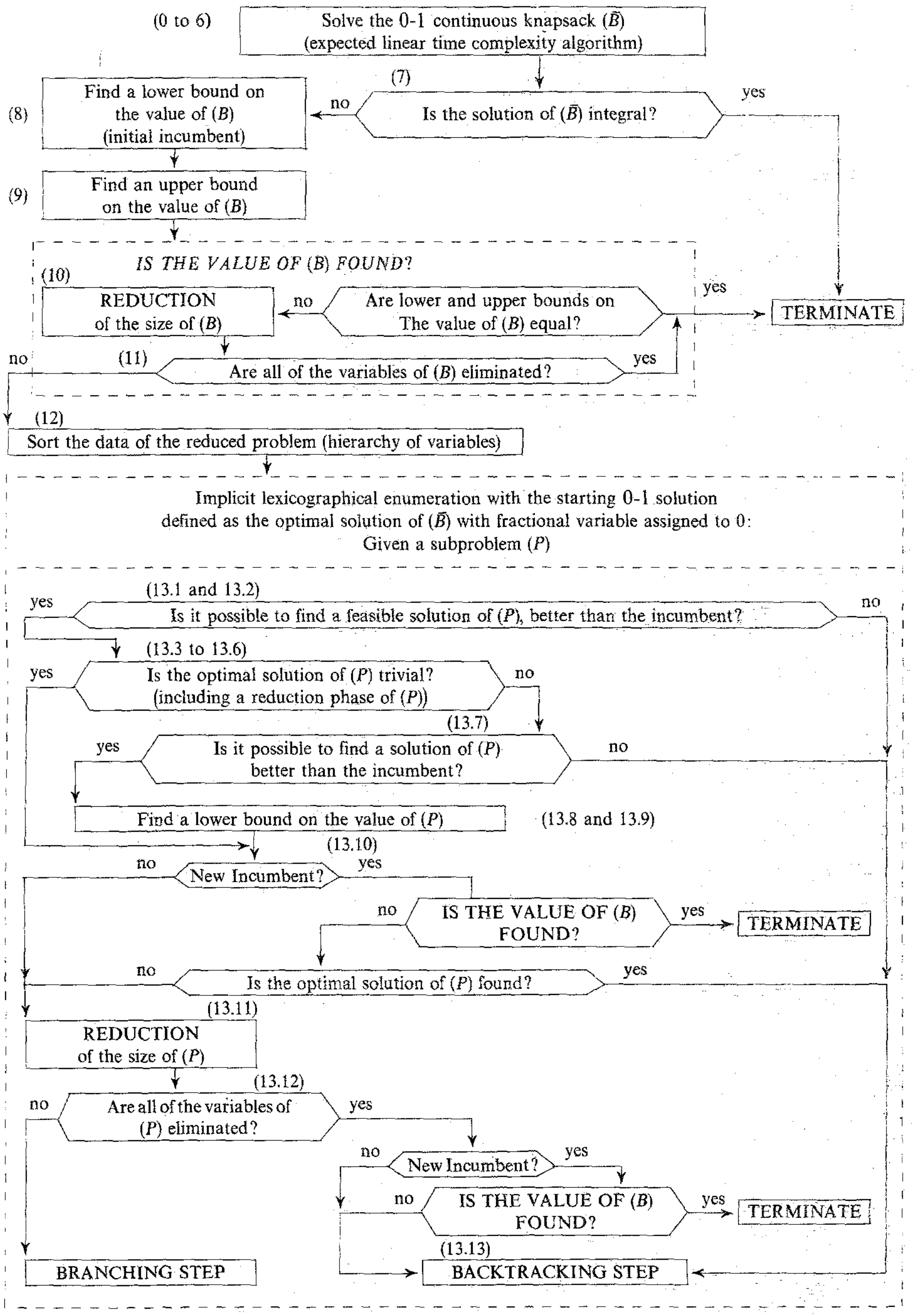
(0 to 6)
Solve the 0-1 continuous knapsack ($\bar{B}$) (expected linear time complexity algorithm)
(7)
Is the solution of ($\bar{B}$) integral?
yes
no
(8)
Find a lower bound on the value of (B) (initial incumbent)
(9)
Find an upper bound on the value of (B)
IS THE VALUE OF (B) FOUND?
(10)
REDUCTION of the size of (B)
no
Are lower and upper bounds on The value of (B) equal?
yes
TERMINATE
no
(11)
Are all of the variables of (B) eliminated?
yes
(12)
Sort the data of the reduced problem (hierarchy of variables)
Implicit lexicographical enumeration with the starting 0-1 solution defined as the optimal solution of ($\bar{B}$) with fractional variable assigned to 0: Given a subproblem (P)
(13.1 and 13.2)
yes
Is it possible to find a feasible solution of (P), better than the incumbent?
no
(13.3 to 13.6)
yes
Is the optimal solution of (P) trivial? (including a reduction phase of (P))
no
(13.7)
yes
Is it possible to find a solution of (P) better than the incumbent?
no
Find a lower bound on the value of (P)
(13.8 and 13.9)
(13.10)
no
New Incumbent?
yes
no
IS THE VALUE OF (B) FOUND?
yes
TERMINATE
no
Is the optimal solution of (P) found?
yes
(13.11)
REDUCTION of the size of (P)
(13.12)
no
Are all of the variables of (P) eliminated?
yes
no
New Incumbent?
yes
no
IS THE VALUE OF (B) FOUND?
yes
TERMINATE
(13.13)
BRANCHING STEP
BACKTRACKING STEP

Fig. 1. A flowchart of the algorithm

4. The FORTRAN Subroutine

4.1 Parameter List

Parameters of the FORTRAN IV subroutine FPK 79 are described at the beginning of the code (see section 6); entrance to the subroutine is achieved by using the statement

CALL FPK 79 (C, A, B, N, XG, $N1$, ISORT).

All the parameters are integer; their meanings and the associated mathematical notation (in brackets) are:

Input parameters

C	objective function (vector)	(c)
A	left hand-side of the constraint (vector)	(a)
B	right-hand side of the constraint (scalar)	(b)
N	problem size (scalar)	(n)
ISORT	scalar associated with the data arrangement for the reduced problem, in connection with the increasing order of the absolute values of the optimal reduced costs of $(\bar{B})$ $=0$ if Quicksort method is performed >0 (see phase 3 (ii)) if Quicksort method with a threshold value is used.	(t)

Output parameters

XG	optimal solution of (B)	(x^*)
$N1$	optimal value of (B)	$(v(B))$

4.2 Main Local Variables

a) n-dimensional vectors:

XE = current solution (i.e. associated with $\underline{v}(B)$) before algorithm stops; optimal solution of (B) at the end of the algorithm (indices are not in the initial order)

XG = solution $\bar{x}$ with $\bar{x}_i = 0$

IP = reference to the initial order of variables and to the heuristic solution:

$$\forall j \in L \quad \underline{x}_j = \begin{cases} 1 & \text{if } IP(j) < -100\,000 \\ 0 & \text{if } -n < IP(j) < 0 \end{cases}$$

X: given a current solution x, allows to compare values of x_j and $\bar{x}_j$:

$$X(j) = \begin{cases} 1 & \text{if } j \in S_0 \cap U \text{ or } j \in S_1 \cap L \\ 0 & \text{otherwise} \end{cases}$$

b) $(n+1)$*-dimensional vector:*

DA = absolute values of the reduced costs arranged in decreasing order *c) scalars*:

M	$=$number of variables (n)
$L1$	$=$right hand-side of the constraint (b)
FVC	$=v(\bar{B})$
SUP	$=\min\{j \mid x_j \text{ not eliminated}\}$
$IZ41$	$=\lfloor v(\bar{B})\rfloor = \bar{v}(B)$
N	$= \sum_{j\in X_1\cup S_1} c_j$
$N1$	$=\underline{v}(B)$

$N3+K2=\bar{v}(P)$ defined in phase 3 (ii)

DIF	$=b-\sum_{j\in X_1\cup S_1} a_j - \sum_{j\in S_2\cap U} a_j$
$K2R$	$=\sum_{j\in S_2} a_j$

4.3 *Code Structure*

		Statements
(i) *Phase 1*:	– Relaxation	12 to154
(ii) *Phase 2*:	– Heuristic method	155 to 176
	– Elimination of variables	178 to 216
(iii) *Phase 3*:	– Hierarchy of variables	217 to 322
	– First current solution	323 to 355
	– Parameters for solving problem (PA)	356 to 380
	– Resolution of problem (PA)	381 to 435
	– New current solution	437 to 470
	– Implicit enumeration	471 to 570
	– Solution in the initial order of indices	571 to 596

5. Computational Results

Subroutine FPK 79 has been tested on a CII HB IRIS 80, on an IBM 370/168 and on a UNIVAC 1110 with a lot of problems with randomly generated – up to 7500 variables – and concrete – up to 200 variables – data; no breakdown occured.

All the times are in milliseconds on a UNIVAC 1110; n denotes the size of problems.

Tables 1, 2, 3 and 5 concern $n=50$-, 100-, 500-, 1000-, 2000-, 5000-, 7500-variable problems with data generated from a uniform distribution:

$$\forall j\in\{1,\dots,n\}\ c_j\in[0,100[\ a_j\in]0,100[$$

and

(i) $b\in\left[\max_{1\le j\le n} a_j, \sum_{j=1}^{n} a_j\right[$ for tables 1, 2 and 5

(ii) $b=\alpha \sum_{j=1}^{n} a_j$ with $\alpha=.2,.5,.8$ for table 3 (only for $n=1000$); a set of 50 problems is considered for each size.

Tables 4 and 6 consider more realistic cases: $n=75$-, 100-, 150-, 200-variable problems with a concrete origin, whose features are summarized in Table 7 (see [3, 11] for more details).

Computational times of Tables 1, 2, 3 and 4 include times for the data arrangement which leads to the solving of the relaxed problem. Tables 5 and 6 are relative to problems whose data are supposed to be arranged so that

$$c_j/a_j \geq c_{j+1}/a_{j+1} \; j=1,2,\ldots,n-1.$$

Table 1 summarizes times phase by phase for algorithm FPK 79 whose total times are compared to those of algorithm MSB 71 (with the use of Quicksort method [10] for data sorting).

Tables 2 to 6 contain comparisons between algorithms FPK 79, MSB 71 and the algorithm 37 of Martello and Toth (the authors' code published in [9] and Quicksort method for data sorting were used). Algorithm FPK 79 is shown to be the fastest method in all these cases.

More extensive computational results can be found in [6].

Table 1. *Average times in milliseconds for randomly generated problems (when data have to be sorted for solving the relaxed problem)*

n	Relaxation	Reduction	Implicit Enumeration	Total Time	gain % MSB 71
50	3.0	1.2	2.4	6.6	31.3
100	5.6	2.4	5.2	13.2	36.8
500	27.9	10.6	11.5	50.0	51.2
1000	50.7	23.6	9.6	83.9	59.5
2000	112.2	31.8	21.2	165.2	62.0
5000	281.0	81.1	28.8	390.9	–
7500	394.3	105.5	33.8	533.6	–

Table 2. *Average times (Maximum times) in milliseconds for randomly generated problems (when data have to be sorted for solving the relaxed problem)*

n	MSB 71	FPK 79	MT 78
50	9.6 (17.0)	6.6 (13.2)	10.3 (20.2)
100	20.9 (31.7)	13.2 (31.3)	22.0 (37.2)
500	102.4 (213.0)	50.0 (105.6)	110.7 (151.2)
1000	207.4 (309.3)	83.9 (142.8)	203.8 (268.2)
2000	435.3 (720.6)	165.2 (266.0)	416.1 (469.9)
5000	–	390.9 (532.0)	1109.0 (1370.0)
7500	–	533.6 (1042.0)	–

Table 3. *Average times in milliseconds for randomly generated problems in* 1000 *variables (when data have to be sorted for solving the relaxed problem)*

		FPK 79				MT 78
		Relaxation	Reduction	Implicit Enumeration	Total Time	
$b=\alpha\sum a_j$	$\alpha=0.2$	56.2	25.9	13.4	95.5	205.7
	$\alpha=0.5$	55.3	25.0	14.2	94.5	209.2
	$\alpha=0.8$	50.1	23.6	8.4	82.1	219.9
$\max a_j \le b < \sum a_j$		50.7	23.6	9.6	83.9	203.8

Table 4. *Average times (Maximum times) in milliseconds for concrete problems* ([3, 11], *Table* 7) *(when data have to be sorted for solving the relaxed problem)*

Series	n	MSB 71	FPK 79	MT 78
1	200	42.1 (78.8)	25.8 (60.0)	42.1 (56.5)
2	100	19.2 (25.8)	12.6 (19.2)	19.0 (23.2)
3	100	28.3 (37.9)	16.1 (26.2)	22.2 (34.8)
4	100	26.2 (37.9)	16.1 (22.0)	22.5 (35.2)
5	100	19.8 (25.4)	12.3 (16.2)	18.5 (22.2)
6	150	259.1 (2581.7)	239.0 (2540.6)	673.2 (7570.8)
7	75	108.8 (1315.7)	52.6 (259.4)	174.2 (1969.2)

Table 5. *Average times (Maximum times) in milliseconds for randomly generated problems (when data are supposed to be arranged so that $c_j/a_j \geq c_{j+1}/a_{j+1}$ $j=1, \ldots, n-1$)*

n	MSB 71	FPK 79	MT 78
50	4.9 (11.5)	4.1 (9.6)	5.6 (15.6)
100	10.2 (22.4)	8.4 (26.1)	11.3 (26.6)
500	32.7(111.3)	25.5 (85.3)	36.8 (77.4)
1000	55.1 (156.9)	40.3 (101.5)	51.5 (116.0)
2000	104.0 (390.5)	67.5 (134.0)	84.8 (138.6)
5000	–	143.4 (243.4)	183.1 (444.2)

Table 6. *Average times (Maximum times) in milliseconds for concrete problems* [3, 11] *(when data are supposed to be arranged so that $c_j/a_j \geq c_{j+1}/a_{j+1}$ $j=1, \ldots, n-1$)*

Series	n	MSB 71	FPK 79	MT 78
1	200	17.0 (53.7)	16.9 (48.4)	17.0 (31.4)
2	100	8.6 (15.1)	7.7 (11.2)	8.3 (12.6)
3	100	12.6 (27.2)	10.9 (20.8)	11.5 (24.2)
4	100	15.6 (27.2)	10.6 (17.0)	11.8 (24.6)
5	100	9.1 (14.7)	7.6. (11.8)	7.9 (11.6)
6	150	238.9 (2561.5)	232.3 (2535.2)	653.0 (7550.6)
7	75	99.9 (1306.7)	48.2 (255.2)	165.2 (1960.4)

Table 7. *Data features of the concrete problems described in* [3, 11]

Series	n	number of problems	set of			mean values	
			c_j	a_j	b	c_j	a_j
1	200	10	[1, 222]	[1, 283]	[500, 12000]	56.1	67.4
2	100	10	[1, 99]	[1, 99]	[200, 4500]	48.4	49.4
3	100	10	[4, 222]	[2, 283]	[500, 6000]	63.7	61.2
4	100	10	[13, 174]	[15, 254]	[500, 6000]	68.9	78.8
5	100	10	[1, 98]	[30, 85]	[200, 4500]	49.5	57.4
6	150	12	[1, 1200]	[1, 3541]	[500, 8000]	91.4	149.1
7	75	22	[1, 1000]	[1, 3541]	[1500, 14000]	160.7	273.9

6. FORTRAN Subroutine FPK 79

```
      SUBROUTINE FPK79  (C,A,B ,N,XG,NI,ISORT)
C
C        THIS SUBROUTINE SOLVES THE 0-1 KNAPSACK PROBLEM
C           ...   NI=C*XG=MAXIMIZE     C(1)*X(1)+...+C(N)*X(N)
C           .                    SUBJECT TO
C        (B).
C           .                          A(1)*X(1)+...+A(N)*X(N)<=B
C           ...                        X(J)=0 OR 1       J=1,...,N     WITH   N>2
C        ALL PARAMETERS ARE INTEGER
C
C        INPUT PARAMETERS :
C          - VECTORS C AND A      ....
C                                     . INCLUDED IN (B)
C          - SCALARS B  AND N     ....
C          - SCALAR ISORT DEALS WITH  THE SORTING OF THE DATA
C                         OF THE REDUCED PROBLEM BY DECREASING
C                         ORDER OF THE REDUCED COSTS IN ABSOLUTE VALUE
C                          .... 0 IF QUICKSORT ALGORITHM  IS USED
C                    ISORT =
C                          ....  THRESHOLD VALUE UNDER WHICH THE
C                                SUBFILES PRODUCED DURING SORTING ARE IGNORED
C
C        OUTPUT PARAMETERS :
C          - SCALAR NI = OPTIMAL VALUE      ....
C                                               . OF (B)
C          - VECTOR XG = OPTIMAL SOLUTION ....
C
C
      INTEGER A(N),C(N),XG(N)
      DIMENSION IP(7500),DA(7501),IAUX1(100),IAUX2(100)
      INTEGER XE(7500),X(7500)
      INTEGER SUP,DIF,CM,CM2,SUP2,VE,EV,XM,XM1,XM2,PLUS
      INTEGER EX,XGM,S,B
      REAL K2,K2R,IVAL
      DO 2 I=1,N
    2 IP(I)=I
      L1=B
C
C        CALL  THE  TIMING  SUBROUTINE
C
C
C        SOLVING THE RELAXED PROBLEM
C
      S=0
      ISEUIL=10
      K=0
      DO 10 I=1,N
      R=A(I)
      X(I)=C(I)
   10 DA(I)=C(I)/R
      IB1=1
      IC1=N
   12 IB=IB1
      IH=IC1
      IF(IH-IB.GE.ISEUIL) GOTO 5
      J1=IB+1
   23 IF(J1.GT.IH) GOTO 301
      J2=J1-1
   91 IF(J2.LT.IB) GOTO 92
      IF(DA(J2).GE.DA(J2+1)) GOTO 92
      J3=J2+1
      IPV=IP(J2)
      VAL=DA(J2)
      IVAL=A(J2)
      IP(J2)=IP(J3)
      A(J2)=A(J3)
      DA(J2)=DA(J3)
      A(J3)=IVAL
      IP(J3)=IPV
      DA(J3)=VAL
      J2=J2-1
      GOTO 91
   92 J1=J1+1
      GOTO 23
    5 J1=IB
      J4=(IH+IB)/2
      J2=J4
      IF(DA(J1).LE.DA(J2)) GOTO 95
      J1=J4
      J2=IB
   95 J3=IH
      IF(DA(J3).GE.DA(J2)) GOTO 96
      J2=IH
      IF(DA(J1).GE.DA(J2)) J2=J1
   96 VAL=DA(J2)
      IVAL=A(J2)
      IPV=IP(J2)
      DA(J2)=DA(IH)
      A(J2)=A(IH)
      IP(J2)=IP(IH)
      IF(IB.GE.IH) GOTO 8
    6 IF (DA(IB).LT. VAL) GOTO 7
      K=K+A(IB)
      IB=IB+1
      IF(IB.GE.IH) GOTO 8
      GOTO 6
    7 DA(IH)=DA(IB)
      A(IH)=A(IB)
      IP(IH)=IP(IB)
    4 IH=IH-1
      IF(IB.GE.IH) GOTO 8
      IF(DA(IH).LE.VAL) GOTO 4
      DA(IB)=DA(IH)
      A(IB)=A(IH)
      IP(IB)=IP(IH)
      K=K+A(IH)
      IB=IB+1
      IF(IB.LT.IH) GOTO 6
    8 DA(IB)=VAL
      A(IB)=IVAL
      K=K+IVAL
      IP(IB)=IPV
      IF(K.LT.L1) GOTO 11
      K=K-A(IB)
      IF (K.LE.L1) GOTO 200
      K=S
      IC1=IB-1
      GOTO 12
   11 IB1=IB+1
      S=K
      GOTO 12
  200 IC=0
      DO 152 I=1,N
      K13=IP(I)
  152 C(I)=X(K13)
      IC1=IB-1
      IF(IC1.EQ.0) GOTO 98
      DO 99 I=1,IC1
   99 IC=IC+C(I)
   98 VE=L1-K
      IF(VE.EQ.0) GOTO 307
      IF(VE.NE.A(IB)) GOTO 199
      VE=0
      IC1=IB
      IC=IC+C(IB)
      GOTO 307
  199 R=A(IB)
      R=IC+VE*C(IB)/R
      GOTO 307
  301 IC=0
      S=IB-1
      DO 151 I=1,N
      K13=IP(I)
  151 C(I)=X(K13)
      IF(IB.EQ.1) GOTO 310
      DO 303 I=1,S
  303 IC=IC+C(I)
  310 CONTINUE
      DO 304 I=IB,IH
      J=I
      K=K+A(I)
      IC=IC+C(I)
      IF(K.GE.L1) GOTO 305
  304 CONTINUE
  305 VE=L1-K
      IC1=J
      IF(VE.EQ.0) GOTO 307
      IB=J
      R=A(IB)
      VE=VE+A(IB)
      IC=IC-C(IB)
      R=IC+VE*C(IB)/R
  307 IV2=VE
      N1=IC
      M=N
C
C        CALL  THE  TIMING  SUBROUTINE
C
C
C        GREEDY ALGORITHM  FOR A LOWER BOUND ON THE VALUE OF (B)
C
      IF(VE.EQ.0) GOTO 70
      EV=VE
      IB1=IB+1
      DO 13 I=IB1,N
      IF (VE.LT.A(I)) GOTO 13
      N1=N1+C(I)
      VE=VE-A(I)
      IP(I)=IP(I)+100000
   13 IP(I)=-IP(I)
      IP(IB)=-IP(IB)
      IZ41=R
      FVC=R
      R=A(IB)
      R=C(IB)/R
      M1=N+1
      M2=N-1
      M3=N-2
      DA(M1)=R
      N1E=N1
      IF(IZ41.EQ.N1) GOTO 188
C
C        REDUCTION ALGORITHM
C
      I=1
      SUP=1
   16 DA(I)=ABS(C(I)-R*A(I))
      NAV=FVC-DA(I)
      IF(NAV.LE.N1) GOTO 15
   17 DA(M)=ABS(C(M)-R*A(M))
      NAV=FVC-DA(M)
      IF(NAV.LE.N1) GOTO 14
      M=M-1
      IF(M.NE.I) GOTO 17
      SUP=I
      GOTO 18
   14 R1=DA(M)
      DA(M)=DA(I)
      DA(I)=R1
      H=IP(M)
      IP(M)=IP(I)
      IP(I)=H
      H=A(M)
      A(M)=A(I)
      A(I)=H
      H=C(M)
      C(M)=C(I)
      C(I)=H
      N=M-1
   15 IF(I.GE.M)GOTO 19
      I=I+1
      GOTO 16
   19 SUP=I+1
   18 IF (SUP.LT.N) GOTO 88
  188 IV2=-1
      SUP=M1
      M=N
      GOTO 93
   88 ISEUIL=8
      IF(ISORT.NE.0) ISEUIL=ISORT
C
C        SORTING OF THE DATA OF THE REDUCED PROBLEM
C
      J=1
      IBB=SUP
      IBH=N
  202 IB=IBB
      IH=IBH
      IF(IH-IB.GE.ISEUIL) GOTO 203
      IF(ISORT.NE.0) GOTO 207
      J1=IB+1
   20 IF(J1.GT.IH) GOTO 207
      J2=J1-1
   21 IF(J2.LT.IB) GOTO 22
      IF(DA(J2).GE.DA(J2+1)) GOTO 22
      J3=J2+1
      IPC=C(J2)
      IPB=A(J2)
      IPV=IP(J2)
      IVAL=DA(J2)
      IP(J2)=IP(J3)
      DA(J2)=DA(J3)
      A(J2)=A(J3)
      C(J2)=C(J3)
      A(J3)=IPB
      C(J3)=IPC
      DA(J3)=IVAL
      IP(J3)=IPV
      J2=J2-1
      GOTO 21
   22 J1=J1+1
      GOTO 20
  203 J1=IB
      J4=(IH+IB)/2
      J2=J4
      IF(DA(J1).LE.DA(J2)) GOTO 35
      J1=J4
      J2=IB
   35 J3=IH
      IF(DA(J3).GE.DA(J2)) GOTO 36
      J2=IH
      IF(DA(J1).GE.DA(J2)) J2=J1
   36 IVAL=DA(J2)
      IPV=IP(J2)
      IPB=A(J2)
      IPC=C(J2)
      DA(J2)=DA(IBB)
      IP(J2)=IP(IBB)
      A(J2)=A(IBB)
      C(J2)=C(IBB)
      IF(IB.GE.IH) GOTO 210
  206 IF(DA(IH).GT.IVAL) GOTO 205
      IH=IH-1
      IF(IB.GE.IH) GOTO 210
      GOTO 206
  205 DA(IB)=DA(IH)
      IP(IB)=IP(IH)
      A(IB)=A(IH)
      C(IB)=C(IH)
  204 IB=IB+1
      IF(IB.GE.IH) GOTO 210
      IF(DA(IB).GE.IVAL) GOTO 204
      DA(IH)=DA(IB)
      IP(IH)=IP(IB)
      C(IH)=C(IB)
      A(IH)=A(IB)
      IH=IH-1
      IF(IB.LT.IH) GOTO 206
  210 DA(IB)=IVAL
      IP(IB)=IPV
      A(IB)=IPB
      C(IB)=IPC
      I=IB
      IF (J.GT.100) GOTO 201
      IAUX1(J)=I
      IAUX2(J)=IBH
      J=J+1
      IF(I.LE.IBB+1) GOTO 207
      IBH=I-1
      GOTO 202
  207 J=J-1
      IF(J.EQ.0) GOTO 208
      I=IAUX1(J)+1
      IBH=IAUX2(J)
      IF(IBH.LE.I) GOTO 207
      IBB=I
      GOTO 202
  208 CONTINUE
      IF(SUP.EQ.1) GOTO 73
   93 ISU2=SUP-1
      DO 74 I=1,ISU2
      XE(I)=1
      XG(I)=1
      IF(IP(I).GT.0) GOTO 74
      XE(I)=0
      XG(I)=0
      IP(I)=-IP(I)
      IF(IP(I).LE.100000) GOTO 74
      XE(I)=1
      IP(I)=IP(I)-100000
   74 CONTINUE
      IF(IV2.LT.0) GOTO 70
   73 DO 75 I=SUP,N
      X(I)=0
      IF(IP(I).GT.0) GOTO 76
      XE(I)=0
      XG(I)=0
      IP(I)=-IP(I)
      IF(IP(I).LE.100000) GOTO 75
      IP(I)=IP(I)-100000
      XE(I)=1
      GOTO 75
   76 XE(I)=1
      XG(I)=1
   75 CONTINUE
C
C        PARAMETERS FOR THE RESOLUTION OF
C        THE AUXILIARY PROBLEM (DENOTED BY (P.A.) )
C
      LM=A(N)
      LM2=A(M2)
      CM=C(N)
      CM2=C(M2)
      EX=XG(M2)
      XGM=XG(N)
      IF(LM-LM2.LE.0) GOTO 81
      SUP2=LM
      INF=LM2
      J=N
      GOTO 82
   81 SUP2=LM2
      INF=LM
      J=N-1
   82 IH=0
      IF(C(J).LT.C(2*N-1-J)) IH=1
      LS=LM+LM2
      DIF=EV+LM2*EX+LM*XGM
      M=N
      N3=IC
      N=N3-CM2*EX-CM*XGM
      S=SUP-1
      K=SUP
C
C        IMPLICIT ENUMERATION ALGORITHM
C        NEW CURRENT SOLUTION BY SOLVING (P.A.)
C
C        CALL  THE  TIMING  SUBROUTINE
C
C
      GOTO 86
   56 X(K)=1
      TP2=N
      DIF1=DIF
      IF(XG(K).NE.1)GOTO 27
      N=N-C(K)
      DIF=DIF+A(K)
      GOTO 28
   27 N=N+C(K)
      DIF=DIF-A(K)
   28 N=N-PLUS
      IF(DIF.GE.0) GOTO 86
      IF(K.NE.M3)GOTO 58
      X(K)=0
      IF(XG(K).EQ.0)K2R=K2R+A(K)
      DIF=DIF1
      N=TP2
      GOTO 48
   86 IF(DIF.GE.LS) GOTO 66
      IF(DIF.LT.INF) GOTO 65
      IF(IH.EQ.1) GOTO 61
      IF(DIF.GE.SUP2) GOTO 64
   61 IF(INF.EQ.LM) GOTO 63
   62 XM2=1
      XM=0
      XM1=DIF-LM2
      GOTO 67
   63 XM2=0
      XM=1
      XM1=DIF-LM
      GOTO 67
   64 IF(INF.EQ.LM) GOTO 62
      GOTO 63
   65 XM2=0
      XM=0
      XM1=DIF
      GOTO 67
   66 XM2=1
      XM=1
      XM1=DIF-LS
   67 CONTINUE
      PLUS=CM2*XM2+CM*XM
      N=N+PLUS
      N3=N
      IF(N.LE.N1)GOTO 47
C
C        NEW CURRENT OPTIMAL SOLUTION
C
      N1=N
      IF(SUP.GT.M3)GOTO 43
      DO 42 I=SUP,M3
      XE(I)=XG(I)
      IF(X(I).EQ.1) XE(I)=1-XG(I)
   42 CONTINUE
   43 XE(M2)=XM2
      XE(M)=XM
      VE=XM1
      IF(IZ41.EQ.N1) GOTO 70
```

```
          IF(SUP .GT.M1)GOTO 45
          ISU2=SUP
          DO 44 I=ISU2,M1
          NAV=FVC-DA(I)
          IF(NAV.GT.N1) GOTO 45
          SUP=SUP+1
          IF(X(I).EQ.1)GOTO 70
       44 CONTINUE
       45 IF(SUP .GE.M)GOTO 70
          IF(ISCRT.NE.0) GOTO 47
       46 IF(X(SUP).NE.1) GOTO 47
          SUP=SUP+1
          IF(SUP.GE.M)GOTO 70
          GOTO 46
C
C              IMPLICIT LEXICOGRAPHICAL SEARCH
C
       47 K=M2
          Z=DA(M1)*XM1
          K2R=DIF
          IF(XM2.NE.EX)Z=Z+DA(M2)
          IF(XM.NE.XGM) Z=Z+DA(M)
       48 IF(K.EQ.SUP)GOTO70
          K=K-1
          IF(X(K).NE.1)GOTO 50
          X(K)=0
          IF(XG(K).NE.1)GOTO 49
          DIF=DIF-A(K)
          N=N+C(K)
          GOTO 51
       49 DIF=DIF+A(K)
          N=N-C(K)
          K2R=K2R+A(K)
       51 Z=Z+DA(K)
          GOTO 48
C              STEP 6
       50 K2=-DA(K)+Z
          IF(N3+K2.LE.N1)GOTO 57
          IF(XG(K).NE.1)GOTO 52
          K2R=K2R+A(K)
          GOTO 53
       52 K2R=K2R-A(K)
C              STEP 1 AND STEP 2
       53 CONTINUE
          IF(K2R.GE.0)GOTO 54
          IF(XG(K).EQ.0)  K2R=K2R+A(K)
          GOTO 48
C              STEP 3,5 AND 6
       54 EV=K2R
          IS2=K+1
          IF(IS2.GT.M) GOTO 77
          DO 55 I=IS2,M
          IF(A(I).LE.K2R)EV=EV-A(I)
       55 CONTINUE
       77 IF(EV.LT.0)GOTO 56
          IF(N3+K2-DA(M1)*EV.GT.N1)GOTO 83
          IF(XG(K).EQ.0)K2R=K2R+A(K)
          GOTO 48
       57 IF(XG(K).EQ.1) K2R=K2R+A(K)
          GOTO 48
       83 IF(IS2.GT.M3) GOTO 56
          DO 84 I=IS2,M3
          IF(A(I).GT.K2R) GOTO 87
          IF(XG(I).EQ.1) GOTO 84
          X(I)=1
          DIF=DIF-A(I)
          N=N+C(I)
          GOTO 84
       87 IF(XG(I).EQ.0) GOTO 84
          X(I)=1
          DIF=DIF+A(I)
          N=N-C(I)
       84 CONTINUE
          GOTO 56
C
C              EXPLICIT LEXICOGRAPHICAL SEARCH
C
       58 H=2
       59 J=M-H
          IF(X(J).EQ.1) GOTO 60
          IF(XG(J).EQ.0) GOTO 69
          X(J)=1
          DIF=DIF+A(J)
          N=N-C(J)
          IF(DIF.GE.0) GOTO 86
          GOTO 58
       60 X(J)=0
          N=N+C(J)
          DIF=DIF-A(J)
       69 H=H+1
          IF(J.GT.K+1)GOTO 59
          IF(K.EQ.SUP)GOTO 70
          X(K)=0
          IF(XG(K).EQ.0)K2R=K2R+A(K)
          N=TP2
          DIF=DIF1
          GOTO 48
       70 CONTINUE
C
C              CALL  THE  TIMING  SUBROUTINE
C
C
C              ALL OF THE RESULTS ARE GIVEN IN
C              THE INITIAL DATA ORDERING
C
          IF(IV2.NE.0) GOTO 89
          DO 173 I=1,IC1
      173 XE(I)=1
          IC1=IC1+1
          DO 175 I=IC1,M
```

```
175 XE(I)=0
    GOTO 312
 89 IF(N1E.EQ.N1) GOTO 312
    IF(S.EQ.0) GOTO 312
    DO 311 I=1,S
311 XE(I)=XG(I)
312 N=M
    DO 71 I=1.M
    K=IP(I)
    X(K)=C(I)
 71 XG(K)=XE(I)
    DO 104 I=1.M
    K=IP(I)
104 XE(K)=A(I)
    DO 103 I=1,M
    C(I)=X(I)
103 A(I)=XE(I)
    RETURN
201 WRITE(6,102)
102 FORMAT('  ARRAY OUT OF BOUNDS IN QUICKSORT SUBROUTINE ')
    RETURN
    END
```

References

[1] Balas, E., Zemel, E.: An algorithm for large zero-one knapsack problems. Operations Research *28*, 1130—1154 (1980).

[2] Fayard, D., Plateau, G.: Resolution of the 0-1 knapsack problem: comparison of methods. Mathematical Programming *8*, 272—307 (1975).

[3] Fayard, D., Plateau, G.: Techniques de résolution du problème du knapsack en variables bivalentes: partie III. Publication 91, Laboratoire de Calcul, Université des Sciences et Techniques de Lille I, France (1977).

[4] Fayard, D., Plateau, G.: Reduction algorithms for single and multiple constraints 0-1 linear programming problems. Proceedings of the Congress Methods of Mathematical Programming, Zakopane, Poland, 1977.

[5] Fayard, D., Plateau, G.: On "an efficient algorithm for the 0-1 knapsack problem, by Robert M. Nauss". Management Science *24*, 918—919 (1978).

[6] Fayard, D., Plateau, G.: Contribution à la résolution des programmes mathématiques en nombres entiers, Thèse d'Etat, Université des Sciences et Techniques de Lille I, France, 1979.

[7] Greenberg, H., Hegerich, R. L.: A branch search algorithm for the knapsack problem. Management Science *16*, 327—332 (1970).

[8] Lawler, E. L.: Fast approximation algorithms for knapsack problems. Mathematics of Operations Research *4*, 339—356 (1979).

[9] Martello, S., Toth, P.: Algorithm for the solution of the 0-1 single knapsack. Computing *21*, 81—86 (1978).

[10] Sedgewick, R.: Implementing Quicksort programs. Communications of ACM *21*, 847—857 (1978).

[11] Walukiewicz, S.: The size reduction of a binary knapsack problem. Bulletin of the Polish Academy of Sciences, Series Sciences and Technics *23* (1975).

[12] Zoltners, A. A.: A direct descent binary knapsack problem. Journal of ACM *25*, 304—311 (1978).

D. Fayard
I.U.T. Orsay
Plateau du Moulon
B.P. 23
91406 Orsay Cedex
France

G. Plateau
U.S.T. Lille I
IEEA Informatique
Bât. M 3
59655 Villeneuve D'Ascq Cedex
France

Verleger: Springer-Verlag, Mölkerbastei 5, A-1010 Wien. — Herausgeber: Prof. Dr. Hans J. Stetter, Institut für Numerische Mathematik der Technischen Universität Wien, Gußhausstraße 27—29, A-1040 Wien. — Redaktion: Gußhausstraße 27—29, A-1040 Wien. — Hersteller: Satz Austro-Filmsatz Richard Gerin, Fischergasse 1, A-1020 Wien, Druck Paul Gerin, Zirkusgasse 13, A-1021 Wien. — Verlagsort: Wien. — Herstellungsort: Wien. — Printed in Austria.