Control
and Cybernetics

Vol. 19 (1990) No. 1-2

# Testable heuristics for the 0-1 knapsack problem

by

BRUNO APOLLONI
Dipartimento di Scienze dell'Informazione
Universita di Milano, Italy

Two kind of heuristics, fixed time and cut time, are proposed in order to use the running time available in solving 0-1 Knapsack problems efficiently. The basic idea is to build approximate algorithms capable of generating, within fixed time limits, significantly different feasible solutions by explicitly taking into account those indeterminate aspects of the problem which the usual heuristics tend to avoid by using suboptimal rules for loading the knapsack. In the case of those algorithms it is possible to prove that they are efficient with respect to some parameters which are independent of the pattern of the exact solution, when this efficiency can be tested, via distribution-free methods, by small sized samples of the knapsack problem coming from the operational situations one is dealing with.

## 1. Introduction

We refer to the knapsack 0-1 problem with unitary specific profit coefficients:

$$\max \sum_{i=1}^{n} x_i w_i$$

$$\sum_{i=1}^{n} x_i w_i \leq C$$

$$x_i = 0 \text{ or } 1 \ (i = 1, \ldots, n)$$

$C$ and $w_i$ positive, $w_i \leq C$ for each $i$, $\sum_{i=1}^{n} w_i > C$

where $C$ is the capacity of the knapsack, $w_i$ is the weight of, and at the same time the profit connected with, the $i$-th object among the $n$, and $x_i$ equals 1 if the object is loaded into the knapsack, 0 otherwise.
The problem falls into the class of the NP-hard problems, therefore at the moment each exact solving algorithm requires an extremely long running time for a large amount of inputs.

In order to focus better the computational problem, let us consider, as an archetype of such a class, the following exact algorithm, EEK, which solves our problem. The algorithm is made up of the following three nested segments.

**Generation of all the subsets of the input objects**
This corresponds to generating all the binary strings $s$, of length $n$, in any order.

**Cutting out of the unfeasible solutions**
This step allows the cutting out of the unfeasible solutions. Let $s_j$ be the $j$-th element of $s$, the rule is the following:

As soon as in a string $s \sum_{j=1}^{k} w_j s_j$ exceeds $C$ for some $k$, go to the next string.

**Selection of the exact solution**
Every string $s$ which is generated by the first step and not eliminated in the second represents a feasible solution. The one which results in the highest value of $\sum_{j=1}^{n} w_j s_j$ at the completion of segment 1 is the exact solution.

Together with this let us consider the usual HGK greedy heuristic algorithm.

```
BEGIN
  SUM: = 0; I: = 0; X(I): = 0 for each I;
  WHILE SUM ≤ CAPACITY DO
  BEGIN
    I: = I + 1;
    SUM: = SUM + W(I);
    IF SUM ≤ CAPACITY
    THEN X (I): = 1;
  END;
END.
```

Many papers have been written on the epistemological aspects of the computational complexity. The argument of this paper stems from the idea which attributes the high computational complexity of exact algorithms like the previous one, to a choice operator which, in the absence of any theoretical support, essentially shifts between the two edges of a fork of a binary tree [Apolloni and Di Gregorio, 1984]. The formal definition and properties of this operator can be found in [Apolloni and Di Gregorio 1983]; here we trivially focus on the following remarks:

a) in the algorithms for our knapsack problem, and more generally for NP problems when the running time is more than linear in the length of the input,

there are some steps after which the computation may continue by executing more than one set of instructions (computational paths). As there is no optimality theorem which prefers one path to another, the next step is selected by means of the above choice operator. This is why, for example, in branch and bound tecniques only practical rules, and not theorems, are available to select some next branch. And for the same motive there is no reason to prefer the direct or opposite lexicographical order, or indeed any other order, in the EEK algorithm, if we disregard worst-case or probabilistical suggestions, which we will deal with later in the paper.
From another point of view, such arbitrarities correspond to the order in which the input data are processed, and they are due to the fact that there are many permutations of the input data, which do not change the problem, but give rise to the same solution.
In reality, given the lexicographical ordering of the enumeration of the strings $s$ in EEK, the order in which the feasible solutions are listed depends upon the particular permutation according to which the items appear in the input. Moreover the list themselves of the feasible solutions can be obtained by applying the algorithm HGK successively to the various permutations of the input.
Thus the points of choice can be moved from the instruction sequences to the permutations of the input which these sequences have to process.

b) the running time needed to reach the optimal solution is generally strongly dependent on the particular sequencing of the computational paths. In fact, each step which involves a choice is associated with various computational paths, all of which must be carried out should no one of them lead to the exact solution. Consequently, if we put all the compositions of computational paths arising from the cartesian product of the sets of choices into one stack, the execution of the algorithm can be seen as a progressive screening of the elements of the stack which continues until the solution is recognized. Indeed the running time is short or long according to the depth of the path which stop the algorithm in the stack, but generally we cannot know a priori how to place the compositions in order to render the depth a slight one.
Actually EEK is broadly independent on this sequencing, but as soon as a less trivial algorithm is sought, such dependency does appaear.
It is clear that the running time is a non-decreasing function of the number of the above points of choice. The ability of heuristic algorithms [Muller-Merbach 1981] to shorten it, lies in the elimination of most of these points by using some arbitrary criteria which are the bases of univocal selections between the paths. So in our problem special sets of paths have been suggested in the literature [Gens and Levner, 1980; Ibarra and Kim, 1975; Lawler, 1979; Martello and Toth, 1984]. Similarly, if we view EEK as a series of applications of HGK, instead of screening all the possible permutations of the objects, some heuristics suggest considering only one, which is derived from a special ordering of the objects.
Actually, once again the statement of the above criteria may be a matter of

choice, when no useful tool is available to appraise the efficiency of the heuristic a priori. Indeed, deterministic parameters, such as the worst case difference between exact and heuristic solutions, are generally meaningless because of the large range of these differences. For instance the worst-case performance ratio $r$ of Martello and Toth is $> (s + 3)/(s + 4)$ when computational time $= 0(n^s)$ is spent to reach the heuristic solution. As $n$ increases one must reasonably set $s$ at a low value, with the result that $r$ is relatively large and the value of the waste could also be very large.

Although in the past years the probabilistic aproach, both in the direction of [Rabin, 1974] and in the direction of [Karp, 1977] gave many illuminating insights on the complexity of algorithms, this approach may appear to be of little use from a strictly statistical and operational point of view, when no a-priori knowledge is avaliable on the input distribution. Broadly speaking one can state that given an algorithm of time complexity $g(n)$, where $n$ is the length of the input, each non trivial probabilistic sentence about its results is based on statistical hypotheses regarding the relevant parameters of the problem, the testing of which needs a computational time at least equal to $0(g(n))$ [Apolloni and Pittelli,1984.]. For example, with reference to the probabilistic performance of the GOLOSO algorithm [D'Atri and Peutch, 1982,], in order to test the basis hypothesis that the weights of the $n$ objects to be loaded in the sack are statistically independent variables, no matter what their distribution, we must extract from the real situation referred to by the problem, a sample whose cardinality is $0(2^n)$.

Faced by this situation in which it is difficult to define efficient heuristics precisely because one does not have the means of evaluating their efficiency, we intend, in the next few pages to propose particular heuristics which **independently of the values of the exact solutions,** show interesting parameters of efficiency that are testable via distribution-free methods using small size samples. In practice, we start from very simple and obvious heuristic rules of sack loading which, however, are administrated with particular care, in the light of the previous considerations made on choice steps, and within fixed limits on the running time. The time available is the main parameter which conditions such heuristics and in function of its handling we will distinguish between *fixed time and cut time heuristics.*

## 2. Fixed time heuristics

Fixed time heuristics are one shot algorithms which follow only one of the computational paths arising from the choice points, so that essentially they consider only one feasible solution by loading the objects in the sack once. Their running time is essentially fixed by the algorithm itself and is generally a very

shortvalued function of the size of the input with respect to the lower bound coming from the complexity of the problem.
Given such an algorithm, when more than its running time is available, complicating the loading rule gives no guarantee about an actual improvement on the solution of the special instance of the problem in hand. So we suggest on the contrary solving the same problem more times by using trivially easy rules which however differ noticeably from each other. In this way we generally expect to obtain significantly different approximate solutions among which the one which comes closest to filling the sack is selected. This strategy gives a more rational theoretical framework to those algorithms which Kornefield [Kornefield, 1982] call combinatorially implosive. In the latter case the various paths coming from the different solution rules cooperate by running in parallel and by exchanging by-product information. Here the simplicity of the problem avoids those exchanges which may prove inefficient.
More particularly, starting from the above greedy algorithm HGK, we construct HGK1:

```
FUNCTION IND (I : integer); external;
BEGIN
    SUM := 0; I := 0; X (I) := 0 for each I;
    WHILE SUM ≤ CAPACITY DO
    BEGIN
        I := I + 1;
        SUM := SUM + W (IND (I));
        IF SUM ≤ CAPACITY
        THEN X (IND (I)) := 1;
    END;
END.
```

This differs from the above by the presence of the function IND(1), and we change the selections in the choice points by simply changing the function IND, i.e. the ordering of the objects.
We examine the characteristics of the algorithms so generated by using the Monte Carlo approach and by varying the input population with regard the number of items, the law of distribution of their sizes, and the sack capacity. Actually, according with the remark of the previous section, when one is dealing with combinatorial problems, a lot of experiments should in general be needed in order to infer some non trivial sentence about their solutions, since the inherent non determinism of those problems induces great spread in the main features, i.e. running time or waste, of the solving algorithms. In this framework a fifty item sized sample and then a thousand item sized sample are equally small.
Therefore our strategy will focus on two kind of experiments: i) those aimed at negating some sentence, so that even a single item should disprove the sentence,

ii) those tailored to simulate special sample statistics whose distribution laws are known to be little spread.

Namely, our sample size will range between 30 and 80, while the number $n$ of objects of the single knapsak problem will be always 50.

Three kinds of distribution law of the weight will be employed:

— uniform within 0 and 100, namely $W = U \times 100$, where U is uniform in (0,1)

— quadratic, where $W = U^2 \times 100$

— quare root, where $W = \sqrt{U \times 100}$

As it was previously pointed out, a usual statistical assumption is that the weights are independent random variables. Such a hypothesis, tough extremely simplifying, is very often totally unrealistic, and hardly testable. Therefore simulated random samples where the correlation between two consecutive items is a random number between — 1 and 1 are rare.

Last, we distiguish the instances on the basis of three capacities of the sacks so computed:

$$\sum_{i=1}^{n} w_i/n \qquad .5 \sum_{i=1}^{n} w_i \qquad .8 \sum_{i=1}^{n} w_i$$

As concerns the algorithms, in HGK we distinguish two main classes of sets of object orderings which we call global and local ordering sets. Whether or not an ordering belongs to one of these types of sets depends on the tie it has with the other elements of the set. To be precise, in global orderings the rank of almost every object in the input changes substantially when passing from one order to another, whereas in local orderings we make only slight variations in the rank of the objects.

The boundary line between the two classes is fuzzy. Following in some sense Kolmogorov [Kolmogorov 1964], we could employ the sum of the lengths of the shortest programs which compute the orders as a measure of globality: the smaller the sum, the more global the set of orderings. In these numerical experiments, we consider the following to be within global orderings set: i) the decreasing order, ii) the increasing, iii) the random, iv) the alternating, (i.e. first the highest weight, then the lowest, then the highest among the remaining, then the lowest among the remaining and so on).

Within the local orderings set, starting from any order we state the other orderings obtained by rules like these: i) change the median item with next item, ii) change the two median items with their neighbouring items and so on. In what follows the local orderings are variants of the decreasing one.

The most relevant conclusions are the following:

a) Behaviour of the single algorithm

As shown in Fig. 1, the distribution law of the wastes is strongly dependent on the distribution law of and the correlation between, the objects to be loaded,

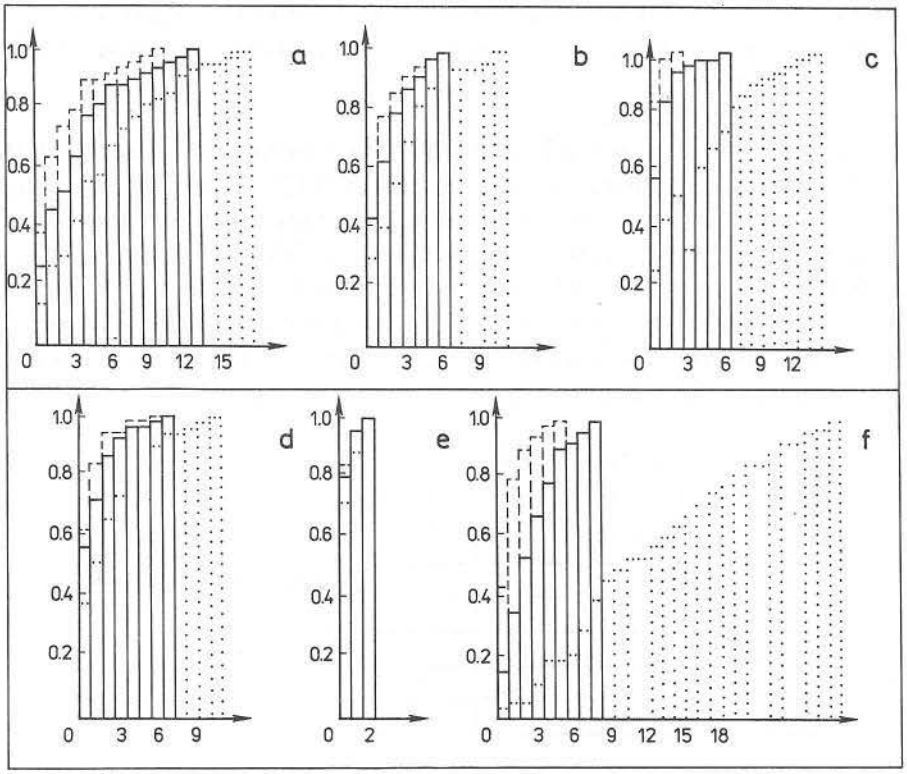


Fig. 1 Sample cumulative distribution function of wastes
Number of objects = 50.
____ increasing ordering
------ decreasing ordering
..... random ordering
1.a Sampling size (ss) = 30, sack capacity (sc) = 0.5* $\Sigma w_j$, weights distribution law (wdl) = U* 100 (U = 0-1 uniform random var.) correlation (corr) = always 0
1.b ss = 60, sc = 0.5 $\Sigma w_j$, wdl = $\sqrt{\text{U}}$* 100, corr. = always 0
1.c ss = 90, sc = 0.8 $\Sigma w_j$, wdl = U* 100, corr. = always 0
1.d ss = 60, sc = 0.5 $\Sigma w_j$, wdl = U* 100, corr. = always 0
1.e ss = 90, sc = 0.5 $\Sigma w_j / n$, wdl = U* 100, corr. = always 0
1.f ss = 30, sc = 0.5 $\Sigma w_j$, wdl = U* 100, corr. = random among the pairs

whatever the the ordering should be. So any probabilistical statement is not quickly testable, even if the general opinion is that decreasing order is the best.

b) Connection between the algorithms

When we are investigating the joint behaviour of the algorithms rather than the individual patterns of the solutions, we are focusing on the structure of the algorithm itself. Thus we disregard the single computational paths connected to the choice, and the law of distribution of the input, so that the relevant properties can be tested by short samples.

The statistic which has been dealt with is the KENDALL Rank Correlation Statistic $\rho$. Starting from a set of objects, its construction is based on a two-way table of experiments.
1st way: capacity of the sack
2nd way: ordering (that is IND) with which the objects are loaded by HGK1
Following the same illustrative strategy as Kendall [Kendall 1975] we imagine that the first way is at $n$ levels and that at each of these levels we can replace the capacity of the sack by an item participating in a qualifying competition. Just like the members of a jury, the orderings of the second way, in number of $m$, award different scores, ranking from 1 to $n$, in function of the waste in the solutions they themselves generate. To be exact, the scores are the decreasing ranks of the wastes examined by the same jury.

ITEMS n = 3

| JURY | 1st way / 2nd way | $I_1$ | $I_2$ | $I_3$ | |
|---|---|---|---|---|---|
| | $0_1$ | 2 | 1 | 3 | $Waste_{13} < Waste_{11} < Waste_{12}$ |
| | $0_2$ | 1 | 2 | 3 | |
| | $0_3$ | 1 | 3 | 2 | |
| m = 4 | $0_4$ | 2 | 3 | 1 | |
| | | $t_1$ | $t_2$ | $t_3$ | $\rightarrow t_a$ |

The main question is whether the juries are substantially concordant, and therefore if the orderings give rise more or less to the same pattern of the wastes in relation to the sack capacities, so that the capacity which leads to the minimal waste within one particular ordering turns out to give virtually minimal waste within the other orderings too, and so on.
Starting from the above $m$ x $n$ table the Kendall statistic $\rho$ is computed as:

$$\rho = \frac{12\ S}{m^2\ (n^3-n)}$$

where $S = \sum_{i=1}^{n} (t_i - t_a)^2$, $t_i$ is the sum of the scores of the i-th sack in correspondance with the various orderings and $t_a$ is the arithmetic mean of the $t_i's$.
By definition, when these scores are concordant the sum $S$ tends to take on high values, and vice versa.
In the hypothesis that the scores disagree so completely that every possible score table has the same probability, the distribution law of $\rho$ is totally free [Wilks 1962] of the distribution law of the scores, and therefore of the distribution law of the wastes, and consequentely of the distribution law of the variables in input to the problem. In Kendall's book the cumulative distribution function of $\rho$ in such a hypothesis is available as a function of $m$ and $n$.

We simulated the law of $\rho$ in the case of global orderings and in the case of local orderings, varying the capacities of the sacks according to the afore mentioned formulas.

On the basis of lots of 60 problems randomly sampled we found that the cumulative histogram of $\rho$ coincides almost completely with the above theoretical distribution law, when dealing with global orders. Vice versa, Fig., 2 the histogram goes to the highest values when the algorithms differ by local orders (see Fig. 2).

Thus the global orderings are not in agreement and consequently by using them we can expect to find approximate solutions which differ in the way previously described. Therefore in those cases where an ordering turns out to be of low efficiency, giving a low score, there will be other orderings of the same set which give a higher one, and which could correspond to lower wastes.

This relation between kind of ordering and uncorrelation should be independent of the joint distribution of the objects, with the result that not only the test statistic but also the property itself appears to be distribution free, and consequently we could assume that global orderings give rise to totally discordant results. Anyway the property can be tested over the special distribution law the set of inputs is coming from.

We now carry out the variance analysis on the same experiments, taking as variate the global orderings and as covariate the capacity of the sacks and using the usual Fisher's statistic:

$$F = \frac{(N-r)\, X'A(A'A)^{-1}C'\{C(A'A)^{-1}C'\}C(AA')^{-1}}{(r-1)X'[I-A(A'A)^{-1}A']X}$$

where the symbols have the usual meaning of the ANOVA [Morrison 1967]. As shown in Fig. 3 the shape of the histogram of this statistic is distorted towards the highest values, thus denouncing a significant difference between the various orderings. Therefore, since the hypothesis that the wastes are distributed in a gaussian curve is one of maximum entropy [Hamming 1980] we may suppose that a significant difference exists between the mean wastes pertaining to various orderings, such that one particular ordering may turn out to be on the average better than the others. With respect to these results, the preceding numerical experiments underline two points:

a) Given the shape of the histograms of Fig.1, and the structure of the greedy algorithms, the characteristics of the algorithm with the least mean waste depend directly on the joint distribution law of the objects and above all on their mutual correlations. As we have already said, such correlations are lengthy to test, so that, for a given class of real problems, e.g. from the world of the industry, it is difficult to ascertain how much a given ordering is better in mean value.

b) Given a set of global orderings, Kendall's statistic suggests that it is very useful to process the input using all these orderings. Indeed, provided that the

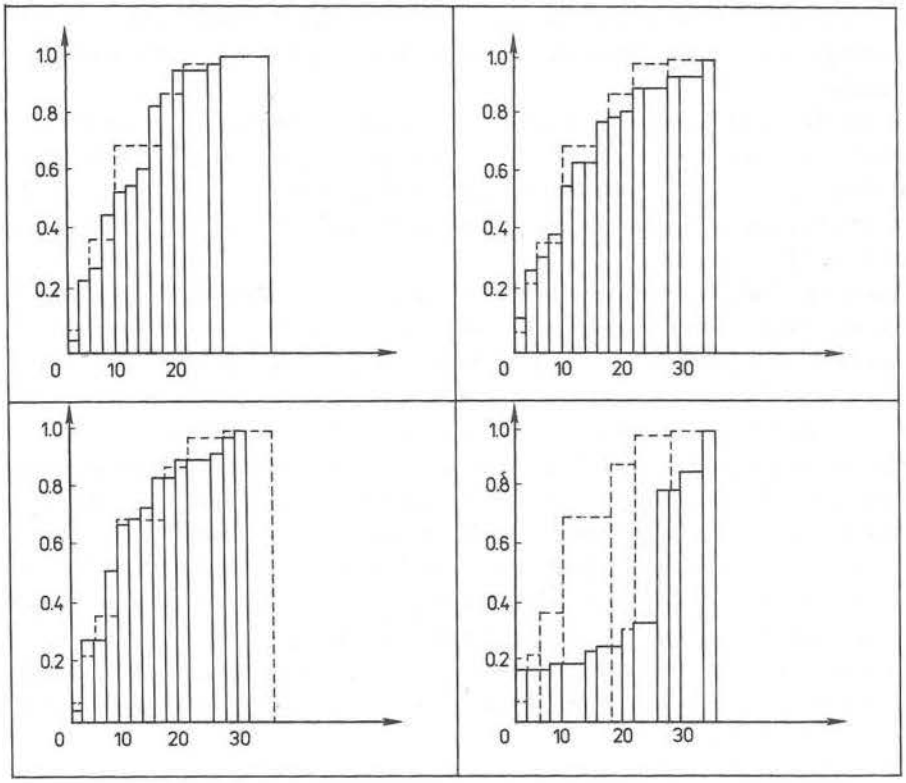


Fig. 2 Theoretical and sampled cumulative distribution function of the Kendall statistic.
Number of objects = 50.
—— sampled distribution function
------ theoretical distribution function
Experiment Plan:
Objects: 1) $\Sigma w_j / n$, 2) $0.5 * \Sigma w_j$, 3) $0.8 * \Sigma w_j$.
Observers
global orders: 1-decreasing order, 2-increasing order
3-random order, 4-alternating order
local order: 1-decreasing order
2-the same as 1 but the median item changes place with the next item
3-the same as 1 but the two items leftside closest to the median change with the right side items
4-the same as 1 but the three...
2.a ss = 60, wdl = U*100, ord. = global, corr. = always 0
2.b ss = 60, wdl = $U^2$*100, ord. = global, corr. = always 0
2.c ss = 60, wdl = U*100, ord. = global, corr. = random among the pairs
2.d ss = 60, wdl = U*100, ord. = local, corr. = always 0

mean wastes relative to these are not enormously different, we could expect that, no matter what the best ordering may be, in those cases when this gives a very high waste some other ordering of the same set can give better results.

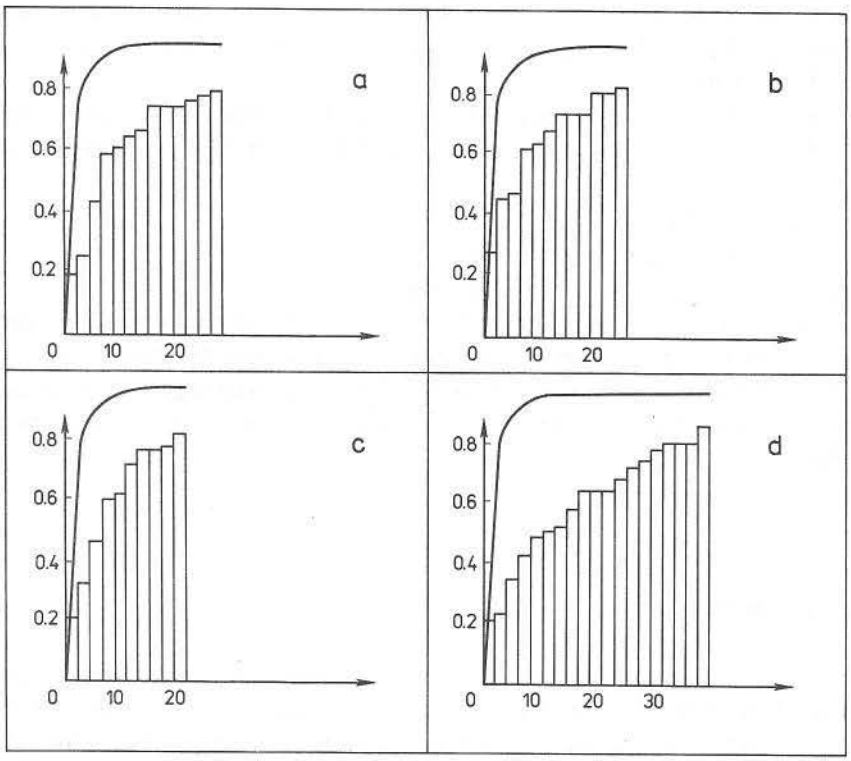


Fig. 3 Theoretical and sampled cumulative distribution function of the ANOVA statistic
Number of objects = 50.
Experiment Plan:
factor levels: 1-random ordering
2-decreasing ordering
3-incteasing ordering
Covariate values: $\Sigma w_j / n$, $0.5 * \Sigma w_j$, $0.8 * \Sigma w_j$
3.a ss = 60, wdl = U*100, corr. = always 0
3.b ss = 60, wdl = $\sqrt{U*100}$, corr. = always 0
3.c ss = 60, wdl = $U^2*100$, corr. = always 0
3.d ss = 60, wdl = U*100, corr. = random among the pairs

### 2.1. Proposed procedure

On the basis of the previous theoretical considerations and of the numerical experiments shown, the following fixed time heuristic procedure is proposed: When one has to solve a set of 0-1 knapsack problem with unitary profit coefficients and a fixed short time is available for each,
a) select a number of highly uncorrelated fixed time algorithms so that the tota! running time equals the available time.

a.1) in order to select highly uncorrelated algorithms start from a fixed time heuristic algorithm, like HGK1, and change the ordering of the objects within a global set of orderings.
a.2) in order to be sure that these orderings, and therefore these algorithms are uncorrelated test them by using the $p$ Kendall statistic on a sample of those problems.
b) if a.2 succeds then
b.1) execute the selected algorithms
b.2) select the solution which fills the sack most.
Of course this proposed procedure represents an example of a general procedure meant for obtaing approximate solutions of highly complex problems.
Note that when we test the un-correlations between orderings, we are assuming that the instances of the problem come all from the same random population we pick the sample from.

## 3. Cut time heuristics

When a limited amount of time is available to obtain a solution to the knapsack problem, then each algorithm really behaves as an approximate algorithm, if this time, as a function of the input length, is less than the lower bound coming from the complexity of the problem. Indeed, we can only be sure that we have got an optimal solution in trivial cases, and, we are almost never sure in advance that this optimal solution will be achieved, so that in most cases we must deal only with feasible solutions again. Consequently, another class according to which we may group our solving algorithms is that of cut-time. In these algorithms we must take into account the fact that their running time my be cut before completion and that, consequently, the algorithms must handle a current solution, to be used as an approximate solution at the cutting moment.
Within this framework, a meaningful parameter by which to appraise the efficiency of the algorithms, no matter if exact or approximate, is the rate at which the best feasible solution comes close to the capacity of the sack. Here once again we take the capacity of the sack and not the optimal solution as our touchstone in order to avoid the lower bounds on the complexity stated in sect. 1.
We propose a heuristic algorithm which seems to show a more favourable rate of improvement of the feasible solution than the usual enumerative algorithms.
In this case we are unable to supply statistics by which some theorem proves to be distribution free; however, the histograms of the above-mentioned rate display some characteristics which, experimentally, appear to be independent of the law of distribution of the input.
The heuristic algorithm CTHK that we propose starts from the following idea: among all the feasible solutions we focus only on those which, starting from

a greedy solution, are produced by the substitution of one or two objects by another, or of one object by two. Obviously, at the moment of each of these substitutions, the best current solution is the best feasible solution available. Of course these are not all the feasible solutions, but they come close to the capacity of the sack, and thanks to the substitutions of one item by two or vice versa, constitute a wide spectrum of candidate solutions.

In order to make this algorithm exact, we ought to extend the heuristics rules to take into considerarion not only those substitutions of order $\leq 2$ already mentioned but all possible substitutions. This kind of extension, however, would require considerable running time, while the number of substitutions that cannot be broken down into substitutions of order $\leq 2$ by our algorithm is generally small.

More precisely the algorithm is made up of three main steps:

**Upward trials**

Starting from the feasible solution, one object inside the sack is substituted by one or two object outside in order to reach the capacity of the sack. If the sack cannot be entirely filled in this way the best solution update the current solution, but the trials continue performing the first tested exchange, in order to prevent any sub-optimal strategy and obtain a wide spectrum of feasible solutions.

**Downward trials**

In this case we start from unfeasible solutions that exceed the capacity of the sack and try to reach the exact capacity by a strategy symmetrical to the previous one. This step does not update the current solution unless the sack is completely filled up.

**Initialization**

This occurs at the beginning of the algorithm and when one of the previous steps cannot continue. At the beginning it consists of a special greedy algorithm. At the end of the first step a particular substitution is performed in order to jump outside the sack capacity, and a similar inverse jumping is performed at the end of the second step. When these jumps cannot be made we start again from the beginning, making sure we avoid passing by the same initial solution.

The Pascal code of the algorithm is reported in the quoted book [Apolloni et al. 1984], here we show its outline.

**Variables**

Scalars:

$n$ = number of objects
CAPACITY = capacity of the sack
$r$ = current waste/surplus
CSUM = initial sum of loaded weights
STFLAG = flag of the step
Vectors:
IND = current solution
BTSOL = best current solution

ETSOL = exact solution
W = weight of the objects
$W_{in}$ = weight of the loaded objects, namely $W_{in}$ (I) = W (I)* IND (I)
$W_{out}$ = weight of the outsack objects, namely $W_{out}$ (I) = W (I)* (1-IND (I))
USIND = objects not useful in the initialization step

**Algorithm**

Step 1: INITIALIZATION
if it is the first pass
then
order the objects in a growing order on
1. the minimum factor $\neq$ 1 of their weights
1.1. the value of the weights having thesame minimum
factor;
USIND (I) := 0 $\forall$ I;
IND (I) := 0 $\forall$ I;
I := 0; CSUM := 0;
repeat
I := I + 1;
if USIND (I) = 0
then CSUM := CSUM + W (I), IND (I): = 1;
until CSUM $\geq$ CAPACITY;
if CSUM < CAPACITY {because of USIND}
then PRINT BTSOL, STOP;
if CSUM = CAPACITY
then PRINT ETSOL, STOP;
else
USIND (I−1) := 1;
I := 0;
repeat
I := I + 1;
CSUM = CSUM−(1−IND (I))* W (I), IND (I) = 0;
until CSUM $\leq$ CAPACITY
if CSUM = CAPACITY then PRINT ETSOL, STOP;
r := CAPACITY−CSUM;
STFLAG := 1;
STEP 2: UPWARD/DOWNWARD TRIALS
a) if $\exists$I: $W_{out}$ (I) = r then Ind (I):= STOP, PRINT ETSOL;
if $\exists$ I, J: $W_{out}$ (I) = r + $W_{in}$ (J), I $\neq$ J
then IND (I) := 1, IND (J) := 0, PRINT ETSOL, STOP;
if $\exists$ I1, I2, J: $W_{out}$ (I1) + $W_{out}$ (I2) = r + $W_{in}$ (J), I1$\neq$I2$\neq$J

then IND (I1) := 1, IND (I2) := 1, IND (J) := 0, PRINT ETSOL, STOP;
if $\exists$ I1, I2, J: $W_{out}$ (I1) + $W_{out}$ (I2) < r + $W_{in}$ (J) I1 $\neq$ I2 $\neq$ J
then
if STFLAG = 1 then
find max $W_{out}$ (I1) + $W_{out}$ (I2) $\forall$ I1, I2, J s.t.
$W_{out}$ (I1) + $W_{out}$ (I2) < r + $W_{in}$ (J) I1 $\neq$ I2 $\neq$ J;
update BTSOL;
take the lowest I1, I2, J s.t.
$W_{out}$ (I1) + $W_{out}$ (I2) < r + $W_{in}$ (J) I1 $\neq$ I2 $\neq$ J;
IND (I1) := 1, IND (I2) := 1, IND (J) := 0;
if IND so obtained equals IND corresponding to BTSOL;
then PRINT BTSOL, STOP;
else go back to the start of this step;
else go to step 3;
STEP 3: CHANGE OF DIRECTION OF THE TRIALS
find max $W_{out}$ (I) + $W_{out}$ (k) and min $W_{in}$ (J) $\neq$ 0, $\forall$ I1, I2, J, I1 $\neq$ I2 $\neq$ J;
if $\exists$ I or K {all the objects are inside}
then go to STEP1
else IND (I) := 1, IND (K) := 1, IND (J) := 0;
r := ABS (CAPACITY–CSUM));
IND (I) := 1–IND (I) $\forall$ I;
STFLAG := 1–STFLAG;
go to STEP 2;

In order to assess the validity of this heuristic with respect not, we repeat, to the approximation but to the improvement rate, we compared that rate with that of the algorithm EEK1, obained from EEK via a modofication of the first two steps.

**Generation of all the subsets of the input objects**

This corresponds to generation of all the binary strings $s$ of length $n$, in the opposite lexicographical order.

**Cutting of the unfeasible solutions**

This step allows the cutting out of entire sets of unfeasible solutions. Provided that the items are ranked according to the increasing order of their weights, the rule is the following:
As soon as in a string $s$ $\exists$ $k$ s.t. $\sum_{j=1}^{k} w_j s_j$ exceeds $C$,

put $s_k = 0$ and continue from the corresponding string the generation of the strings.

Moreover the task of step 1 is obtained by the following very easy algorithm [Johnson 1963]

1. Start with a string of all 1
2. If items $s_j = 1$ do not exist then stop

3. Starting from the rigth, find the first item = 1
4. Set this item to 0 and all the items at its right side to 1
5. Go to 2

Thanks to the simplicity of its structure, among the exact algorithms for 0-1 Knapsack problems, EEK1 appears to be very efficient.

For simplicity of illustration the above mentioned rate has been evaluated in an indirect way by sampling problems, in which the exact solution completely filled the sack and by tracing the cumulative distribution function of the time at which the full capacity of the sack is attained both from CTHK and from EEK1 (see Fig. 4).

The better behaviour of the first algorithm, which is evident in each of the various cases simulated, leads us to conjecture that the fact that the first algorithm has a better rate, could be true almost independently of the law of distribution of the input, just as the experiments of Fig. 1 allow us conjecture that the decreasing ordering is generally the best one. Moreover, thanks to the cutting strategy of the exact algorithm, the benefit which comes from using the heuristic algorithm is greater when the distribution law of the weights is distorted toward the lowest values with respect to the uniform distribution.

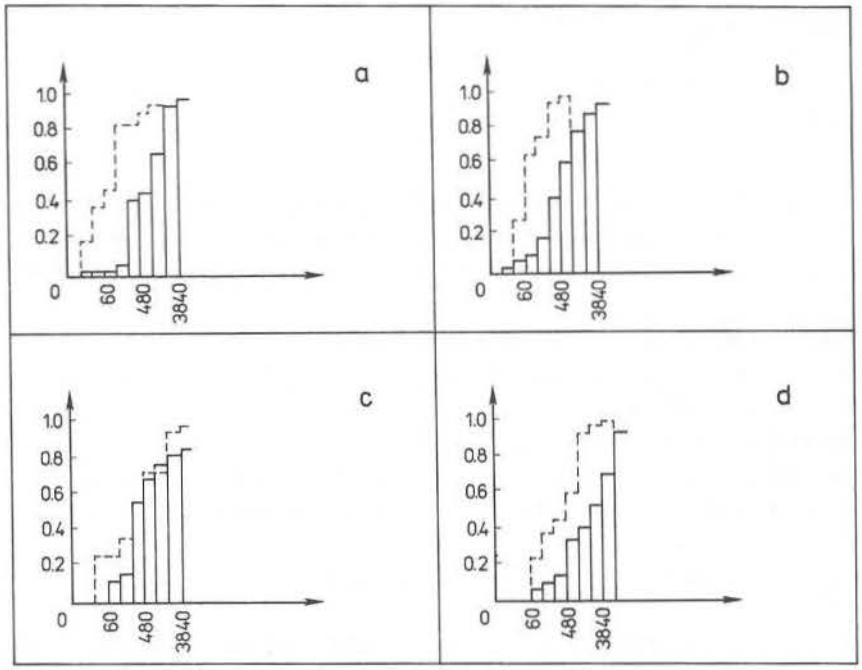


Fig. 4 Sampled cumulative distribution function of the running time of the full capacity solutions Number of objects = 50. ------ Cut time heuristic algorithm —— Enumerative exact algorithm sack capacity = mixed among $\Sigma w_j/n$, 0.5 $\Sigma w_j$, 0.8 $\Sigma w_j$ 4.a ss = 40, wdl = U*100, corr. = always 0 4.b ss = 40, wdl = $U^2$*100, corr. = always 0 4.c ss = 40, wdl = $\sqrt{U}$*100, corr. = random among the pairs 4.d ss = 80, wdl = U*100, corr. = always 0

## 4. Conclusions

Together with the demand for optimality, the need to perform choices emerges in the decision problems when the problem is computationally complex and the size of the input is large. In these situations, when the probability theory is useless, there is room for such theories as fuzzy set theory [Baas and Kwakernaak 1977] alternative sets [Vopenka 1979] or choice theory [Apolloni and Di Gregorio 1983] in the context of classical or multimodal logics [Lukasiewicz 1953.]
The problem is to preserve this demand for optimality in the presence of an operator, such as the choice operator, which, by definition, breaks all the causality chains, and thus the optimality chains.
In this paper, in line with a developing theory of choice and from a strictly pragmatic point of view, we have proposed two approaches to using the available running time more profitably in solving a Subset-Sum Problem in an approximate way.
The interest of such algorithms does not reside so much in the sack loading rules as in the management of these rules which takes into account the presence of the choice operator, with the aim of obtaining, within the constraints of fixed time, a significant spectrum of approximate solutions. The quality of these algorithms lies in the fact that, in order to appreciate their efficiency, we may refer to some statistics that we prove, or at least we strongly conjecture in some cases, to be distribution free. Thus the efficiency of these algorithms is effectively testable by means of small sized samples coming from the actual operational problem we are dealing with.

## References

[1] Apolloni B., Di Gregorio S. About a formal non probabilistic theory of choices. Proc. 7-th Int.Congr.of Logic, Methodology and Philosophy of Sciences. Salzburg 1983, 115–119.
[2] Apolloni B., Di Gregorio S. Combinatorial problems in Natural Science. Computational complexity and inherent properties. *Epistemologia*, 7 (1984), 1–30.
[3] Apolloni B., Pittelli A. Complex statistical test on computational complex problems. *Quaderni di teoria degli algoritmi*, (1984), 1–24.
[4] Apolloni B., Pezzella F., Canale D., Pizzuti C. Il problema di knapsack: raccolta di programmi di calcolo. Progetto Finalizzato transporti (Sottoprogetto II), (1984), 1–122.
[5] Baas M., Kwakenaak H. Rating and ranking of multi-aspects alternatives using fuzzy sets. *Automatica*, 13 (1977), 47–58.
[6] D'Atri G., Puech C. Probabilistic analysis of the subset-sum problem. *Discrete Applied Mathematics*, **4** (1982), 329–334.
[7] Gens G.V., Levner E.V. Fast approximation algorithms for knapsack type problems. Lecture notes in Control and Information Sciences, **23** (1980), 185–194.
[8] Hamming R.W. Coding and information theory. New York, Prentice Hall 1980.
[9] Ibarra O.H., Kim C.E. Fast approximation algorithms for the knapsack and sum of subset problems. *J.ACM*, **22 (1975),** 463–468.
[10] Johnson S.M. Generation of permutation by adjacent transposition. *Math.of Comput.* (1963), 282–285.

[11] KARP R.M. Probabilistic analysis of partitioning algorithms for the travelling salesman in the plane. *Mathematics of Operations Research*, **2** (1977), 209–224.
[12] KENDALL M.G. Rank correlation methods. London, Ch. Griffin, 1975.
[13] KOLMOGOROV A.N. Three approaches to the quantitative definition of information problems.*J.of Inform. Transm.* **1**(1965), 4–7.
[14] KORNFELD W.A. Combinatorially implosive algorithms. *Communications of the ACM*, **25** (1982) 10, 734–738.
[15] LAWLER E.L. Fast approximation algorithms for combinatorial problems. *Mathematics of Operation Research*, **4** (1979), 339–356.
[16] ŁUKASIEWICZ A system of modal logic. *J.Comp.Syst.*, (1953), 111–149
[17] MARTELLO S., TOTH P. Worst-case analysis of greedy algorithms for the subset-sum problem. *Mathematical Programming*, **28** (1984), 198–205.
[18] MORRISON D.F. Multivariate statistical methods. New York, Mc Graw-Hill, 1967.
[19] MULLER-MERBACH H. Heuristics and their design, a survey. *Eur.J.Op.Res.*, **8** (1981), 1–23.
[20] RABIN M.O. Probabilistic Algorithms. In: Algorithms and complexity, new directions and recent trends. J.F.Traub, ed. New York 1974.
[21] VOPENKA P. Mathematics in the alternative set theory.Amsterdam, North-Holland 1979.
[22] WILKS S.S. Mathematical statistics. New York, J.Wiley 1962.

*Received, December 1988.*

**Testowalne heurystyki dla binarnego zadania załadunku**

Dla efektywnego rozwiązywania binarnego zadania załadunku przy ograniczonym czasie obliczeń zaproponowano dwie heurystyczne metody konstruowania rozwiązań zwane „fixed time" i „cut time".

Podstawową ideą jest tu konstrukcja takich algorytmów aproksymujących które generują, w ograniczonym przedziale czasu, istotnie różniące się między sobą rozwiązania dopuszczalne w oparciu o te cechy problemu, które są zwykle pomijane w znanych heurystykach używających reguł suboptymalnych przy załadunku „plecaka". W przypadku zaproponowanych algorytmów można udowodnić, że są one efektywne względem pewnych parametrów niezależnych od struktury rozwiązania dokładnego, oraz że współczynnik efektywności może być oszacowany za pomocą metod bezdystrybucyjnych, na podstawie małych próbek generowanych przez rozpatrywane zagadnienie załadunku.

## Тестируемые эвристические методы для бинарной задачи загрузки

Для эффективного решения бинарной задачи загрузки, при ограничении времени на вычисления, предлагаются два эвристических метода построения решений, называемые „fixed time" и „sut time".

Основная идея состоит в построении таких аппроксимирующих алгоритмов, которые генерерируют, на ограниченном отрезке времени, существенно отличающиеся между собой допустимые решения на основе тех факторов, которые обычно не берутся во внимание в известных эвристических подходах, использующих субоптимальные правила при загрузке „рюкзака". Для случая предлагаемых алгоритмов можно доказать, что они эффективны по отношению к некоторым параметрам, независимым от структуры точного решения, а также что коэффициент эффективности может быть оценен с помощью нераспредемительных методов, на основе малой выборки генерируемой рассматриваемой задачей загрузки.