STOCHASTIC INTEGER PROGRAMMING BY DYNAMIC PROGRAMMING

by B.J. Lageweg, J.K. Lenstra, A.H.G. Rinnooy Kan and L. Stougie

Abstract Stochastic integer programming is a suitable tool for modeling hierarchical decision situations with combinatorial features. In continuation of our work on the design and analysis of heuristics for such problems, we now try to find optimal solutions. Dynamic programming techniques can be used to exploit the structure of two-stage scheduling, bin packing and multiknapsack problems. Numerical results for small instances of these problems are presented.

Key words: *stochastic integer programming, distribution model, two-stage decision model, scheduling, bin packing, multiknapsack, dynamic programming.*

1 Introduction

Stochastic programming problems are mathematical programming problems of which not all the parameters are known with certainty. It is usually assumed that the unknown parameters have a known probability distribution. There are two types of optimization models in stochastic programming.

The first one is the *distribution model*, in which one has to determine an optimal decision for each realization of the stochastic parameters. As a result, one obtains the complete probability distribution of the optimal solution value to the stochastic programming problem. This is the 'wait and see' approach, in which the decision is made only when perfect information is available. It is of largely theoretical interest.

The second model is the *two-stage decision model*, in which one has to determine a decision that is optimal in expectation. In evaluating a decision, one takes into account the costs of a recourse decision that may be taken when, at a later stage, the realization of the stochastic parameters becomes known. This is the 'here and now' approach, in which the decision is made given imperfect information. The concept of a recourse decision should be interpreted broadly. It includes not only emergency actions in case of particularly unfortunate realizations but any action that is appropriate under the given circumstances.

Research in this area is so far almost exclusively concerned with *stochastic linear programming*. In the distribution model, each realization of the stochastic parameters leads in this case to an ordinary linear program. In the two-stage decision model, the recourse problem at the second stage is a linear program, the right-hand side of which is usually determined by the overall decision at the first stage. This implies that the expected second stage cost is a convex function of the first stage decision variables (WETS (1983)). Successful algorithms for the linear two-stage problem heavily exploit this convexity property.

By imposing integrality constraints on some of the decision variables we enter the area of *stochastic integer programming*. The complexity of the problems under

consideration is thereby increased dramatically. One reason for this is that linear programming is well solved while integer programming is NP-hard, so that the solution of a deterministic subproblem may require much more time. Another reason is that, in the two-stage decision model, integrality of the second stage decision variables can cause nonconvexities and even discontinuities in the expected second stage cost function (BLAIR and JEROSLOW (1982)), so that the algorithms for the linear case cannot be adapted.

Stochastic integer programming does not only present many theoretical challenges, it is also a practical tool for modeling certain *hierarchical decision situations* that arise in operations management planning and control. Such situations require a series of decisions over time at an increasing level of detail and with an increasing amount of information being available. At least two decision levels can usually be recognized: an *aggregate level*, at which one has to decide upon the acquisition of resources, given vague information about what certain tasks will require of them, and a *detailed level*, at which one has to decide upon the allocation of resources to tasks, given precise information about the requirements. Integrality constraints may appear at the first level, when the resources come in discrete units only, and at the second level, when the allocation problem is of a combinatorial nature.

Given the formidable difficulty of stochastic integer programming, most research in the area has so far concentrated on the design and analysis of *approximation algorithms*. This approach is exemplified in (DEMPSTER et al. (1981), (1983); FRENK et al. (1984); MARCHETTI-SPACCAMELA et al. (1984)), where simple heuristics are proposed for a variety of two-stage production and distribution planning problems. Probabilistic analyses of the heuristics then provide exact statements about the quality of the approximations, such as some form of asymptotic optimality. A general framework for this approach is given in (LENSTRA et al. (1984)).

In this paper, we are interested in *optimization algorithms*. We will consider stochastic integer programs of a very special structure. The stochastic parameters will have a discrete distribution with a finite number of points with positive density. Moreover, each realization will lead to a combinatorial optimization problem that is solvable by a *dynamic programming* routine. The overall stochastic optimization problem will then be solved by a single giant recursion that combines the separate dynamic programming computations for all the individual realizations. This can be done only for problem instances of a relatively small size.

The following three sections illustrate our approach on two-stage *scheduling*, *bin packing*, and *multiknapsack* problems. In each section, we first formulate the problem in question, then present the dynamic programming algorithm and describe its implementation, and finally discuss our numerical results. The discussion includes a comparison with results obtained by heuristics for the scheduling and bin packing problems.

Our computational experience gives empirical insight into the shape of the value functions of stochastic integer programming problems and shows that the discontinuities and nonconvexities mentioned above do indeed occur. Further

investigations have shown that one has to distinguish between discrete and continuous distributions of the stochastic parameters; in the latter case, no discontinuities will occur under certain conditions (STOUGIE (1985)). The results we have obtained so far should be regarded as no more than a first step towards a theory for stochastic integer programming.

Boldface characters will denote random variables.

2 Scheduling

2.1 Problem formulation

The two-stage scheduling problem studied in this section was first formulated in (DEMPSTER et al. (1983)). At the aggregate level, one has to decide on the number X of *identical parallel machines* that are to be acquired, while knowing the *cost* c of a single machine, the number n of *jobs* that are to be processed, and the probability distribution of the vector $\omega = (\omega_1,...,\omega_n)$ of their *processing times*. It is assumed that the ω_j are independent and identically distributed random variables with expectation μ. At the detailed level, after X has been determined, a realization $\omega \in \Omega$ of ω becomes known, where Ω denotes the set of all realizations, and one has to decide on a *schedule* in which each machine processes at most one job at a time, job j is processed during an uninterrupted time period of length ω_j ($j = 1,...,n$) and no job is processed prior to time 0, so as to achieve a minimum value $Y^*(X,\omega)$ of the maximum job completion time. The total cost of the acquisition decision X and the optimal scheduling decision is denoted by $V^*(X,\omega) = cX + Y^*(X,\omega)$.

In the two-stage decision model, the objective is to determine a value $X^* \in \mathbb{N}$ such that the expected total cost is minimized:

$$EV^*(X^*,\omega) = \min_{X \in \mathbb{N}} \{EV^*(X,\omega)\}.$$

In the distribution model, the objective is to determine a function $X^\circ : \Omega \rightarrow \mathbb{N}$ such that for each $\omega \in \Omega$ the actual total cost is minimized:

$$V^*(X^\circ(\omega),\omega) = \min_{X \in \mathbb{N}} \{V^*(X,\omega)\}, \forall \omega \in \Omega.$$

Previous work on this problem concerned the design and analysis of a two-stage heuristic (DEMPSTER et al. (1983)). This heuristic sets the number of machines equal to the value of X that minimizes the *lower bound* $V^{LB}(X) = cX + n\mu / X$ on $EV^*(X,\omega)$ and assigns the jobs to the machines by a *list scheduling* rule. (In our computational experiments, we used the *longest processing time* rule, which puts the jobs on a list in order of nonincreasing processing times and successively assigns the next job on the list to the earliest available machine; this rule has a better worst case performance than arbitrary list scheduling (GRAHAM et al. (1979)). The relative error of the heuristic tends to 0 as n tends to infinity for various measures of stochastic convergence (LENSTRA et al. (1984)).

2.2 Dynamic programming

The second stage scheduling problem of determining $Y^*(X,\omega)$ for given X and ω is NP-hard (GAREY and JOHNSON (1979)). We will consider the situation in which the processing times can assume only k distinct values $a_1,...,a_k$, for a fixed value of k. Let us denote by $\omega = [n_1,...,n_k]$ the vector of processing times in which the value a_j occurs n_j times, for $j = 1,...,k$.

One can obtain an optimal schedule on X machines by assigning a certain subset of jobs optimally to $X-1$ machines and putting the remaining jobs on another machine. This observation leads to the following recurrence relations:

$$Y^*(X,[n_1,...n_k]) = \min\{\max\{Y^*(X-1,[n_1-l_1,...,n_k-l_k]),$$
$$Y^*(1,[l_1,...,l_k])\}$$
$$\mid 0 \leqslant l_j \leqslant n_j (j = 1,...,k)\} \quad (X > 1),$$

$$Y^*(1,[n_1,...,n_k]) = \sum_{j=1}^{k} n_j a_j.$$

Computation of $Y^*(X,\omega)$ by a dynamic programming algorithm based on this recursion requires $O(X\Pi_{j=1}^{k} n_j)$ time, which is exponential in k but polynomial for fixed k.

In the more general context of the two-stage scheduling problem, we assume that the processing times have a discrete distribution with k integral values $a_1,...,a_k$ in its support. The independence of the processing times implies that $\omega = [\mathbf{n}_1,...,\mathbf{n}_k]$ has a multinomial distribution. The idea is now to go through the entire recursion once in order to compute $Y^*(X,\omega)$ for all values $X \in \{1,...,n\}$ and for all realizations $\omega \in \Omega$, where Ω is given by

$$\Omega = \{[n_1,...,n_k] \mid 0 \leqslant n_j \leqslant n \ (j = 1,...,k),\ n_1 + ... + n_k = n\}.$$

The distribution model is then solved by the selection, for each $\omega \in \Omega$, of a value of X that minimizes $V^*(X,\omega) = cX = Y^*(X,\omega)$. The two-stage decision model is solved by the determination of a value of X that minimizes $EV^*(X,\omega) = cX + \sum_{\omega \in \Omega} Pr\{\omega = \omega\} Y^*(X,\omega)$.

A straightforward application of the above dynamic programming algorithm requires $O(n^k)$ comparisons for each of the $O(n^{k+1})$ pairs (X,ω), and hence $O(n^{2k+1})$ time altogether. The multinomial probabilities are easily computed within this time bound.

A more efficient implementation of the algorithm is obtained as follows. Let $a_1 = \max\{a_1,...,a_k\}$. It is not hard to see that, for any X and $\omega = [n_1,...,n_k]$

$$\lceil \textstyle\sum_{j=1}^{k} n_j a_j / X \rceil \leqslant Y^*(X,[n_1,...,n_k]) \leqslant \lceil \sum_{j=1}^{k} n_j a_j / X \rceil + a_1 - 1.$$

The lower bound is trivial, and the upper bound follows from the observation that any list scheduling algorithm will start every job strictly before the lower bound. Further, we assume without loss of generality that in the above recurrence relations the second maximand attains the maximum:

$$Y^*(X,[n_1,...,n_k]) = Y^*(1,[l_1,...,l_k]) \text{ for some } l_1,...,l_k.$$

We can therefore restrict our attention to vectors $[l_1,...,l_k]$ that yield a value $Y^*(1,[l_1,...,l_k])$ within a given range of a_1 integers. This implies that only a single value of l_1 has to be considered for given $l_2,...,l_k$ and that $O(n^{k-1})$ comparisons suffice for each pair (X,ω). The overall running time is thereby reduced to $O(n^{2k})$.

Other, more intricate, refinements lead to a running time of $O(n^{2k-1}a_1^{2k-3} \log na_1)$. Although that implementation is more efficient for small values $a_1,...,a_k$, it is of little avail in view of the results that will be presented in Section 3.2.

2.3 Computational results

The dynamic programming algorithm was coded in PASCAL and run on a CD Cyber 170-750 to solve several instances of the two-stage scheduling problem. The solution of instances with 100 jobs and two possible processing time values or with 50 jobs and three processing time values required about 30 seconds. The values of k considered are admittedly small, but the values of n are realistic and the running times are such that our brute force approach should not be dismissed on grounds of manifest inefficiency.

We illustrate the numerical results on a set of representative instances given by

$$c = 1,$$

$$n = 1,...,100,$$

$$k = 2,\ a_1 = 18,\ a_2 = 14,\ Pr\{\omega_j = a_1\} = Pr\{\omega_j = a_2\} = \tfrac{1}{2}\ (j = 1,...,n)$$

Figure 1 shows four functions of the number of jobs:
- the minimal lower bound $\min_X\{V^{LB}(X)\}$ mentioned in Section 2.1;
- the minimal expected total cost $EV^*(X^*,\omega)$ (the optimum for the distribution model, averaged over all realizations);
- the expected approximate total cost obtained by the heuristic mentioned in Section 2.1.

Note that the last three functions are defined only for integral n; linear interpolation has been applied to improve the presentation. The distribution model yields slightly better results than the two-stage decision model on average, as expected. A comparison between the optima and the lower and upper bounds confirms that the absolute differences are significant while the relative differences disappear with increasing problem size.

For the case that $n = 100$, Figure 2 shows three functions of the first stage decision variable, the number X of machines:
- the lower bound $V^{LB}(X)$;
- the expected total cost $EV^*(X,\omega)$ in case of an optimal second stage decision;
- the expected total cost in case of an approximate second stage decision.

Note that we have interpreted X as a continuous variable: acquisition of a

fractional machine costs a fraction of c but yields no benefit at the second stage; the vertical line segments correspond to discontinuities. In spite of the smoothing effect due to averaging over all realizations, both the optimal and the approximate cost functions are highly nonconvex and multimodal. The functions consist of a first stage component, which is linear and increasing, and a second stage component, which is nonconvex and nonincreasing. Addition of the two components can turn the nonconvexities into local minima, and small values of c appear to be most effective in this respect.

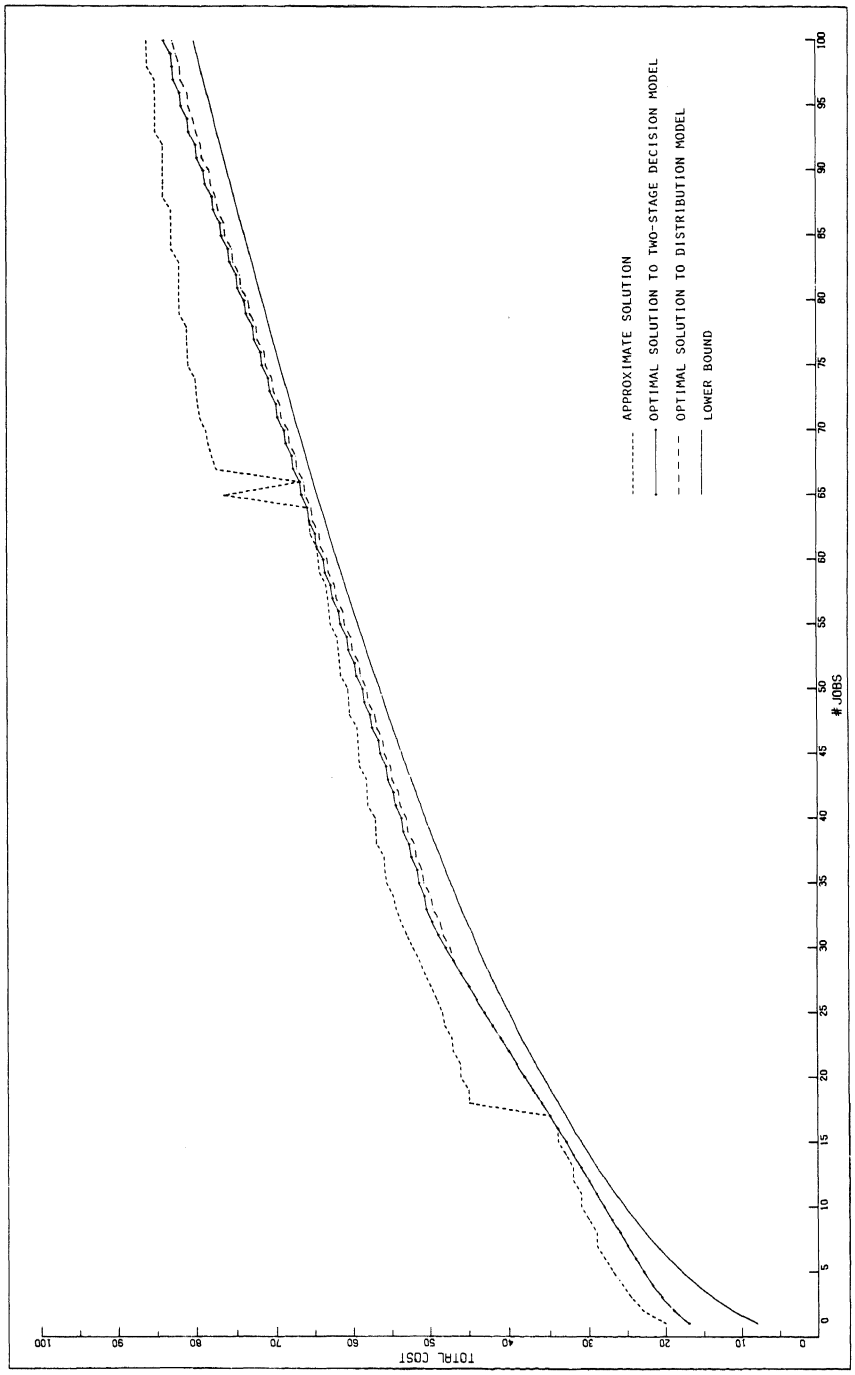

Figure 1. Scheduling: the total cost as a function of the number of jobs.

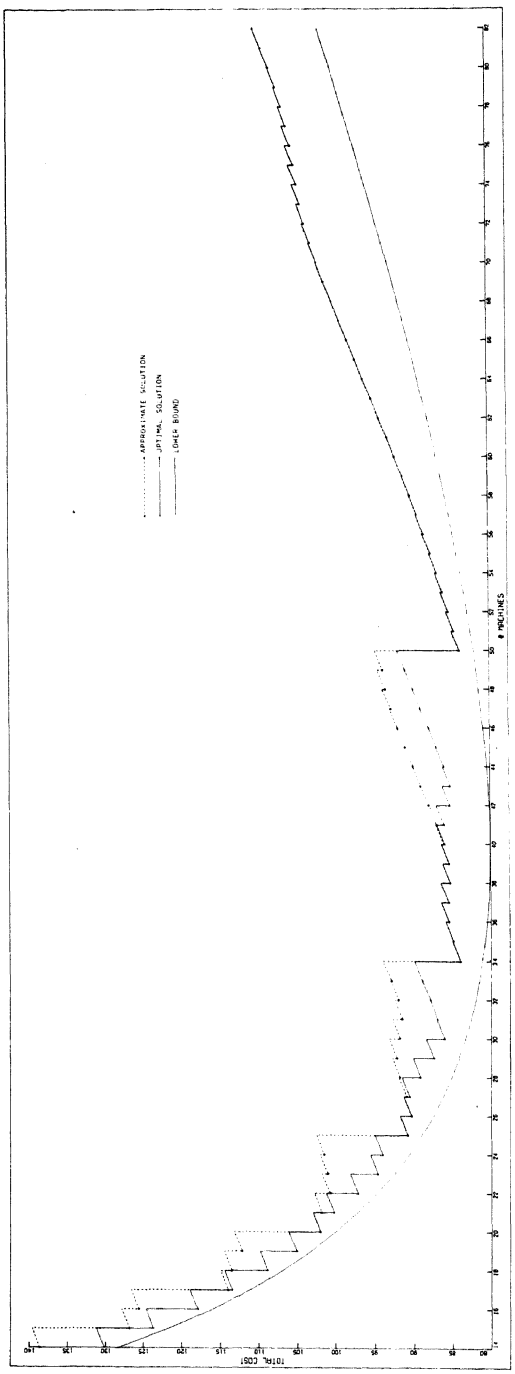

Figure 2. Scheduling: the total cost as a function of the number of machines.

3 Bin Packing

3.1 Problem formulation

The two-stage bin packing problem is formulated as follows. At the aggregate level, one has to decide on the *capacity* Y *of bins*, while knowing the *cost* d of one unit of capacity, the number n of *items* that are to be packed into the bins, and the probability distribution of the vector $\omega = (\omega_1,...,\omega_n)$ of the item *weights*. It is again assumed that the ω_j are independent and identically distributed random variables with expectation μ. At the detailed level, after Y has been determined, a realization $\omega \in \Omega$ of ω becomes known, and one has to decide on a *packing* in which each item is assigned to a bin and the total weight of the items assigned to the same bin does not exceed its capacity Y, so as to achieve a minimum number $X^*(Y,\omega)$ of bins needed. The total cost of the first stage decision Y and the optimal second stage decision is denoted by $W^*(Y,\omega) = dY + X^*(Y,\omega)$.

In the two-stage decision model, the objective is to determine a value $Y^* \in \mathbb{R}_+$ such that

$$EW^*(Y^*,\omega) = \min_{Y \in \mathbb{R}_+} \{EW^*(Y,\omega)\}.$$

In the distribution model, the objective is to determine a function $Y^\circ : \Omega \to \mathbb{R}_+$ such that

$$W^*(Y^\circ(\omega),\omega) = \min_{Y \in \mathbb{R}_+} \{W^*(Y,\omega)\}, \forall \omega \in \Omega.$$

This problem is the symmetric counterpart of the two-stage scheduling problem from the previous section. One can view items as jobs, weights as processing times, bins as machines and their capacity as a job completion deadline, but now the order of the decisions is reversed. In fact, the above cost structure is quite natural in this context. First, a delivery date for the jobs is negotiated, whereby the cost of extending this date by one unit is independent of the number of machines that will turn out to be needed later on.

In analogy to the two-stage scheduling heuristic given at the end of Section 2.1, one can consider the following two-stage bin packing heuristic. The bin capacity is set equal to the value of Y that minimizes the *lower bound* $W^{LB}(Y) = dY + n\mu / Y$ on $EW^*(Y,\omega)$, and the items are packed into bins by the *first fit decreasing* rule, i.e., the items are taken in order of nonincreasing weights and each next item is assigned to the first bin that has enough capacity to accommodate it. This heuristic can be shown to have several strong properties of asymptotic optimality (STOUGIE (1985)).

3.2 Dynamic programming

The second stage bin packing problem of determining $X^*(Y,\omega)$ for given X and ω is NP-hard (GAREY and JOHNSON (1979)). We will again consider the situation in which the stochastic parameters can assume only k values $a_1,...,a_k$, for a fixed k, and write $\omega = [n_1,...,n_k]$ to denote the vector in which the value a_j occurs n_j times,

for $j = 1,...,k$.

The following dynamic programming algorithm is due to (HELD, KARP and SHARESHIAN (1963)). Let $C(Y,\omega)$ be the total amount of capacity needed to pack items with weights specified by ω into bins of capacity Y. It is assumed that $C(Y,\omega)$ includes the slack capacity of each bin (which is equal to Y minus the total weight of the items assigned to that bin) except for the slack capacity of the last bin. Thus, if $C(Y,\omega) = XY - \Gamma$ with $X \in \mathbb{Z}_+$ and $0 \leqslant \Gamma < Y$, then an optimal packing requires X bins and the last bin has a slack capacity Γ. Let $\Delta(Y,\omega,a)$ be the extra capacity needed when an item with weight a is added to this packing:

$$\Delta(Y,\omega,a) = \begin{cases} a & \text{if } \Gamma \geqslant a, \\ \Gamma + a & \text{if } \Gamma < a. \end{cases}$$

It is not hard to see that

$$C(Y,[n_1,...,n_k]) = \min_{1 \leqslant j \leqslant k : n_j > 0} \{C(Y,[n_1,...,n_{j-1},n_j - 1,n_{j+1},...,n_k]) + \Delta(Y,[n_1,...,n_{j-1},n_j - 1,n_{j+1},...,n_k],a_j)\}$$
$$(n_1 + ... + n_k > 0),$$

$$C(Y,[0,...,0]) = 0.$$

We finally have that $X^*(Y,\omega) = \lceil C(Y,\omega) / Y \rceil$.

For the two-stage bin packing problem, we make the same assumptions concerning the distribution of the stochastic parameters as in Section 2.2 and apply the same strategy to obtain solutions to both stochastic optimization models. Since the values $a_1,...,a_k$ are integral, there is no loss of generality in considering only integral capacities Y. Let $a_{\max} = \max\{a_1,...,a_k\}$ and note that $1 \leqslant Y \leqslant na_{\max}$. The algorithm requires a fixed number of comparisons for each of the $O(n^{k+1}a_{\max})$ pairs (Y,ω), and hence $O(n^{k+1}a_{\max})$ time altogether.

A more efficient implementation of the algorithm is obtained as follows. Let $a_{\text{sum}} = \sum_{j=1}^{k} n_j a_j$. It is not hard to see that, for any Y and $\omega = [n_1,...,n_k]$

$$\lceil a_{\text{sum}} / Y \rceil \leqslant X^*(Y,[n_1,...,n_k]) \leqslant 2 \lceil a_{\text{sum}} / Y \rceil - 1.$$

The lower bound is trivial. The upper bound is a performance guarantee of the following simple heuristic: deal with the items in a fixed order and fill each of $\lceil a_{\text{sum}} / Y \rceil$ bins successively, thereby splitting an item if necessary; next, reassign each of the split items to a separate bin, of which no more than $\lceil a_{\text{sum}} / Y \rceil - 1$ will be needed. Addition of the first stage cost yields

$$dY + a_{\text{sum}} / Y \leqslant W^*(Y,[n_1,...,n_k]) \leqslant dY + 2a_{\text{sum}} / Y + 1.$$

These lower and upper bound functions are both convex and unimodal in Y. The function $W^*(Y,\omega)$ therefore attains its minimum for a value of Y that is bounded by the two values of the argument for which the lower bound is equal to the minimum of the upper bound. A straightforward calculation shows that the latter

values are given by $(\frac{1}{2}+(2a_{\text{sum}}d)^{\frac{1}{2}} \pm (a_{\text{sum}}d+(2a_{\text{sum}}d)^{\frac{1}{2}}+\frac{1}{4})^{\frac{1}{2}})/d$. This implies that for all n^k realizations ω only $O((na_{\max}/d)^{\frac{1}{2}})$ values of Y have to be considered. The overall running time is thereby reduced to $O(n^{k+\frac{1}{2}}a_{\max}^{\frac{1}{2}}d^{-\frac{1}{2}})$.

Due to the relation between the two-stage scheduling and bin packing problems that was observed above, the $Y^*(X,\omega)$ values from Section 2.2 could be used to derive the $X^*(Y,\omega)$ values needed here and vice versa, as long as the set $\{a_1,...,a_k\}$ is the same in both cases. The former recursion has the advantage of requiring strictly polynomial time; the latter one is pseudopolynomial but much faster for small values $a_1,...,a_k$.

3.3 Computational results

For the typical problem instance given by

$$d = 1,$$

$$n = 100,$$

$$k = 2, a_1 = 18, a_2 = 14, Pr\{\omega_j = a_1\} = Pr\{\omega_j = a_2\} = \frac{1}{2}\ (j = 1,...,n),$$

Figure 3 shows three functions of the first stage decision variable, the capacity Y:
- the lower bound $W^{LB}(Y)$;
- the expected total cost $EW^*(Y,\omega)$ in case of an optimal second stage decision;
- the expected total cost in case of an approximate second stage decision.

An investigation of these and other results leads to the same conclusions concerning running time, quality of lower and upper bonds, and the occurrence of multiple local minima as in Section 2.3.

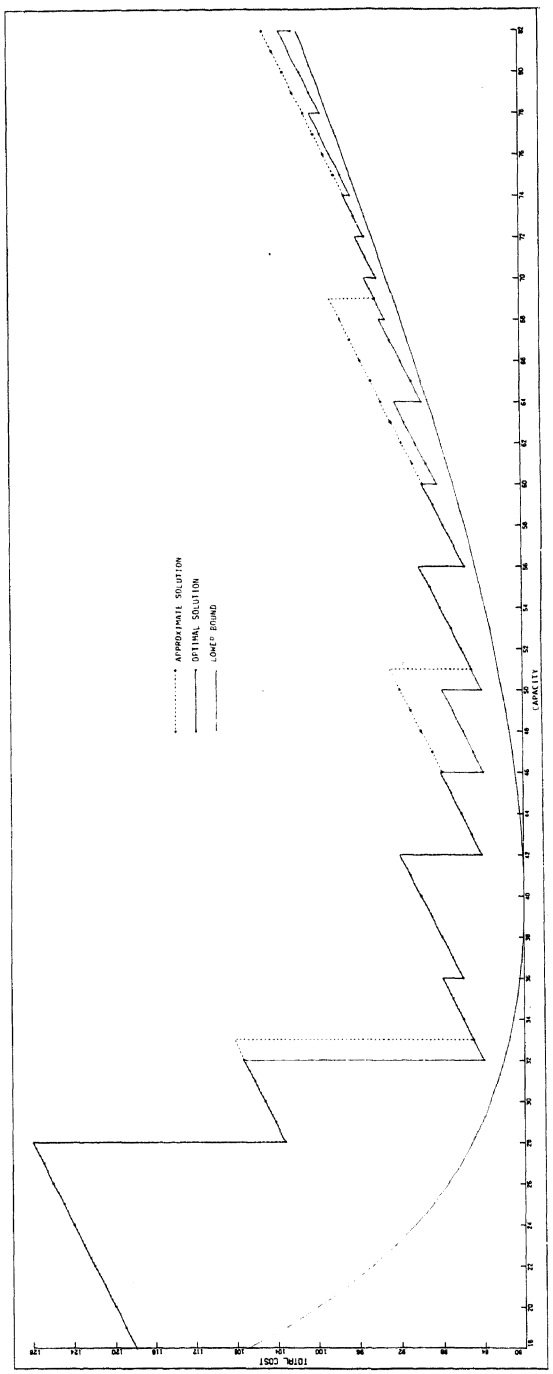

Figure 3. Bin packing: the total cost as a function of the bin capacity.

4 Multiknapsack

4.1 Problem formulation

The two-stage multiknapsack problem that we will consider here can be viewed as a *capital budgeting* problem. At the aggregate level, one has to decide on the sizes $X_1,...,X_m$ of m *budgets* that are to be reserved for financing a number of *projects*, while knowing the *cost* c_i of reserving one unit of budget i ($i = 1,...,m$), the *requirement* r_{ij} of project j out of budget i ($i = 1,...,m$, $j = 1,...,n$), and the probability distribution of the vector $\omega = (\omega_1,...,\omega_n)$ of *revenues* that the projects will yield. It is assumed that all c_i, r_{ij} and ω_j are nonnegative and that the r_{ij} are integral. At the detailed level, after $X = (X_1,...,X_m)$ has been determined, a realization $\omega \in \Omega$ of ω becomes known, and one has to decide on a *selection* S of the projects that maximizes the total revenue $Y^*(X,\omega)$ within the budget constraints:

$$Y^*(X,\omega) = \max_{S \subseteq \{1,...,n\}} \{\sum_{j \in S} \omega_j \mid \sum_{j \in S} r_{ij} \leqslant X_i \ (i = 1,...,m)\}.$$

The total profit of the budgeting decision X and the optimal selection decision is denoted by $Z^*(X,\omega) = -\sum_{i=1}^{m} c_i X_i + Y^*(X,\omega)$.

In the two-stage decision model, the objective is to determine a vector $X^* \in \mathbb{R}_+^m$ such that

$$EZ^*(X^*,\omega) = \max_{X \in \mathbb{R}_+^m} \{EX^*(X,\omega)\}.$$

In the distribution model, the objective is to determine a function $X^\circ : \Omega \to \mathbb{R}_+^m$ such that

$$Z^*(X^\circ(\omega),\omega) = \max_{X \in \mathbb{R}_+^m} \{Z^*(X,\omega)\}, \forall \omega \in \Omega.$$

4.2 The distribution model

The knapsack problem, i.e., the second stage problem with $m = 1$, is already NP-hard (GAREY and JOHNSON (1979)). Surprisingly, the distribution model is easily solved to optimality. For each $\omega \in \Omega$, the selection $S(\omega)$ of profitable projects is given by $S(\omega) = \{j \mid \omega_j - \Sigma_{i=1}^m c_i r_{ij} > 0\}$. The minimum budgets needed to finance these projects are equal to $X_i^\circ(\omega) = \Sigma_{j \in S(\omega)} r_{ij}$ ($i = 1,...,m$), and the corresponding total profit is

$$Z^*(X^\circ(\omega),\omega) = \sum_{j \in S(\omega)} (\omega_j - \sum_{i=1}^m c_i r_{ij}), \forall \omega \in \Omega.$$

In the situation that each revenue ω_j can assume only k distinct values, the determination of X° requires $O(mn)$ computations for each of k^n realizations ω.

4.3 Dynamic programming

The second stage multiknapsack problem is solvable by a classical dynamic programming algorithm from (BELLMAN (1957)). Let $F_j(X,\omega)$ be the maximum revenue

if only the first j projects can be selected, for given budgets $X = (X_1,...,X_m)$ and revenues $\omega = (\omega_1,...,\omega_n)$. An optimal selection is either restricted to the first $j-1$ projects or includes project j:

$$F_j((X_1,...,X_m),\omega) = \max\{F_{j-1}((X_1,...,X_m),\omega),$$
$$F_{j-1}((X_1-r_{1j},...,X_m-r_{mj}),\omega) + \omega_j\} \quad (j = 1,...,n),$$
$$F_0((X_1,...,X_m),\omega) = \begin{cases} 0 & \text{if } X_1 = \cdots = X_m = 0, \\ -\infty & \text{otherwise.} \end{cases}$$

Since the requirements r_{ij} are integral, also the budgets X_i can be assumed to be integral. Computation of $Y^*(X,\omega) = F_n(X,\omega)$ requires a single comparison for each of $\Pi_{i=1}^m X_i$ vectors $X' \leqslant X$ at each of n successive stages, and hence $O(n\Pi_{i=1}^m X_i$ time altogether.

For the two-stage multiknapsack problem, we again consider the situation in which each revenue ω_j can assume only k distinct values, for a fixed k. Let $R_i = \Sigma_{j=1}^n r_{ij}$ and note that $0 \leqslant X_i \leqslant R_i$ $(i = 1,...,m)$. At stage j, only the k^j different realizations of $(\omega_1,...,\omega_j)$ need to be distinguished $(j = 1,...,n)$. The algorithm therefore has to consider $O(k^j\Pi_{i=1}^m R_i)$ pairs (X,ω) at stage j. Summation over all j yields an $O(k^n\Pi_{i=1}^m R_i)$ time bound for the computation of all $Y^*(X,\omega)$ and also for the determination of a budget vector X^* that is optimal in expectation.

4.4 Computational results

The dynamic programming algorithm was coded in PASCAL and run on a CD Cyber 170-750 to solve several instances of the two-stage knapsack problem. We set $m = 1$ at the outset and did not attempt to solve proper multiknapsack problems, for which $m \geqslant 2$. We assumed independence of the revenues ω_j and tried to make the second stage knapsack problem nontrivial by specifying a high correlation between the expected revenue $E\omega_j$ of project j and its budget requirement r_{1j}. The solution of instances with twelve projects and two possible revenue values for each of them required about ten seconds.

For the problem instance given by

$$m = 1,\ c = 1,$$
$$n = 12,\ Pr\{\omega_j = a_{1j}\} = Pr\{\omega_j = a_{2j}\} = \tfrac{1}{2}\ (j = 1,...,n),$$

with the values of r_{ij}, a_{1j}, a_{2j} $(j = 1,...,n)$ given in Table 1, Figure 4 shows the expected total profit $EZ^*((X_1),\omega)$ as a function of the budget size X_1. Note that the profit is shown only for integral X_1; the line segments that start from the points shown with a slope $-c_1$ and that indicate the profit for fractional X_1 have been deleted. Even if we restrict our attention to integral values of X_1, the profit function has many local maxima.

Table 1. *Knapsack: numerical data.*

j	1	2	3	4	5	6	7	8	9	10	11	12
r_{1j}	5	2	9	13	10	8	4	7	10	6	4	9
a_{1j}	7	4	12	17	15	12	5	9	14	9	6	11
a_{2j}	3	1	6	11	8	7	1	4	7	7	2	8

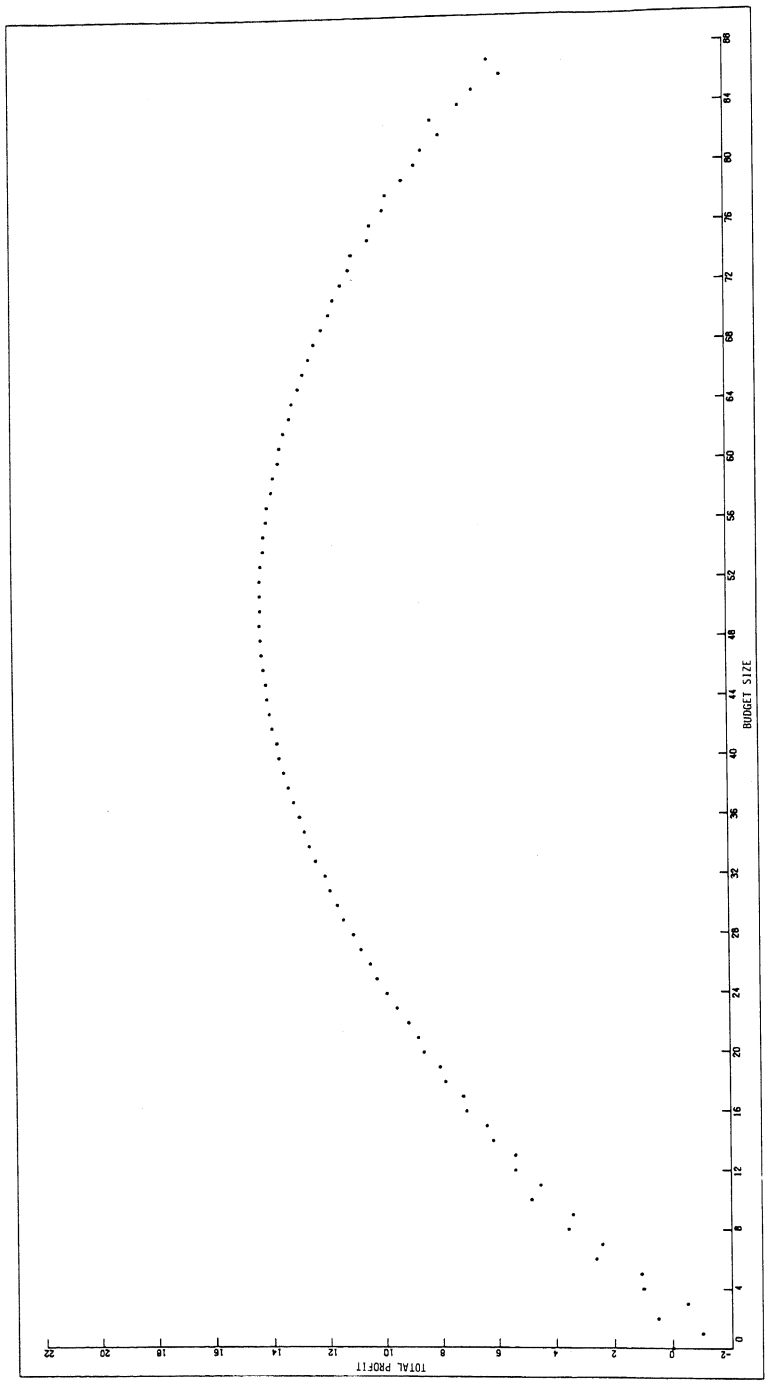

Figure 4. Knapsack: the total profit as a function of the budget size.

References

BELLMAN, R.E. (1957), *Dynamic Programming,* Princeton University Press, Princeton, NJ.

BLAIR, C.E. and R.C. JEROSLOW (1982), The value function of an integer program, *Math. Programming 23,* 237-273.

DEMPSTER, M.A.H. , M.L. FISHER, L. JANSEN, B.J. LAGEWEG, J.K. LENSTRA and A.H.G. RINNOOY KAN (1981), Analytical evaluation of hierarchical planning systems, *Oper. Res. 29,* 707-716.

DEMPSTER, M.A.H., M.L. FISHER, L. JANSEN, B.J. LAGEWEG, J.K. LENSTRA and A.H.G. RINNOOY KAN (1983), Analysis of heuristics for stochastic programming: results for hierarchical scheduling problems, *Math. Oper. Res. 8,* 525-537.

FRENK, J.B.G., A.H.G. RINNOOY KAN and L. STOUGIE (1984), A hierarchical scheduling problem with a well-solvable second stage, *Ann. Oper. Res. 1,* 43-58.

GAREY, M.R. and D.S. JOHNSON (1979), *Computers and Intractability: a Guide to the Theory of NP-Completeness,* Freeman, San Francisco.

GRAHAM, R.L., E.L. LAWLER, J.K. LENSTRA and A.H.G. RINNOOY KAN (1979), Optimization and approximation in deterministic sequencing and scheduling: a survey, *Ann. Discrete Math. 5,* 287-326.

HELD, M., R.M. KARP and R. SHARESHIAN (1963), Assembly-line balancing - dynamic programming with precedence constraints, *Oper. Res. 11,* 442-459.

LENSTRA, J.K., A.H.G. RINNOOY KAN and L. STOUGIE (1984), A framework for the probabilistic analysis of hierarchical planning systems, *Ann. Oper. Res. 1,* 23-42.

MARCHETTI-SPACCAMELA, A., A.H.G. RINNOOY KAN and L. STOUGIE (1984), Hierarchical vehicle routing problems, *Networks 14,* 571-586.

STOUGIE, L. (1985), *Design and Analysis of Algorithms for Stochastic Integer Programming,* Ph. D. thesis, Centre for Mathematics and Computer Science, Amsterdam.

WETS, R.J.-B. (1983), Stochastic programming: solution techniques and approximation schemes, in BACHEM A. , M. GRÖTSCHEL, B. KORTE (eds.), *Mathematical Programming: the State of the Art - Bonn 1982,* Springer, Berlin, 566-603.

B.J. Lageweg
J.K. Lenstra
L. Stougie
Centre for Mathematics
and Computer Science
P.O. Box 4079
1009 AB Amsterdam
The Netherlands

A.H.G. Rinnooy Kan
Econometric Institute
Erasmus University Rotterdam
P.O. Box 1738
3000 DR Rotterdam
The Netherlands