Planning and Control of Transportation Systems: The Stochastic Knapsack Problem with Random Weights *

Professor Cynthia Barnhart

Massachusetts Institute of Technology
Cambridge, Massachusetts
77 Massachusetts Avenue
Cambridge, MA 02139

A Final Report to:
The New England (Region One) UTC
Massachusetts Institute of Technology
77 Massachusetts Avenue, Room 1-235
Cambridge, Massachusetts

May 10, 2000

* Supported by a grant from the US Department of Transportation, University Transportation Centers Program

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Planning and Control of Transportation Systems: The Stochastic Knapsack Problem with Random Weights		5. Report Date **May 10, 2000** 6. Performing Organization Code
7. Author(s) **Prof. Cynthia Barnhart**		8. Performing Organization Report No.
9. Performing Organization Name and Address **Massachusetts Institute of Technology** **77 Massachusetts Avenue** **Cambridge, MA 02139**		10. Work Unit No. (TRAIS) 11. Contract or Grant No. **DTRS95-G-0001**
12. Sponsoring Agency Name and Address **New England (Region One) UTC** **Massachusetts Institute of Technology** **77 Massachusetts Avenue, Room 1-235** **Cambridge, MA 02139**		13. Type of Report and Period Covered **Final Report 9/1/97-9/30/99** 14. Sponsoring Agency Code

15. Supplementary Notes
Supported by a grant from the US Department of Transportation, University Transportation Centers Program

16. Abstract

When transportation plans are developed, problem parameters are often not known with certainty. In addition, the parameters may vary throughout the time period during which the plan is implemented. Thus, improvements can often be made in the quality of such plans by accounting for this variability. Incorporating stochasticity into problems that are already large-scale and difficult to solve, however, is a significant challenge. Heuristic methods are often helpful, as is de-coupling a large problem into a series of smaller problems for which solution techniques may already exist. To contribute to the building of a toolkit for tackling such complex planning problems in a robust manner, we have chosen to focus on solving the stochastic knapsack problem with random weights (*SKPRW*).

The deterministic knapsack problem is a classical problem with a wide range of transportation applications and a substantial body of literature. In this problem, there are a collection of objects, each with a given weight and value. The objective is to choose the set of objects with maximum collective value without exceeding an upper bound on their combined weight. We extend this deterministic knapsack problem literature by allowing the weights of each object to be random.

We can view the SKPRW as a resource allocation problem. A planner is choosing amongst a series of business opportunities (for example, possible shipping customers). Each customer requires some quantity of a resource such as time, labor, physical materials, or space on a transportation link. The planner has a known, fixed, and finite supply of this resource. The problem is to select the set of customers that best utilize this capacity. This is further complicated by the fact that each customer's demand for the limited resource is not known fully at the time of the decision. For example, a freight transportation company may be choosing to commit to a customer without knowing exactly how much freight that customer will need to ship on any given day. Thus we can view this problem as a knapsack problem, where each object (here, the business opportunities) has a weight and a value and the knapsack has a fixed capacity. We are seeking to select the optimal set of objects, given the complicating factor that the weights are random variables.

In this report we provide a number of motivations and formulations for the SKPRW. We also establish several properties useful in solving it. We then suggest algorithmic approaches to exploit these properties, in some instances providing polynomial solution techniques. Finally, we conclude with a case study yielding basic empirical insights.

17. Key Words	18. Distribution Statement		
stochastic optimization, robust planning, stochastic knapsack problem with random weights			
19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages **12**	22. Price **$60,000.00**

Form DOT F 1700.7 Reproduction of form and completed page is authorized

The Stochastic Knapsack Problem with Random Weights: A Heuristic Approach to Robust Transportation Planning

Amy Mainville Cohn
Cynthia Barnhart, MIT
Massachusetts Institute of Technology
Cambridge, MA 02139

ABSTRACT

When transportation plans are developed, problem parameters are often not known with certainty. In addition, the parameters may vary throughout the time period during which the plan is implemented. Thus, improvements can often be made in the quality of such plans by accounting for this variability. However, incorporating stochasticity into problems that are already large-scale and difficult to solve is a significant challenge. Heuristic methods are often helpful, as is de-coupling a large problem into a series of smaller problems for which solution techniques may already exist. To contribute to the building of a toolkit for tackling such complex planning problems in a robust manner, we have chosen to focus on solving the stochastic knapsack problem with random weights (*SKPRW*).

The deterministic knapsack problem is a classical problem with a wide range of transportation applications and a substantial body of literature. In this problem, there are a collection of objects, each with a given weight and value. The objective is to choose the set of objects with maximum collective value without exceeding an upper bound on their combined weight. We extend this deterministic knapsack problem literature by allowing the weights of each object to be random.

We can view the SKPRW as a resource allocation problem. A planner is choosing amongst a series of business opportunities (for example, possible shipping customers). Each customer requires some quantity of a resource such as time, labor, physical materials, or space on a transportation link. The planner has a known, fixed, and finite supply of this resource. The problem is to select the set of customers that best utilize this capacity. This is further complicated by the fact that each customer's demand for the limited resource is not known fully at the time of the decision. For example, a freight transportation company may be choosing to commit to a customer without knowing exactly how much freight that customer will need to ship on any given day. Thus we can view this problem as a knapsack problem, where each object (here, the business opportunities) has a weight and a value and the knapsack has a fixed capacity. We are seeking to select the optimal set of objects, given the complicating factor that the weights are random variables.

In this paper we provide a number of motivations and formulations for the SKPRW. We also establish several properties useful in solving it. We then suggest algorithmic approaches to exploit these properties, in some instances providing polynomial solution techniques. Finally, we conclude with a case study yielding basic empirical insights.

INTRODUCTION

Our study of the stochastic knapsack problem was initially motivated by a search for robust planning heuristics for a variant of the vehicle routing problem. In our routing problem, a home heating fuel delivery company selects a subset of its regular customers for delivery each day, then assigns customers to drivers and designs driver routes. The customers' demands are not known exactly but instead are based on limited forecasts.

There is a significant body of literature on the vehicle routing problem with stochastic demands (VRPSD). In particular, Bertsimas *et al.* (1995), Bertsimas and Simchi-Levi (1986), Bertsimas and Van Ryzin (1993), Gendreau *et al.* (1996 [1]), Gendreau *et al.* (1996 [2]), Bertsimas (1992), Dror *et al.* (1989), and Gendreau *et al.* (1995) provide a number of algorithmic approaches to these problems. In most of this VRPSD literature, it is assumed that all customers must be serviced. If capacity is exceeded, the driver returns to the depot to gain additional capacity in order to service any remaining customers. The problem therefore includes the decision of when to return to the depot and how to order the customers. Our problem was slightly different in that there is the option of selecting only a subset of the customers to service. Those selected customers not serviced by the initial capacity are neglected altogether (and usually added to the list of potential customers for the next day's schedule) and there is usually a penalty assigned for failing to service them. Thus the selection process is an integral part of solving the problem.

Currently, most fuel companies conduct this selection process using the average demand of each customer. In some cases, this results in schedules that fail to service selected customers, often at the cost of lost revenue and, more importantly, customer good will. In other cases, the schedule does not utilize the day's capacity fully, resulting in inefficiencies.

We began to investigate the selection portion of the problem in order to produce heuristics for more robust planning. In order to gain insights into the impact of different levels of demand uncertainty on the problem, we considered a vastly simplified version. Specifically, we assumed that the network of customers in the selection is very dense (perhaps a highly populated urban area). Therefore, it might be reasonable to de-couple the selection and routing questions. Our hope was to evaluate the impact of incorporating demand uncertainty in the selection problem and then again when routing these selected customers. We could then gain some insights into the possibility of improving upon the optimal solution for the deterministic estimates by using heuristics to incorporate demand variability.

We viewed the selection problem as a variation of the classical knapsack problem. Each customer has some demand (which is unknown to us) and some value, which is a function of demand. We also have a fixed capacity and wish to choose those customers who best utilize our capacity. We therefore sought to formulate the selection problem as a stochastic knapsack problem with random weights and solve this problem using the best techniques available in the literature.

In searching the literature, we found that the study of stochastic knapsack problems is limited almost exclusively to two cases -- in which the value of the objects is random, and in which the objects themselves arrive as part of a stochastic process. For a more detailed coverage of these problems, see Caraway *et al* (1993), Steinberg and Parks (1979), Henig (1990), Morita *et al* (1989), Papastavrou *et al* (1996), Tamaki (1986), Righter (1989), Ross and Tang (1989), and Kleywegt and Papastavrou (1998). Given the importance of the knapsack problem and its prominence in a wide variety of application areas, we felt that contribution could be made in extending the literature to consider those cases where the stochastic component of the problem was the object weights. Therefore, this paper moves away from the vehicle routing problem and focuses exclusively on an introductory study of the stochastic knapsack problem with random weights (SKPRW).

PROBLEM FORMULATION

We formulate the SKPRW in the following manner. We assume that the capacity of the knapsack is known a priori. The value of each object is a linear function of its weight, where the scale factor is also known. The weights of the objects are independent random variables which are not necessarily identically distributed.

As in the standard formulation of the knapsack problem, we represent each object i by a binary variable x_i which is set to one if the object is chosen and zero otherwise. However, the standard constraint that the weight of the chosen objects not exceed the knapsack capacity is no longer valid, since the weights do not become known until after all decisions have been made. In fact, it is no longer clear what the objective function of the problem should be. We propose a formulation based on an approach similar to second-stage recourse problems.

In second-stage recourse problems, we assume that first-stage decisions are made, then the random variables are realized, and then some second-stage recourse action is taken to account for any violated constraints. In this problem, we view events as unfolding in the following way: A decision maker chooses a collection of objects to include in the knapsack. (These objects might represent fuel customers, building contracts, or any other service commitment that utilizes some portion of a fixed resource of time, labor, or material). The weight of the objects (i.e. the amount of resources required) then becomes known. If the total weight is less than the knapsack capacity, then a reward is accrued based on the weights of each object. If the total weight exceeds capacity, we still accrue reward in the same manner. However, an additional per-unit penalty is charged for any weight exceeding the capacity. This penalty might be the cost associated with failing to complete a project as promised, or the cost of purchasing additional outside resources at a higher cost.

For example, a moving company might commit to a number of households for a particular time period, not knowing the exact size of each family's move. If the total capacity of the moves exceeds the company's capabilities, they must either sub-contract the excess work, or pay some sort of penalty and loss of good will for failing to meet their commitments.

Thus, our problem is as follows: *maximize the expected value of the chosen objects minus the expected penalty of exceeding capacity*. The only constraint is that an object must be included exactly once or not at all. Objects may not be split and there is only one of each object available. Throughout the remainder of this paper, we will consider this formulation.

Notation

We denote the capacity of the knapsack by a. The penalty factor is d per unit of weight exceeding capacity. The reward per unit weight of object i is r_i. The penalty and reward factors are assumed to be positive. The expected value of the weight of object i is μ_i. The expected weight above capacity given a set of objects S is indicated by $E[>a|S]$[1]. We indicate a set of selected objects by the notation $\{x_i\}$, where the x_i's are the decision variables. Using this notation, the problem formulation is simply

$$\begin{aligned} &\max \sum r_i \mu_i x_i - d * E[> a \mid \{x_i\}] \\ &s.t. \\ &x_i \in \{0,1\}. \end{aligned} \tag{1}$$

We further assume that each object i has an independent and Normally distributed weight with mean $\mu > 0$ and standard deviation σ (see Cohn (1998) for a discussion of the case where object weights are derived from a discrete distribution).

[1] $E[> a \mid S] \equiv E\left[\max\left(0, \left(\sum_{i \in S} \mu_i\right) - a\right)\right]$

Consider a collection of objects $\{S\}$. We note that the distribution of the sum of the weights in this set is Normal (μ, σ), where μ is the sum of the object means and σ is the square root of the sum of the object variances[2]. In Cohn (1998) it is shown that the objective value of this set is

$$\left(\sum_{i \in S} r_i \mu_i\right) - d * \left[\sigma * f\left(\frac{a-\mu}{\sigma}\right) + (\mu - a) * G\left(\frac{a-\mu}{\sigma}\right)\right] \tag{2}$$

where $f(\cdot)$ is the density function of the Standard Normal distribution, and $G(\cdot)$ is the inverse cumulative density (often denoted by $1-\Phi(\cdot)$).

Our goal is to use this closed-form objective function to determine efficient optimization techniques for the SKPRW. We begin by considering the case where all objects have the same distribution but possibly distinct reward factors. We then continue to assume that all objects have the same mean but allow distinct standard deviations. Finally, we consider the case where reward factor, mean, and standard deviation all vary by object.

PROBLEM CLASSES

Identically Distributed Weight

We begin by assuming that all objects have the same mean μ, standard deviation σ, and reward factor $r > 0$. The capacity is a and the per-unit penalty is $d > 0$. There are m objects from which to choose. This problem reduces to determining the number of objects $n \leq m$ to be included in the optimal solution.

The objective value of including k objects, $f(k)$, is

$$(k * r * \mu) - d * \left[\sqrt{k}\sigma * f\left(\frac{a-k\mu}{\sqrt{k}\sigma}\right) + (k\mu - a) * G\left(\frac{a-k\mu}{\sqrt{k}\sigma}\right)\right] \tag{3}$$

It is shown in Cohn (1998) that $f(n)$, the objective value of including n objects, is monotonically increasing over the range [0, n*] and monotonically decreasing over the range [n*, m], where n^* is the optimal number. Therefore, a linear search method can be used to find the optimal solution efficiently.

Now let us allow the reward factors of the objects to vary. Clearly, given two identically distributed objects, we always prefer the object with the higher reward factor. In fact, it is shown in Cohn (1998) that this problem can be efficiently solved to optimality with a greedy algorithm. Specifically, we rank the objects by decreasing order of reward factor, then continue to accept objects until the first object for which the objective value decreases.

Same Means, Varying Standard Deviations

We next consider the case where objects have the same mean but different standard deviations. Intuitively, it would seem that objects with lower standard deviations would be preferable if the reward factors are the same, since the reward depends only on the mean but the penalty depends on both the mean and the variance. This intuition is proven in Cohn (1998) to be correct. Therefore, if all objects have the same reward factor, then we can rank objects by increasing standard deviation and again employee a greedy algorithm.

We now allow the rewards to vary as well. Clearly, if the rewards vary but decrease in the same order in which the standard deviations increase, we can still use a greedy algorithm. However, this is usually not the case. Therefore, we next consider the case where the parameters of the Normal distributions are allowed to vary freely. Given two objects with distinct means, variances, and reward factors, it is not always possible to establish the dominance relationships which permit the use of greedy algorithms. Therefore, we consider a branching approach to solving this problem.

Varying Parameters

[2] Note that the objective value takes on similar form if the weights are dependent; however issues related to dominance that are required by our branching algorithm necessitate independence.

In the branching approach to solving the SKPRW, we generate a binary tree where nodes represent potential objects to be included and for each node the two branches represent including or rejecting that object. In its most basic implementation, this approach will consider 2^n possibilities. Therefore, strong pruning and dominance techniques to reduce the number of nodes considered are critical to the solvability of any given problem. We discuss these techniques in the following section. Then, in the case study that follows, we provide some basic empirical observations.

Dominance

Dominance relationships allow us to prune sub-trees from the problem, reducing the number of nodes that must be considered. For example, in the prior section we note that given two objects with the same mean and reward factor but different variances, the object with the smaller variance makes the greater contribution to the objective value. Therefore, if we reject the object with smaller variance, we can immediately reject the object with larger variance as well, and thus the sub-tree associated with it does not need to be evaluated.

The following conditions, proven in Cohn (1998), allow us to establish dominance relationships.

- If $\mu_1 = \mu_2$, $r_1 \geq r_2$, and $\sigma_1 \geq \sigma_2$, then object 1 dominates object 2.
- If the current objective value decreases when adding an object, the branch associated with accepting the object can be pruned
- If $\mu_1 \leq \mu_2$, $\sigma_1 \leq \sigma_2$, and $\mu_1 * r_1 \geq \mu_2 * r_2$, then object 1 dominates object 2. That is, if one of two objects has lower mean and variance, but higher expected reward, then it dominates.

These conditions, applied appropriately, can dramatically reduce computation time, as we will see in the case study.

Ranking

A critical step in the algorithm is to *rank* objects to determine the order in which they will be considered in the tree. In addition to evaluating dominant objects before evaluating their dominated objects, whenever possible we would like to evaluate objects that belong to the optimal solution early in the tree. A good ranking approach will dictate this.

In the case study, we use the following ranking approach. We first rank by number of dominated objects. For example, if object *a* dominates three other objects and object *b* only dominates two other objects, then we will rank *a* before *b*. Then, given that we want to increase the rewards significantly without significantly increasing the penalty, we choose objects in order of decreasing ratio of squared reward factor to standard deviation (i.e. r_i^2 / σ_i). We do this because we want high rewards and low variances to maximize reward per unit of penalty. Note that empirical investigation could yield other strong ranking methodologies.

Bounds

In addition to pruning through dominance, we seek to eliminate certain branches from the tree by determining that their maximum potential value is less than the current lower bound on the optimal solution value. In the next section, we suggest a depth-first method for generating a strong lower bound. We refer to this as *plunging*.

Plunging for Lower Bounds

Begin by accepting the first object in the ranked list. Continue to evaluate subsequent objects from the ranked list, at each step computing the solution value of the expanded set. If the objective value increases, continue along the "accept" branch. If the objective value decreases, prune the "accept" branch, as well as all "accept" branches associated with dominated objects, then continue to investigate the remaining objects by moving along the current object's "reject" branch.

Upper Bounds

As we move lower in the list, we may reach a point where we are unlikely to find further objects that increase the objective function. To expedite the process of evaluating these objects, we suggest three types of upper bounds on the possible contribution of an individual object based on those objects already accepted. (For a more detailed discussion of this subject, see Cohn (1998)).

- Type I Upper Bound: Given a branch on which we have rejected certain objects using either branching decisions or dominance, the maximum additional value of this branch is bounded from above by the sum of each remaining object's expected reward, since we assume penalties to be non-negative. This technique will be seen in the case study that follows.
- Type II Upper Bound: If we consider objects from an ordered perspective, we can think of the first a units of weight as contributing reward only, and any additional units of weight contributing both one unit of reward and at least one half unit of penalty, since $(a - \mu) / \sigma$ is negative and thus $G((a - \mu) / \sigma)$ is at least .5. Thus, we can bound the maximum contribution of a set of remaining candidate objects in the following manner: Rank all remaining objects by decreasing reward factor. Select objects in this order, acquiring rewards according to the

object means and reward factors, until the residual capacity has been exceeded by the sum of the means. For the remainder of the list, only select those objects for whom the reward is at least twice the penalty. For these objects, only incur revenue of the mean times the reward less one-half the per-unit penalty. Since we have "filled the capacity" with maximum revenue reward and underestimated the penalty on all remaining weight, this clearly produces an upper bound on the maximum objective value of this set of objects. If this upper bound is less than the current lower bound, we can prune the branch. This bound may be useful early in a branch for quickly eliminating poor quality branches -- for example, those for which key objects have already been rejected. We demonstrate this bounding technique in the case study.

- Type III Upper Bound: Our third branching method is particularly useful when the current objects have a collective mean that is close to the capacity. Given the current collection of objectives in a branch, we can generate an upper bound on the potential of each object not already included or rejected - call this ξ_i. This upper bound is computed by using the penalty component $G((a - \mu) / \sigma)$ of the current objects and applying it to the new object i. Note that this will underestimate the penalty, providing an upper bound on the total contribution of the object. We then sum over all such i the maximum of $(0, \xi_i)$, since we reject any object whose contribution is bounded from above by a negative value. Clearly the branch cannot improve beyond its current state by more than this sum. If this bound is less than the current lower bound, we can prune the branch, as we will show in the case study.

As the case study will demonstrate, careful use of dominance and bounding to prune branches can significantly reduce the number of nodes that we must consider.

CASE STUDY

To better understand the branching algorithm's bounding and pruning components, we have studied in detail a specific instance of the problem. Our objective was not to make general claims about the algorithm's performance. Instead, we sought to use a test problem both for generating new ideas for bounding and pruning, and also to demonstrate the basic algorithmic approach. In addition, the case study provides us with an environment in which to make some primitive observations about incorporating stochasticity explicitly.

In this section, we provide a detailed discussion of the algorithmic techniques as applied to this problem instance. Recall that the basic step of the algorithm is to consider accepting or rejecting objects in some ordered fashion through a tree structure. The dominance and pruning rules described in previous sections allow us to reduce the number of nodes which we must explicitly review. The order in which we choose to traverse the tree will also significantly impact the number of branches explicitly considered. We propose below a set of rules for examining the tree. We by no means claim that these are the most efficient rules; further empirical and theoretical study is required to make any specific claims. However, these rules have allowed us to determine the optimal solution to this instance of the problem in a reasonable manner.

The steps to the algorithm are as follows.

1. Order the objects by decreasing number of dominated objects. In cases of a tie, and particularly for those objects that dominate no other objects, rank by ratio of squared reward factor to standard deviation.

2. Generate a lower bound on the optimal objective value by plunging from the root. Recall that by plunging, we mean evaluating objects in ranked order and accepting each object unless it decreases the objective value. If it decreases the objective value, we reject it and continue to evaluate the remaining ordered objects. The accepted objects in this branch (let us call it branch one) provide us with a feasible solution X_1 and a lower bound $Z(1)$. This is the current highest lower bound, which we denote by LB. Add this branch to the (initially empty) list of pending branches.

3. Consider the current pending branch with the highest lower bound. Choose the first accepted object in ranked order that does not already have a "reject" sub-tree branching from it. Consider the sub-tree formed by rejecting this object.

4. Generate an upper bound on this branch by the methods described in earlier sections. If the upper bound is less than the current lower bound, prune the branch.

5. If the upper bound is greater than the current lower bound, plunge through this branch as decribed earlier. This yields a new feasible solution and lower bound. Add the branch to the list of pending branches. If this branch's lower bound is greater than LB, update LB and the current optimal solution.

6. If the current pending branch with the highest lower bound does not have any remaining accepted objects without a sub-tree, remove this branch from the pending list.

7. Repeat steps 3 - 6 until the pending list is empty, which indicates that we have completely evaluated the tree.

We demonstrate these steps for a sample data set in the following section.

Problem Instance

The following data set contains 15 randomly generated fuel customers. Each customer has a mean generated from a Normal distribution with parameters (225, 25). The reward factors are randomly selected and equally likely to be 1, 2, or 3. The penalty is 5. The variances are selected from a Uniform [5, 50] distribution. The capacity is 2000. The data is provided below.

Object	Mean	Reward Factor	Variance
1	212	2	47
2	203	2	21
3	246	3	42
4	223	2	21
5	230	2	15
6	233	1	10
7	235	2	11
8	222	2	33
9	210	1	36
10	299	2	42
11	256	2	25
12	250	3	19
13	194	1	24
14	207	3	22
15	182	1	14

Dominance

We begin by noting the following dominance relationships.

- Object 2 dominates object 9.
- Object 3 dominates object 10.
- Object 12 dominates objects 10 and 11.
- Object 14 dominates objects 1,8,10, and 11.

In all cases, this is because the dominating object has higher expected reward but lower mean and variance.

Ranking

Using the rules discussed earlier, we establish the following ranking:

Object	Reason
14	Dominates 1, 8, 10, and 11
12	Dominates 10 and 11
3	Dominates 10; $r^2 / \sigma = 1.39$
2	Dominates 9; $r^2 / \sigma = 0.87$
7	$r^2 / \sigma = 1.21$
5	$r^2 / \sigma = 1.03$
4	$r^2 / \sigma = 0.87$
11	$r^2 / \sigma = 0.80$
8	$r^2 / \sigma = 0.70$
10	$r^2 / \sigma = 0.62$
1	$r^2 / \sigma = 0.58$
6	$r^2 / \sigma = 0.32$
15	$r^2 / \sigma = 0.27$
13	$r^2 / \sigma = 0.20$
9	$r^2 / \sigma = 0.17$

Lower Bound

Our next step is to produce an initial lower bound for the optimal value. We do so by plunging, investigating the tree in a depth first manner (see Figure 1). Starting with object 14, we compute the objective value as we add each subsequent item. If the item increases the objective value, we

continue on the "accept" branch. If it decreases the objective value, we prune that branch, and continue on the branch associated with rejecting that object. The results are depicted in the following table.

Objects		Value
14		621
14, 12		1371
14, 12, 3	2109	
14, 12, 3, 2		2515
14, 12, 3, 2, 7		2985
14, 12, 3, 2, 7, 5		3445
14, 12, 3, 2, 7, 5, 4	3891	
14, 12, 3, 2, 7, 5, 4, 11		4403
14, 12, 3, 2, 7, 5, 4, 11, 8	4487	
14, 12, 3, 2, 7, 5, 4, 11, 8, 10	3590	reject object 10
14, 12, 3, 2, 7, 5, 4, 11, 8, 1	3851	reject object 1
14, 12, 3, 2, 7, 5, 4, 11, 8, 6	3555	reject object 6
14, 12, 3, 2, 7, 5, 4, 11, 8, 15	3759	reject object 15
14, 12, 3, 2, 7, 5, 4, 11, 8, 13	3711	reject object 13
14, 12, 3, 2, 7, 5, 4, 11, 8, 9	3647	reject object 9

Upper Bounds on Individual Objects

Note that in order to reach this solution, we evaluated all fifteen objects, computing the true objective value associated with adding each object in turn. We could also have reduced the computation time by taking advantage of Type III bounding to reject some of the objects without explicitly computing the new objective value. Specifically, note that the objective value of objects 14 through 8 is 4487 -- call this set of objects S. The distribution of the summed weights of these objects is Normal with mean 2072 and variance 209. When we add a new object to this set, the total reward increases by the expected reward of the new object, but the amount of weight beyond capacity increases by at least one half of the mean of the new object, since $G((a - \mu_S) / \sigma_S)$ is greater than one-half and $f((a - \mu_S) / \sigma_S)$ is always positive. For example, the reward contribution of object 6 is 1 * 233 = 233, but the penalty will be at least 5 * 233 * 0.5 and thus we can reject this object. In general, the penalty will be at least $2.5 * \mu_i$ for any new object i added to set S. Therefore, we can bound the contribution of object i by $(r_i - .5p) * \mu_i$, where p is 5 in this instance. Since all remaining objects have rewards less than 2.5, we know that there is no potential for further improvement.

Next Branch

We now have a lower bound of 4487 on the objective value. We next return to the root of the master tree. We consider rejecting the first object in this branch -- in this case, object 14. Note that object 14 dominates objects 1, 8, 10, and 11. Therefore, we can eliminate the branches associated with including these objects as well. Using Type I branching, which ignores all penalties and includes only rewards, we see that the value of this branch is bounded from above by 4089, which is less than our current lower bound. Therefore, we can prune this branch. Clearly, object 14 is part of the optimal solution.

Next Branch

We move to the next available node of the best (and only) pending branch and consider rejecting object 12. Through dominance we can also reject objects 11 and 10. Using Type I branching, we get an upper bound of 4828, which is greater than the current lower bound and thus worth pursuing. However, with Type II branching we get an upper bound of 4242. Since this is less than our current lower bound, we prune this branch as well.

Next Branch

The original branch is still the best pending branch. In this branch, the next ranked object is object 3. We determine that when rejecting object 3 (and therefore object 10, which it dominates), the remaining objects' potential is bounded through Type II branching by 4707. Since this is higher than our current lower bound, the branch is worth pursuing. We plunge along this branch, accepting objects 14, 12, 2, 7, 5, 4, 11, 8, and 1. The value of this branch is 4343. Since this is lower than our current best solution, we add this branch to our pending list. We then return to the original branch and continue in this manner. We proceed according to this algorithmic approach until no branches remain in the pending list.

Solution

Since there are 15 objects to be evaluated, at worst a branching approach could require the evaluation of $2^{15} = 32{,}768$ possibilities. Our algorithm considers 159 nodes before demonstrating the optimality of set {14, 12, 3, 2, 7, 5, 4, 8, 1}. The objective value of this set is 4,618.

This is by no means significant in enabling us to make any claims about the effectiveness of our algorithm. However, it does leave us optimistic that, with further evaluation of ranking, pruning, and branching choices, we may be able to solve many instances of this problem in a reasonable manner. This is a potential area for future research.

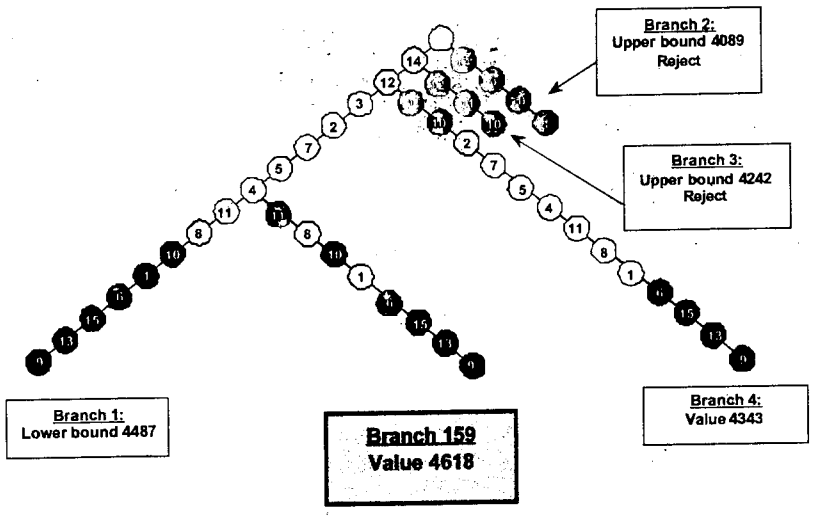

Figure 1 The branching tree

Value of Incorporating Stochasticity and Heuristic Approaches

Our study of this problem was initially motivated by the question of how the variability of customer demand impacts the selection process. Therefore, it is natural to ask how the case study's results compare to the deterministic instance.

We begin by computing the optimal solution to the deterministic knapsack problem for which the expected weights are used and the total weight is constrained to not exceed 2,000. The solution to this problem is to include objects 3, 4, 5, 7, 10, 11, 12, and 14. When we compute the expected value of this set using the objective function of our stochastic program, we determine the value of this solution to be 4,595.

For this problem instance, incorporating stochasticity explicitly has improved our results. However, we also recognize that, at least for some set of pathological instances, solving our stochastic program may require exponential time. This leads us to ask whether heuristic methods that incorporate stochasticity might out-perform optimal deterministic methods. This is a broad and difficult question which provides a rich area for research.

As a very basic initial step, we simply consider changing the capacity constraint in the deterministic case. The results are shown below for six values.

Capacity Limit	**Optimal Set**		**Stochastic Objective Value**
2,000	3, 4, 5, 7, 10, 11, 12, 14		2595
1,800	1, 2, 3, 4, 7, 8, 12, 14		4299
2,200	3, 4, 5, 7, 8, 10, 11, 12, 14	4199	
2,400	1, 3, 4, 5, 7, 8, 10, 11, 12, 14	3563	
2,050	1, 2, 3, 4, 5, 8, 11, 12, 14	4556	
2,030	3, 4, 5, 7, 10, 11, 12, 14		4595

In each case, we solved the deterministic problem with the specified capacity limit to yield an optimal set. We then computed this set's expected objective value using the same objective function as the stochastic program. We note two things in particular. First, restricting or loosening the capacity constraint does not improve the objective value in this instance. Second, replacing the capacity limit with values close to the actual mean which the optimal stochastic solution produces does not yield the same solution. This supports the idea that we must include variance in some fashion.

To try to incorporate variance more directly, we next consider the case where again the problem is solved deterministically, but the objective value of each object is altered to discourage objects with high variance. We make two changes to the original deterministic formulation. Again, we alter the capacity constraint. Since we know (in hindsight) that the optimal set's expected mean for the stochastic formulation is 2028, we

"guess" at a capacity constraint of 2030. We also replace the current objective coefficients of $r_I * \mu$ with $(r_I * \mu) + (\theta / \sigma^2)$, where θ is some positive scale factor. That is, we favor those objects with smaller variances. The following table shows the optimal set of objects for this deterministic problem, using a number of values for θ The objective value is computed using the stochastic penalty function.

Weight	Objective Set		Stochastic Objective Value
1	3, 4, 5, 8, 10, 11, 12, 14		4569
10	2, 3, 4, 5, 10, 11, 12, 14		4531
100	1, 2, 3, 4, 5, 7, 8, 12, 14		4618
1000	1, 2, 3, 4, 5, 7, 8, 12, 14		4618
10000	2, 3, 4, 5, 6, 7, 12, 14, 15	4251	

Note that when the weight θ is 100 or 1000, the results are the same as the stochastic optimal solution. While this brief analysis by no means suggests any conclusive results, it raises the following interesting question: If we can "guess" a tight upper bound on the expected mean of the optimal stochastic solution, can we find a solution close to the optimal one through a series of deterministic knapsack problems in which we incorporate the variance into the objective coefficient? Questions such as these are a useful starting point to approaching the vast and challenging area of robust planning.

CONCLUSIONS

Our study of the SKPRW has led us in two directions. We have formulated this problem in a manner consistent with many real-world applications and developed algorithms to solve this problem. We have also begun to investigate the role that stochasticity plays in solving many planning problems, gaining some basic insights and, even more importantly, many ideas for further research. We conclude by summarizing our results in these two areas.

Areas for Further Research in the SKPRW

Our algorithmic approach to solving the SKPRW relies heavily on establishing strong bounds and dominance relationships, and wisely choosing the order in which to investigate different branches of the tree. In the worst case, our approach may take exponential time. This is not surprising, given that the deterministic knapsack problem is computationally complex. However, in practice, as with the deterministic problem, we suspect that for many problem instances, we will actually need to evaluate only a small number of branches.

One important area for further research is to analyze branching rules more carefully and to generate tighter bounds in order to decrease the number of branches that the algorithm considers. Such results may be very application-specific. There may also be general results that apply to all instances of the problem.

Another important issue is to relax some of the assumptions that were made in our formulation. First, we have had initial preliminary success in a sampling approach which will grant us far greater flexibility in the choice of object weight distributions.

Another assumption that we might consider relaxing is the requirement that the penalty parameter d is constant and thus the penalty is linear in the expected weight beyond capacity. We suggest two obvious alternatives.

- Allow for non-linear penalty functions. For example, the penalty might increase exponentially as the expected weight beyond capacity increases.

- Allow for the penalty terms d_I to vary by object. Note that this changes the problem from a true knapsack structure in that we must determine not only what objects to choose, but which to penalize when capacity is exceeded. This problem might be an obvious extension to our fuel delivery problem, for example, where we need to determine which of the selected customers to skip if customer demands exceed capacity.

Finally, multiple-knapsack problems and bin-packing problems have many similarities to the single knapsack problem. These problems also have an array of uses in robust transportation planning. Perhaps some of the techniques developed here can be adapted for these problems.

Stochastic Programming for Robust Planning: Basic Insights

The original motivation behind solving the SKPRW was to recognize the difference between transportation plans and their implementations, and to try to improve the quality of this transition.

If we compare the value of the optimal solution to the SKPRW to the stochastic value of the deterministic optimal set, then the stochastic problem will always yield at least as good a solution. This is because all feasible solutions for the deterministic problem are feasible for the stochastic problem but not vice versa. This fact led us to pose the question of whether some heuristic might improve upon the deterministic result with less computational expense than the exact stochastic algorithm.

Clearly, we cannot suggest conclusive results of any type from our single case study. However, we pose a number of questions for future research.

- Are there heuristic measures for incorporating stochasticity that guarantee better results than the deterministic solution, and require less computational effort than solving the SKPRW exactly?
- Under what conditions can we de-couple a large planning problem into iterative steps, where we can solve each step by incorporating stochasticity?
- Are there other classical problems with rich deterministic literature but limited stochastic consideration which may have a wide range of uses in robust planning?

These and other similar questions promise a rich and rewarding area for extensive future research.

REFERENCES

Bertsimas, D., (1992) A Vehicle Routing Problem with Stochastic Demand. *Operations Research* **40 #3**, 574 – 585.

Bertsimas, D., Chervi, P., and Peterson, M., (1995) Computational Approaches to Stochastic Vehicle Routing Problems. *Transportation Science* **29 #4**, 342 – 352.

Bertsimas, D. and Simchi-Levi, D., (1986) A New Generation of Vehicle Routing Research: Robust Algorithms, Addressing Uncertainty. *Operations Research* **11 #4**, 619 – 626.

Bertsimas, D. and Van Ryzin, G., (1993) Stochastic and Dynamic Vehicle Routing in the Euclidean Plane with Multiple Capacitated Vehicles. *Operations Research* **41 #1**, 60 – 76.

Caraway, R., Schmidt, S., and Weatherford, L., (1993) An Algorithm for Maximizing Target Achievement in the Stochastic Knapsack Problem with Normal Returns. *Naval Research Logistics* **40**, 161 – 173.

Cohn, A.M., (1998) The Stochastic Knapsack Problem with Random Weights. Operations Research Center working paper, MIT, Cambridge, Massachusetts.

Dror, M., Laporte, G., and Trudeau, P., (1989) The Vehicle Routing Problem with Stochastic Demands: Properties and Solution Frameworks. *Transportation Science* **23 #3**, 166 – 176.

Gendreau, M., Laporte, G., and Seguin, R., (1995) An Exact Algorithm for the Vehicle Routing Problem with Stochastic Demands and Customers. *Transportation Science* **29 #2**, 143 – 155.

Gendreau, M., Laporte, G., and Seguin, R., (1996) Stochastic Vehicle Routing. *European Journal of Operations Research* **88**, 3 – 12.

Gendreau, M., Laporte, G., and Seguin, R., (1996) A Tabu Search Heuristic for the Vehicle Routing Problem with Stochastic Demands and Customers. *Operations Research* **44 #3**, 469 – 477.

Henig, M.I., (1990) Risk Criteria in a Stochastic Knapsack Problem. *Operations Research* **38 #5**, 820 – 825.

Kleywegt, A. and Papastavrou, J., (1998) The Dynamic and Stochastic Knapsack Problem. *Operations Research* **46 #1**, 17 – 35.

Morita, H., Ishii, H., and Nishida, T., (1989) Stochastic Linear Knapsack Programming Problem and Its Application to a Portfolio Selection Problem. *European Journal of Operations Research* **40**, 329 – 336.

Papastavrou, J., Rajagopalan, S., and Kleywegt, A., (1996) The Dynamic and Stochastic Knapsack Problem with Deadlines. *Management Science* **42 #12**, 1706 – 1718.

Righter, R., (1989) A Resource Allocation Problem in a Random Environment. *Operations Research* **37 #2**, 329 – 338.

Ross, K.W. and Tsang, D.H.K., (1989) The Stochastic Knapsack Problem. *IEEE Transactions on Communications* **37 #7**, 740 – 747.

Steinberg, E. and Parks, M.S., (1979) A Preference Order Dynamic Program for a Knapsack Problem with Stochastic Rewards. *Journal of the Operations Research Society* **30 #2**, 141 – 147.

Tamaki, M., (1986) A Generalized Problem of Optimal Selection and Assignment. *Operations Research* **34 #3**, 486 – 493.