European Journal of Operational Research 239 (2014) 625–635

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

An adaptive stochastic knapsack problem ☆

Kai Chen *, Sheldon M. Ross

Epstein Department of Industrial and Systems Engineering, University of Southern California, Los Angeles, CA 90089, United States

ARTICLE INFO

Article history:
Received 29 August 2013
Accepted 19 June 2014
Available online 27 June 2014

Keywords:
Decision process
Dynamic programming
Stochastic knapsack

ABSTRACT

We consider a stochastic knapsack problem in which the event of overflow results in the problem ending with zero return. We assume that there are n types of items available where each type has infinite supply. An item has an exponentially distributed random weight with a known mean depending on its type and the item's value is proportional to its weight with a given factor depending on the item's type. We have to make a decision on each stage whether to stop, or continue to put an item of a selected type in the knapsack. An item's weight is learned when placed to the knapsack. The objective of this problem is to find a policy that maximizes the expected total values. Using the framework of dynamic programming, the optimal policy is found when $n = 2$ and a heuristic policy is suggested for $n > 2$.

1. Introduction

In the classic knapsack problem, given a set of items whose values and weights are deterministic, the objective is to find a subset of items to put in the knapsack in order to maximize the total values without incurring overflow. The knapsack problem and its variants have many applications in such areas as transportation scheduling, projects selection, resource allocation/management, and others.

This paper considers the zero return if broken (ZRB) knapsack problem. In ZRB knapsack problem, an item's weight is unknown before being put in but follows a known distribution; its value per unit weight is given and determined by the item's type. It's assumed that once the knapsack is broken from breaching the capacity constraint, all the existing items in the knapsack are wiped out without any salvage value, i.e., no additional items can be inserted into the now empty knapsack and we stop there. The ZRB knapsack problem is an adaptive stochastic knapsack problem, for which a policy is defined as a schedule to put in items sequentially which adapts to the information feedback on the updated system state. The objective is to find a policy which maximizes the expected total return.

The ZRB knapsack problem can be applied in the situations where breaking knapsack triggers the wipeout effect, e.g., over-utilization of the credit line freezes account actions; medicine overdosing negates the desired function, etc. Another application is in space exploration where loading over the capacity limit of a space craft leads to total lost of all on-board cargos. The ZRB knapsack problem is also important as its optimal expected return provides a lower bound on the optimal expected return in any adaptive knapsack problem which yields the same return as in our model when stopping occurs before the knapsack reaches capacity but where the return when the knapsack's capacity is exceeded is any arbitrary nonnegative function of the sequence of types and values of the items in the knapsack.

1.1. Literature review

The 0–1 knapsack problem (see Kellerer, Pferschy, & Pisinger (2004)) is one of NP-hard problems including traveling salesman problems, integer programming, etc. Martello, Pisinger, and Toth (2000) give a comprehensive review with further discussions on techniques commonly used in solving the knapsack problem. A stochastic knapsack problem (SKP) differs from the classic model by allowing randomness in the candidate items' weights or values (or both). According to how we assign items to the knapsack, there exist two categories of SKP: static and adaptive.

In a static SKP, the only decision made is to choose a subset of items that are simultaneously put in the knapsack. Kosuch and Lisser (2010, 2011) discuss a static SKP which assumes that a cost proportional to the overflow (that is, the amount by which the sum of the weights of the items put into the knapsack exceed its capacity) is incurred whenever the knapsack's capacity is exceeded, as well as one with a constraint on the probability of exceeding the capacity. They propose methods for locating upper and lower bounds to complement the branch and bound search. Under the assumption of normal distributions on items weights, Cohn and

☆ This material is based upon work supported by the U.S. Army Research Laboratory and the U.S. Army Research Office under Grant Number W911NF-11-1-0115.

* Corresponding author. Tel.: +1 2133001118.
E-mail addresses: kaic@usc.edu (K. Chen), smross@usc.edu (S.M. Ross).

http://dx.doi.org/10.1016/j.ejor.2014.06.027
0377-2217/

Mit (1998) explore dominance rules among candidate items and illustrate a search algorithm based on this. Merzifonluoğlu, Geunes, and Romeijn (2012) extend the discussion to include a penalty cost for capacity overflow and a salvage value for unused capacity. With results from the study of a continuous relaxation of the problem, Merzifonluoglu et al. develop a customized branch and bound search for the optimal decision and a high-quality heuristic policy. Lee and Oh (1997) discuss the asymptotic property when the knapsack capacity increases, and they use the asymptotic value-to-capacity ratio to approximate the optimal solution for a large knapsack capacity.

In an adaptive SKP, the decision maker has the option to select an available item or to stop on each stage while taking into account the latest knapsack state information from the system feedback. Some adaptive SKP papers have avoided the possibility of a broken knapsack by assuming deterministic item weights; others have included penalty terms to account for any overcapacity amount; and others have imposed a chance constraint on the overflow probability. In a SKP that assumes random values but deterministic weights, Iravani, Ilhan, and Daskin (2011) discussed the adaptive target achievement problem where the objective is to maximize the probability of achieving a target total value. They give a heuristic policy with limited look-ahead capabilities and show that it performs well. Lu, Chiu, and Cox (1999) consider the project selection problem with a deadline where projects with unknown value but deterministic resource requirements arrive to the system one by one according to a stochastic process. They give a simple form of the optimal project acceptance rule, although it's usually computationally expensive to lay out the decision map. Lin, Lu, and Yao (2008) discuss a similar problem in revenue management, where offers with stochastic price and quantity information arrive at each tim e point, and the decision is whether to accept or decline the offer. They propose a class of switch-over policies and find the optimal one in the class which has asymptotic optimality as the problem size scales up. And they also apply the result in the dynamic/flexible pricing model. Van Slyke and Young (2000) study the finite horizon SKP and its applications in yield management. In SKP with random weights, Schilling (1994) gives results on the asymptotic optimal values. Ross and Tsang (1989) formulate the network admission problem in SKP and present an optimal static control by dynamic programming. Derman, Lieberman, and Ross (1978) consider a renewal decision problem that is equivalent to an adaptive SKP where the value of an item depends only on its type rather than being proportional to its weight, and where the problem continues until the knapsack is broken at which point a final return equal to the sum of the values of al l but the final item put in the knapsack is earned. Dean, Goemans, and Vondrdk (2004) study the benefit of adaptivity in the SKP with random weights where they assume the final overflowing item contributes no value. They bound the adaptivity gap, which measures the ratio of the optimal adaptive policy value to the optimal static policy value, to a factor of four; and they also devise a polynomial-time adaptive policy that approximates the optimal policy with a factor of $3+\varepsilon$ for any positive ε. Kleywegt and Papastavrou (1998, 2001) define a class of very comprehensive SKP, the dynamic stochastic knapsack problems (DSKP). It assumes items with unknown values and weights arrive to the system stochastically. The item's value and weight are revealed upon its arrival and the decision to accept or to reject has to be made. Penalty incurred by rejection, holding cost, salvage value of items, etc. are all incorporated in the DSKP. The structural results on the optimal policy for DSKP are given in the two papers.

1.2. Outline

In Section 2, the ZRB knapsack problem is defined and formulated mathematically under the dynamic programming framework. Preliminary notations are given in this part before proceed to explore the characteristics of the model structure. We show a type preference order and demonstrate the optimal stopping rule in the latter part of the section. In Section 3, we discuss the problem when $n=2$ and propose an optimal policy in this case. The general ideas behind the optimality proof are given alongside supporting propositions which lead to the core theorem in this section. We also summarize into an easy-to-implement action selection strategy from the optimal policy for $n=2$. In Section 4, we try to generalize the optimal policy for $n=2$ to a policy that's applicable for any n using the same logic. We evaluate the generalized policy and analyze its limitations. A second heuristic policy for general n is then given and tested in a numerical example. We conclude the paper with a brief introduction of our ongoing work in Section 5.

In this paper, we define the indicator function $I_A=1$ if the event A occurs, otherwise $I_A=0$. We put the proofs which involve mainly algebraic manipulations in the Appendix A.

2. Problem setting

Consider a knapsack with a deterministic capacity w. There are n different types of items available to be put to the knapsack, and each type has infinite supply of items. A type i item, $1\leqslant i\leqslant n$, has value v_iW_i, where v_i is a deterministic positive value and W_i is the item's weight where $W_i\sim Exp(w_i)$ and w_i is the mean weight. It is assumed that an item's weight is independent with the weights of other items both within the same type and between types. At each stage, we can either choose to stop and leave the system with all the existing values in the knapsack, or, we can choose to select an item of any type to put to the knapsack. An item's weight is immediately revealed after its being put to the knapsack. If the knapsack is broken, because total weights of items in the knapsack exceed the capacity, we are forced out of the system with no return at all. Otherwise, we move to the next stage. We call the model defined above as the zero return if broken (ZRB) knapsack problem. The objective of the ZRB knapsack problem is to find a policy that achieves the maximal expected return.

2.1. Dynamic programming framework

The ZRB knapsack problem can be formulated in a dynamic programming framework. Let (r,v) be the state variable of the model where r is the remaining capacity and v is the total values of items in the knapsack. Let $V(r,v)$ be the optimal expected value function at state (r,v).

Optimality Equations where $\lambda_i=\frac{1}{w_i},\ \forall i\in[1,n]$,

$$V(r,v)=\max\left\{v,\max_{i=1,\dots,n}\left\{\int_0^r\lambda_i e^{-\lambda_i t}V(r-t,v+v_i t)dt\right\}\right\}$$
$$V(r,v)=0,\text{if } r<0. \tag{1}$$

Given n types of different items, we first want to discard those types which will never be used by an optimal policy.

Proposition 1. *If $v_i<v_j,\ w_i>w_j$, then a type i item should never be used.*

Proof. The idea of the proof comes from Smith (1978). We construct a composite component which consists of N type j items, where N is a geometric distributed random variable with parameter w_j/w_i. It is easy to see this composite component has weight that is exponentially distributed with mean $\frac{w_j}{w_j/w_i}=w_i$. Since $v_i<v_j$, it is always better to replace type i item with this composite component because the composite item has higher unit weight value than type i item does, and at the same time the weight of the

composite component has exactly the same distribution as that of type i item. □

With Proposition 1, from now on we always assume without loss of generality that: **Item types are ordered where** $v_1 < v_2 < \cdots < v_n$ **and** $w_1 < w_2 < \cdots < w_n$.

A natural question for this dynamic programming model is to determine the optimal stopping time. In analogy to one-stage look-ahead rule for optimal stopping problems, the candidate stopping rule for the ZRB knapsack problem is: if it's better to stop now than insert one more item of any type and then stop, then just stop. We call this stopping rule the one-stage look-ahead rule for the ZRB knapsack problem. According to this rule, we should stop at state (r, v) if

$$v \geqslant \int_0^r \lambda_i e^{-\lambda_i t}(v + v_i t)dt, \quad \forall 1 \leqslant i \leqslant n. \tag{2}$$

It's easy to check that if (r, v) satisfies (2), then (r', v') satisfies this condition for any r', v' where $r' < r$ and $v' > v$.

Lemma 1. *It is optimal to stop in state* (r, v) *if and only if* (r, v) *satisfies the condition* (2).

Proof. Our proof is similar to the one at Ferguson (2012) (for Theorem 2 of Chapter 5). Let's first define $V^m(r, v)$ as the optimal value function at state (r, v) for the m-stage version of the problem. Now, we only have to prove the following two parts:

(1) The stopping rule specified by (2) is the optimal one for the m-stage problem.
(2) At any state (r, v), $V^m(r, v) \to V(r, v)$ as $m \to \infty$.

By observing the stopping domain is closed under one-stage look-ahead rule for any finite-stage problem, it is easy to prove part (1) by induction. For part (2), let's denote f as the optimal policy to our infinite-stage problem, and f_m is the truncated policy from policy f applied to the m-stage problem. Let $\widetilde{V}_{f_m}(r, v)$ be the total value we'll get by applying policy f_m to the m-stage problem starting from state (r, v), and let $\widetilde{V}_f(r, v)$ be the total value for the infinite-stage problem.

Since:

$$\widetilde{V}_{f_m}(r, v) \to \widetilde{V}_f(r, v) \ a.s., \quad \text{when } m \to +\infty,$$

and $|\widetilde{V}_{f_m}(r, v)| < v + \max_i\{v_i r\}$, by dominated convergence theorem, we have:

$$\lim_{m\to\infty} E[\widetilde{V}_{f_m}(r, v)] = E[\widetilde{V}_f(r, v)].$$

Therefore,

$$V(r, v) = E[\widetilde{V}_f(r, v)] = \lim_{m\to\infty} E[\widetilde{V}_{f_m}(r, v)] \leqslant \lim_{m\to\infty} V^m(r, v).$$

Combining the above inequality with the fact that: $V(r, v) \geqslant V^m(r, v)$, $\forall m$, we have:

$$V(r, v) = \lim_{m\to\infty} V^m(r, v) \qquad \square.$$

Lemma 1 implies the optimal policy when there is only one type of item available. Assuming only type i items are available whose parameters are v_i and w_i, from the condition (2), when the state point (r, v) is on the optimal stopping boundary, we must have:

$$v = \int_0^r \lambda_i e^{-\lambda_i t}(v + v_i t)dt.$$

Solving for v on the above equation, we obtain the optimal stopping boundary when only type i items are available:

$$v = \frac{v_i}{\lambda_i}(e^{\lambda_i r} - 1 - \lambda_i r). \tag{3}$$

The function $v_i(\cdot)$ of $r, r \geqslant 0$, is defined with the above solution, and we call $v_i(\cdot)$ as the critical curve of type i where $1 \leqslant i \leqslant n$.

We show in the following proposition that when the state point (r, v) is above the critical curve of type i, then the optimal policy never chooses a type i item at (r, v).

Proposition 2. *At state* (r, v), *if* $v > v_i(r)$ *where* $v_i(\cdot)$ *is the critical curve of type i, then it's never optimal to select type i item at state* (r, v).

Proof. Assuming there are two identical knapsacks, both of which are currently at state (r, v). Let policy π be any policy which selects a type i item at state (r, v). We apply policy π in the first knapsack starting from state (r, v). After policy π puts in a type i item in the first knapsack, if this move breaks the first knapsack, the second knapsack stops at state (r, v); otherwise, skipping the first move, we start to replicate in the second knapsack the rest of moves by policy π in the first knapsack, i.e., except for the first move, whatever policy π puts into the first knapsack, we put in exactly the same item in the second knapsack. The corresponding items for the two knapsacks have the same weight and the same values by coupling their random sources. Let's assume the weight of the type i item policy π puts in the first knapsack at state (r, v) is W_i, where $W_i \sim Exp(w_i)$; and assume the total weights and values of the rest of items policy π puts into the first knapsack after the first move, if it does not break the first knapsack, are $W_{\sum}$ and $V_{\sum}$ respectively, where $W_{\sum}$ and $V_{\sum}$ are dependent on each other and $W_{\sum}$ is also dependent on W_i. Now let's denote the event of $W_i \leqslant r$ as e_i; and the event of $W_i + W_{\sum} \leqslant r$ as $e_{\sum}$. For the first knapsack, the expected return from applying policy π is:

$$E\left[\left(v + v_i W_i + V_{\sum}\right) I_{e_{\sum}}\right].$$

For the second knapsack, the expected return from applying the policy developed above is at least:

$$E[v(1 - I_{e_i})] + E\left[\left(v + V_{\sum}\right) I_{e_{\sum}}\right].$$

We also have:

$$E[v(1 - I_{e_i})] + E\left[\left(v + V_{\sum}\right) I_{e_{\sum}}\right] - E\left[\left(v + v_i W_i + V_{\sum}\right) I_{e_{\sum}}\right]$$
$$= E[v(1 - I_{e_i})] - E\left[v_i W_i I_{e_{\sum}}\right] \geqslant E[v(1 - I_{e_i})] - E[v_i W_i I_{e_i}] > 0,$$

where the first inequality from $I_{e_{\sum}} \leqslant I_{e_i}, a.s.$, and the second inequality from the given assumption that $v > v_i(r)$. Therefore, the policy applied in the second knapsack by not putting in the type i item at state (r, v) is always better than the original policy π. □

3. Optimal policy for $n = 2$

In this section, a thorough analysis is given when $n = 2$.

3.1. Preliminary analysis

We first want to find the optimal value function when $n = 1$. *Now let's assume there is only one type of items available with the parameter* v_1 *and* $w_1 (= \frac{1}{\lambda_1})$.

When only one type of item is available, $V(r, v) = v$ if $v \geqslant v_1(r)$. If $v < v_1(r)$, the optimal policy is to put in items until the state

enters the stopping domain. In the latter case, see Fig. 1, after an item being inserted at state (r, v), because the unit value per weight of a type 1 item is v_1, the next state must be on the line of slope $-v_1$ which passes the original state point (r, v). For any state (r, v) outside the stopping domain, let's denote the intersection point between the curve $v_1(\cdot)$ and the line of slope $-v_1$ passing (r, v) as $(R_{11}(r, v), V_1(r, v))$, solving for which we obtain:

$$R_{11}(r, v) = \frac{\log(v\lambda_1/v_1 + \lambda_1 r + 1)}{\lambda_1}. \tag{4}$$

Starting from any state (r, v) where $v < v_1(r)$, we focus on the last item the optimal policy put in the knapsack before the state enters the stopping domain specified by $v_1(\cdot)$ or the knapsack breaks. We can think of the weight of this item as flowing continuously until the item is completely in knapsack. Since the item is the last one before the state enters the stopping domain, as its weight flows in, there must exist an intermediate state which hits on the stopping boundary at $(R_{11}(r, v), V_1(r, v))$. Due to the memoryless property of the exponential distribution, the remaining weight of the item is exponentially distributed with mean w_1. Therefore, putting in this last item is equivalent to putting in an item at state $(R_{11}(r, v), V_1(r, v))$. On the stopping boundary $v_1(\cdot)$, it is indifferent to stop or to put in one more item and then stop. So $V_1(r, v)$ is the expected total return from the optimal policy at state (r, v) when $v < v_1(r)$. Let's use $V_1(r, v)$ as the optimal value function at any state (r, v) when only type 1 items are available. From the above discussions, we have:

$$V_1(r, v) = \begin{cases} 0, & \text{if } r < 0 \\ v, & \text{if } v \geqslant v_1(r) \\ v_1(R_{11}(r, v)), & \text{otherwise.} \end{cases} \tag{5}$$

When only type 1 items are available, we denote the optimal policy implied by the optimal stopping rule as *Policy One*, which has the value function $V_1(r, v)$ at state (r, v).

3.2. Policy statement and optimality proof for $n = 2$

In the following, we assume there are two types of items available ($n = 2$): type 1 with parameter v_1 and w_1; type 2 with parameter v_2 and w_2.

When $v < v_j(r)$, $R_{ij}(r, v)$ is defined as follows: $(R_{ij}(r, v), v_j(R_{ij}(r, v)))$ is the intersection point between the curve $v_j(\cdot)$ and the line of slope $-v_i$ passing (r, v), i.e., when we continually put in type i items, the system state crosses the critical curve of type j at the point $(R_{ij}(r, v), v_j(R_{ij}(r, v)))$. In the following, for any function $g(\cdot)$, $E_i[g(T)]$ is defined as the expectation of $g(T)$ where $T \sim Exp(w_i)$.

We define *Policy Two* when $n = 2$ as follows:

Step 1: at current state (r,v), we calculate $V_1(r, v)$ and $E_2[V_1(r - T, v + v_2T)]$;
Step 2: if $V_1(r, v) \geqslant E_2[V_1(r - T, v + v_2T)]$, we follow *Policy One* at state (r, v). If *Policy One* did not call for stopping, we update the state and go to *Step 1*;
Step 3: if $V_1(r, v) < E_2[V_1(r - T, v + v_2T)]$, we put in an item of type 2, update the state and go back to *Step 1*.

In the above policy statement, $V_1(r, v)$, which has the expression in (5), is the expected total return by applying *Policy One* through out all states starting from (r, v) until stop; $E_2[V_1(r - T, v + v_2T)]$, where

$$E_2[V_1(r - T, v + v_2T)] = \int_0^r \lambda_2 e^{-\lambda_2 t} V_1(r - t, v + v_2 t)dt,$$

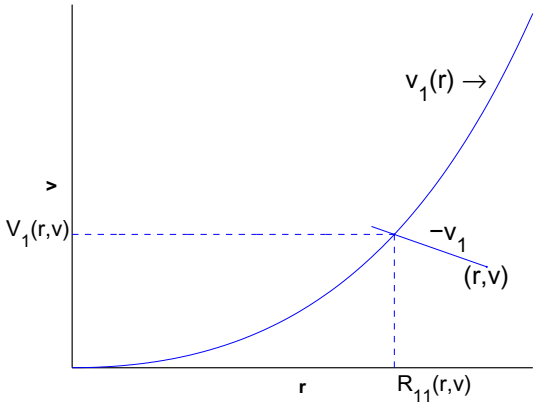

Fig. 1. Assuming only type 1 items available, at state (r, v) where $v < v_1(r)$, we will keep inserting items until the state enters the stopping domain determined by the curve $v_1(\cdot)$. We denote $(R_{11}(r, v), V_1(r, v))$ as the intersection point between $v_1(\cdot)$ and the line passing (r, v) with slope $-v_1$.

is the expected return by putting in a type 2 item at state (r, v) and then applying *Policy One* at all following states until stop.

We first show that if *Policy Two* follows *Policy One* at any state, it will do so afterward until it stops.

Lemma 2. *If* $V_1(r, v) \geqslant E_2[V_1(r - T, v + v_2T)]$, *then* $V_1(r - h, v + v_1h) \geqslant E_2[V_1(r - h - T, v + v_1h + v_2T)]$, $\forall h \leqslant r$.

Proof. See the proof in Appendix A. □

When *Policy Two* follows *Policy One* at state (r, v), if we instead put in a type 2 item at (r, v), *Policy Two* still follows *Policy One* at the updated state.

Lemma 3. *If* $V_1(r, v) \geqslant E_2[V_1(r - T, v + v_2T)]$, *then* $V_1(r - h, v + v_2h) \geqslant E_2[V_1(r - h - T, v + v_2h + v_2T)]$, $\forall h \leqslant r$.

Proof. See the proof in Appendix A. □

With Lemmas 2 and 3, we can prove when *Policy Two* follows *Policy One* at current state (r, v), its value function satisfies the optimality equations in (1).

Corollary 1. *At state* (r, v), *if* $V_1(r, v) \geqslant E_2[V_1(r - T, v + v_2T)]$, *then* $V_2(r, v)$, *the value function of Policy Two at* (r, v), *satisfies:*

$$V_2(r, v) = \max\left\{v, \max_{i=1,2}\left\{\int_0^r \lambda_i e^{-\lambda_i t} V_2(r - t, v + v_i t)dt\right\}\right\}.$$

i.e., the value function at state (r, v) *satisfies the optimality equations.*

Proof. See the proof in Appendix A. □

In the case when *Policy Two* selects a type 2 item at state (r, v), we show the value function $V_2(r, v)$ of *Policy Two* at state (r, v) still satisfies the optimality equations.

Lemma 4. *At state* (r, v) *where* $V_1(r, v) \leqslant E_2[V_1(r - T, v + v_2T)]$, *we still have:*

$$V_2(r, v) = \max\left\{v, \max_{i=1,2}\left\{\int_0^r \lambda_i e^{-\lambda_i t} V_2(r - t, v + v_i t)dt\right\}\right\}.$$

Proof. See the proof in Appendix A. □

Corollary 1 and Lemma 4 together yield the optimal policy theorem.

Theorem 1 (Optimal Policy Theorem when $\mathbf{n = 2}$). *Policy Two is the optimal policy for the ZRB knapsack when consider only two types of items available where $v_1 < v_2$, and $w_1 < w_2$.*

3.3. Optimal action theorem for $n = 2$

Policy Two is the optimal policy with two available types where $v_1 < v_2$ and $w_1 < w_2$. It turns out that to apply *Policy Two*, it is not necessary to calculate $V_1(r, v)$ and $E_2[V_1(r - T, v + v_2T)]$ at every state (r, v). We will show in the following Theorem 2, which is also called Optimal Action Theorem for $n = 2$, that the optimal decision from *Policy Two* is usually easy to find without complex calculations. To prepare the proof of Theorem 2, we need the proposition below which reveals the relation between the critical curve $v_1(\cdot)$ of type 1 and the critical curve $v_2(\cdot)$ of type 2.

Proposition 3. *Given $v_1 < v_2$, $\lambda_1 > \lambda_2$, and $v_i(\cdot)$, $i = 1, 2$, the critical curve of type i, we must have either*

$$v_1(r) > v_2(r), \quad \forall r > 0,$$

or $\exists r_0 > 0$, such that:

$$v_1(r) \begin{cases} < v_2(r), & \text{when } 0 < r < r_0, \\ = v_2(r), & \text{when } r = r_0, \\ > v_2(r), & \text{when } r > r_0. \end{cases}$$

Proof. See the proof in Appendix A. □

Proposition 3 says the critical curve $v_1(\cdot)$ is either always above the curve $v_2(\cdot)$, or there may exist a section in the beginning of the horizontal line where $v_1(\cdot)$ is below $v_2(\cdot)$ but eventually $v_1(\cdot)$ lies above $v_2(\cdot)$. This observation will be used in the proof of the following optimal action theorem.

Theorem 2 (Optimal Action Theorem for $\mathbf{n = 2}$). *Assuming only two types of items available where $v_1 < v_2$ and $w_1 < w_2$, the optimal action at state (r, v) implied by the optimal policy Policy Two is given as follows (see Fig. 2):*

Case 1 ($v \geqslant \max\{v_1(r), v_2(r)\}$): stop;
Case 2 ($v_2(r) \leqslant v \leqslant v_1(r)$): choose type 1 item;
Case 3 ($v_1(r) \leqslant v \leqslant v_2(r)$): choose type 2 item;
Case 4 ($v \leqslant \min\{v_1(r), v_2(r)\}$ and $R_{22}(r, v) \leqslant R_{21}(r, v)$): choose type 2 item;
Case 5 ($v \leqslant \min\{v_1(r), v_2(r)\}$ and $R_{22}(r, v) > R_{21}(r, v)$): there exists $r' > R_{22}(r, v)$, if $r \leqslant r'$, choose type 1 item; otherwise, choose type 2 item.

Proof. *Case 1* ($v \geqslant \max\{v_1(r), v_2(r)\}$). It is optimal to stop from the one-stage look-ahead rule.

Case 2 ($v_2(r) \leqslant v \leqslant v_1(r)$). It is optimal to put in type 1 item from Proposition 2 and the one-stage look-ahead rule.

Case 3 ($v_1(r) \leqslant v \leqslant v_2(r)$). It is optimal to put in type 2 item from Proposition 2 and the one-stage look-ahead rule.

Case 4 ($v \leqslant \min\{v_1(r), v_2(r)\}$ and $R_{22}(r, v) \leqslant R_{21}(r, v)$).

Suppose at state (r, v),

$$V_1(r, v) \geqslant E_2[V_1(r - T, v + v_2T)],$$

then by Lemma 3, at state $(r - h, v + v_2h)$ where $h \in [0, r]$, it is optimal either to put in a type 1 item or stop.

Given

$$v \leqslant \min\{v_1(r), v_2(r)\} \quad \text{and } R_{22}(r, v) \leqslant R_{21}(r, v),$$

from Proposition 3, we must have:

$$v_1(r') < v_2(r'), \quad \forall r' \in (0, R_{21}(r, v)),$$

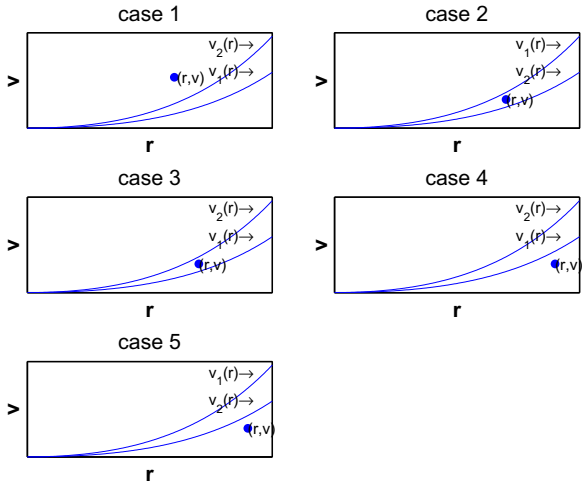

Fig. 2. Assuming only two types of items available where $v_1 < v_2$ and $w_1 < w_2$, at state (r, v), there are 5 different cases regarding the relative position of the state point (r, v) w.r.t. the critical curves $v_1(\cdot)$ and $v_2(\cdot)$.

which implies $\forall h \in (r - R_{21}(r, v),\ r - R_{22}(r, v))$,

$$v_1(r - h) < v_2(r - h),$$

and state $(r - h, v + v_2h)$ satisfies *Case 3*. Therefore, it is optimal to choose type 2 item at $(r - h, v + v_2h)$ when $h \in (r - R_{21}(r, v),\ r - R_{22}(r, v))$. So we have found a contradiction with the supposition:

$$V_1(r, v) \geqslant E_2[V_1(r - T, v + v_2T)].$$

With the given conditions, we must have

$$V_1(r, v) < E_2[V_1(r - T, v + v_2T)],$$

and the optimal action at (r, v) is to put in a type 2 item.

Case 5 ($v \leqslant \min\{v_1(r), v_2(r)\}$ and $R_{22}(r, v) > R_{21}(r, v)$). Given

$$v \leqslant \min\{v_1(r), v_2(r)\} \quad \text{and } R_{22}(r, v) > R_{21}(r, v)),$$

from Proposition 3, $v_1(\cdot)$ is above $v_2(\cdot)$ at $[R_{21}(r, v), \infty)$.

Therefore, when $r - R_{22}(r, v) \leqslant h \leqslant r - R_{21}(r, v)$, the state $(r - h, v + v_2h)$ must satisfy *Case 2*, on which it is optimal to choose type 1 item. As we've shown in the proof for Lemma 3 in the Appendix A, there must exist a positive $h' > 0$ and if we denote

$$r' = R_{22}(r, v) + h',\ v' = v_2(R_{22}(r, v)) - h'v_2,$$

the optimal policy is indifferent to put in a type 1 item or a type 2 item at state (r', v') and we must have: at (r, v), if $r \leqslant r'$, the optimal action is choose type 1 item; otherwise, choose type 2 item. □

4. Two heuristic policies for general *n*

From the previous discussions, one option to consider in the general model is to generalize the idea behind *Policy Two* to develop a policy for the model of any *n*. We'll present the generalized policy for any *n* in this section. This policy's limitation in implementation will be discussed. We also give a second heuristic policy and a numerical example is given to partially evaluate the performance of these heuristic policies.

4.1. Generalized policy n

In the following, *Policy n* is a generalized version of *Policy Two* for any *n*. As the starting point, *Policy 1* is defined with *Policy One*. In the following policy statement, for any $k < n, V_k(r, v)$ is

the value function of *Policy k*. Given *Policy k*, *Policy k* + 1 is defined as follows:

Step 1: at current state (r, v), we compare $V_k(r, v)$ with $E_{k+1}[V_k(r - T, v + v_{k+1}T)]$;
Step 2: if $V_k(r, v) \geqslant E_{k+1}[V_k(r - T, v + v_{k+1}T)]$, we use *Policy k* at (r, v) till stop; otherwise, insert one item of type $k + 1$, update the state, and go back to *Step 1*.

Remark 1. The above definition is not a direct generalization from the same idea behind *Policy Two* because we cannot prove the analog of Lemma 2 for the case of general n. However, this version is much simpler to implement compared to the generalized policy in original form which does the comparison on each stage. We will show in the following discussion that even this simpler version of generalized policy requires unacceptable amount of computational power for big n.

From the policy definition, *Policy Two* is obvious an instance of the policy when $n = 2$. We proved the optimality of *Policy Two* by showing its value function $V_2(r, v)$ always satisfies the optimality equations. In that proof, we worked on $V_1(r, v)$ which has an explicit analytical form as in Eq. (5) to reveal the characteristics of $V_2(r, v)$. However, when $n \geqslant 3$, there is no explicit analytical expression for $V_{n-1}(r, v)$ from which we can replicate the old proof to show the optimality of $V_n(r, v)$. Therefore, on the one hand, we cannot use the same technique to check whether *Policy n* is optimal or not for general n; on the other hand, if we want to implement this policy for $n \geqslant 3$, because the value function involved has no analytical form, the value function has to be simulated at each decision making point. This becomes a severe bottleneck as the problem size scales up. The running time of the implementation of *Policy n* rises exponentially as n increases.

4.2. A second heuristic policy

We introduce a second heuristic policy based on the two statements: (a) at state (r, v), only the types whose critical curves are above the state point should be considered, which are called feasible types for state (r, v); (b) if there are multiple feasible candidate types after considering (a), it's better to select the one with higher unit value per weight. The first statement has already been proved in Proposition 2. Statement (b) is not generally true since we can find explicit counter examples in Optimal Action Theorem for $n = 2$. However, when the state (r, v) is not close to those critical curves of feasible types, choosing the one with the highest unit value per weight could be a good choice. The logic behind it is the intuition that if capacity allows, we'd rather put in higher value items. We'll show in a numerical example that this heuristic policy has the potential to perform well for our problem.

The heuristic policy is developed from ideas in statements (a) and (b). At any non-stopping state (r, v), the heuristic policy selects the type which has the highest unit value per weight among all the feasible candidate types determined by statement (a). The policy is implemented as follows:

At current state (r, v), define the feasible types set $S = \{i \in [1, n] : v \leqslant v_i(r)\}$.
While $S \neq \emptyset$;
  Let $h = \max_{k \in S} k$, and put in an item of type h.
  Update the state from (r, v) to (r', v').
  If $v' \leqslant 0$, stop the program.
  Update $S = \{i \in [1, n] : v' \leqslant v_i(r')\}$.
End while.
Stop the program.

4.3. Numerical example

How to evaluate the heuristic policy when we do not know the optimal policy for the ZRB knapsack problem of $n \geqslant 3$? Our workaround is to consider the discretized version of the ZRB knapsack problem. By discretization, we assume items' weights conform to the discrete geometric distributions instead of the continuous exponential distributions, and all other problem settings are unchanged. Dynamic programming enables us to find the optimal value function at any state in the discretized model. Our implicit assumption in this part is:

If one policy works well for the discretized model, it can also do well for the original model.

We give a numerical example here on the discretized model for $n = 3$. In this example, we have three types of items available where[1]

$$v_1 = 2, p_1 = 0.8; \quad v_2 = 3, p_2 = 0.6; \quad v_3 = 4, p_3 = 0.4.$$

Note, items' preference order is still respected since:

$$v_1 < v_2 < v_3, \quad \text{and } \frac{1}{p_1} < \frac{1}{p_2} < \frac{1}{p_3}.$$

For each state $(N, 0)$, where N is a positive integer representing the knapsack capacity, we'll compute in a computer program the expected return at this state respectively by three different policies: the optimal policy implied by dynamic programming, *Policy n* for $n = 3$, and the heuristic policy. The computer program is written in C++ language. The expected return from *Policy n* and that from the heuristic policy are generated in simulation. The detail results of this numerical example are shown in the table below.

N	DP	*Policy n*	Second heuristic policy
20	65.98	63.09	65.51
40	143	138.4	141.5
60	221.1	216.9	216.2
80	299.5	292.7	297.6
100	378.6	368.9	375.6
120	457.8	451.7	457.9
140	537.2	529.4	530.3
160	616.7	607.1	618.3
180	696.2	684.7	694.2
200	775.7	762.4	768.7

From this example, we see the second heuristic policy has good performance compared to the optimal one in this discretized model. This gives us confidence that this heuristic works for the original adaptive BKP model as well. Another observation from this example is: *Policy n*, for general n, is a sub-optimal policy for the original model, although it's optimal when $n = 2$.

5. Conclusion and ongoing work

This paper considers an adaptive SKP, the ZRB knapsack problem. In this model, there are n types of items available and each type has infinite supply; an item has an exponentially distributed weight and its value is proportional to its weight with a deterministic factor. One has to select a type of item to put to the knapsack or choose to stop on each stage. The objective is to find a policy that maximizes the expected total return. In this paper, the model was formulated in the framework of dynamic programming. We proved items' preference order and showed the one-stage look-

[1] v_i is unit value per weight for type i; p_j is the parameter of the geometric distribution for the weight of type i items, and $\frac{1}{p_i}$ is the mean weight.

ahead rule is always the optimal stopping rule for the ZRB knapsack problem in our model. An in-depth analysis was given for the model when there are two types of items available. We presented and proved an optimal policy for $n = 2$. For general n, we discussed the generalized policy derived from the optimal policy of $n = 2$. A second heuristic policy was presented and the performance of the two heuristics were evaluated on a numerical example of $n = 3$.

In Chen and Ross (2013a), we extend the ZRB knapsack problem to include exponential capacity. Under certain cases on the assumption of the joint distribution function of an item's weight and value, we can either find optimal policies or transform the model to an equivalent static SKP model. In Chen and Ross (2013b), we consider two static SKP models, the static ZRB knapsack problem and the SKP with simple recourse and penalty cost. Both models try to maximize the expected total returns although with different objective functions. We show that given certain constraints on the distributions of items' random weights, the expected return functions in the two models both have a unimodality property which can be used in the search for optimal policies.

Appendix A

A.1. Proof of Lemma 2

Case 1: $v \geqslant v_1(r)$.

Under this case (check Case 1 in Fig. 3) where the point (r, v) is above the curve $v_1(\cdot)$, it's ready to see

$$V_1(r-h, v+v_1h) = v + v_1h, \quad \forall 0 \leqslant h \leqslant r.$$

Given

$$V_1(r, v) \geqslant E_2[V_1(r-T, v+v_2T)],$$

we have:

$$v \geqslant E_2[(v+v_2T)I_{T\leqslant r}],$$

which implies (r, v) is above the critical curve $v_2(\cdot)$. Therefore, for $h \in [0, r]$, the state point $(r-h, v+v_1h)$ is always in the stopping domain, which implies:

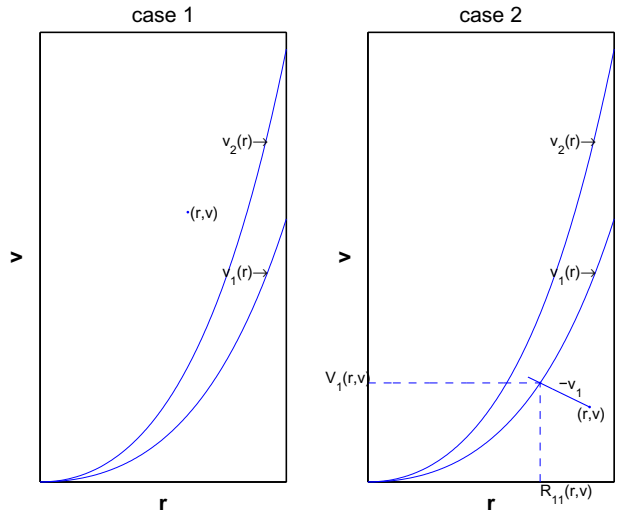

Fig. 3. Given $V_1(r, v) \geqslant E_2[V_1(r-T, v+v_2T)]$, Case 1 considers $v \geqslant v_1(r)$, from Eq. (5), we know $V_1(r-h, v+v_1h) = v + v_1h$, $\forall 0 \leqslant h \leqslant r$; Case 2 considers $v > v_1(r)$, which means the state (r, v) is outside the stopping zone determined by boundary $v_1(\cdot)$.

$$V_1(r-h, v+v_1h) \geqslant E_2[V_1(r-h-T, v+v_1h+v_2T)].$$

Case 2: $v < v_1(r)$.

In this case, we have: $r > R_{11}(r, v)$. For $h \in [0, r]$, let's define the function

$$H(h) := V_1(r-h, v+v_1h) - E_2[V_1(r-h-T, v+v_1h+v_2T)].$$

$H(h)$, where $h \in [0, r]$, is a continuous function of h with the given assumption that $H(0) \geqslant 0$. The lemma will immediately follow if we can prove $H(h)$ is an increasing function in h when $h \in [0, r]$.

Case 2.1: $0 \leqslant h \leqslant r - R_{11}(r, v)$.

The first term of $H(h)$ is constant on h because

$$V_1(r-h, v+v_1h) = v_1(R_{11}(r, v)), \quad \forall h \in [0, r - R_{11}(r, v)].$$

Let's expand the 2nd term of $H(h)$ and find its partial derivative w.r.t h. We try to prove the derivative of the expectation term is negative in order to show $H(h)$ is increasing in h when $0 \leqslant h \leqslant r - R_{11}(r, v)$.

Denote $r_h = R_{21}(r-h, v+v_1h)$, we have:

$$\begin{aligned}
&E_2[V_1(r-h-T, v+v_1h+v_2T)] \\
&= \int_0^{r-h} \lambda_2 e^{-\lambda_2 t} V_1(r-h-t, v+v_1h+v_2t)dt \\
&= \int_0^{r-h-r_h} \lambda_2 e^{-\lambda_2 t} V_1(r-h-t, v+v_1h+v_2t)dt \\
&\quad + \int_{r-h-r_h}^{r-h} \lambda_2 e^{-\lambda_2 t}(v+v_1h+v_2t)dt,
\end{aligned}$$

$$\begin{aligned}
&\frac{dE_2[V_1(r-h-T, v+v_1h+v_2T)]}{dh} \\
&= \int_0^{r-h-r_h} \lambda_2 e^{-\lambda_2 t}(V_1(r-h-t, v+v_1h+v_2t))_h dt \\
&\quad + (r-h-r_h)_h \lambda_2 e^{-\lambda_2 t} V_1(r-h-t, v+v_1h+v_2t)|_{t=r-h-r_h} \\
&\quad + \int_{r-h-r_h}^{r-h} \lambda_2 e^{-\lambda_2 t}(v+v_1h+v_2t)_h dt \\
&\quad + (r-h)_h \lambda_2 e^{-\lambda_2 t}(v+v_1h+v_2t)|_{t=r-h} \\
&\quad - (r-h-r_h)_h \lambda_2 e^{-\lambda_2 t}(v+v_1h+v_2t)|_{t=r-h-r_h} \\
&= \int_0^{r-h-r_h} \lambda_2 e^{-\lambda_2 t} 0 dt \\
&\quad + (-1-(r_h)_h)\lambda_2 e^{-\lambda_2(r-h-r_h)}(v+v_1h+(r-h-r_h)v_2) \\
&\quad + \int_{r-h-r_h}^{r-h} \lambda_2 e^{-\lambda_2 t} v_1 dt - \lambda_2 e^{-\lambda_2(r-h)}(v+v_1h+(r-h)v_2) \\
&\quad - (-1-(r_h)_h)\lambda_2 e^{-\lambda_2(r-h-r_h)}(v+v_1h+(r-h-r_h)v_2) \\
&= \int_{r-h-r_h}^{r-h} \lambda_2 e^{-\lambda_2 t} v_1 dt - \lambda_2 e^{-\lambda_2(r-h)}(v+v_1h+(r-h)v_2) \\
&= v_1 e^{-\lambda_2(r-h-r_h)} - v_1 e^{-\lambda_2(r-h)} - \lambda_2 e^{-\lambda_2(r-h)}(v+v_1h+(r-h)v_2) \\
&= e^{-\lambda_2(r-h)}\left(v_1 e^{\lambda_2 r_h} - v_1 - \lambda_2(v+v_1h+(r-h)v_2)\right).
\end{aligned}$$

Because $r_h = R_{21}(r-h, v+v_1h)$, from the definition of $R_{21}(\cdot,\cdot)$,

$$v_1(r_h) = v + v_1h + (r-h-r_h)v_2.$$

On the other hand,

$$v_1(r_h) = \frac{v_1}{\lambda_1}(e^{\lambda_1 r_h} - 1 - \lambda_1 r_h).$$

Therefore we have:

$$v + v_1h + v_2(r-h) = \frac{v_1}{\lambda_1}(e^{\lambda_1 r_h} - 1 - \lambda_1 r_h) + v_2 r_h. \tag{a.1}$$

In order to prove the partial derivative is negative, we only have to show:

$$v_1 e^{\lambda_2 r_h} - v_1 - \lambda_2(v+v_1h+(r-h)v_2) < 0.$$

By equation (a.1), we have:

$$\begin{aligned}
&v_1 e^{\lambda_2 r_h} - v_1 - \lambda_2(v + v_1 h + (r-h)v_2) \\
&= v_1 e^{\lambda_2 r_h} - v_1 - \lambda_2\left(\frac{v_1}{\lambda_1}(e^{\lambda_1 r_h} - 1 - \lambda_1 r_h) + r_h v_2\right) \\
&= v_1 \sum_{k=0}^{\infty} \frac{(\lambda_2 r_h)^k}{k!} - v_1 - \frac{\lambda_2 v_1}{\lambda_1} \sum_{k=2}^{\infty} \frac{(\lambda_1 r_h)^k}{k!} - \lambda_2 v_2 r_h \\
&= \sum_{k=2}^{\infty} \frac{v_1 \lambda_2 (r_h)^k \left((\lambda_2)^{k-1} - (\lambda_1)^{k-1}\right)}{k!} + v_1 + v_1 \lambda_2 r_h - v_1 - v_2 \lambda_2 r_h \\
&< 0,
\end{aligned}$$

where the last inequality holds because $v_1 < v_2$, and $\lambda_1 > \lambda_2$.

Case 2.2: $r - R_{11}(r, v) \leqslant h \leqslant r$

$$\begin{aligned}
H(h) &= V_1(r-h, v + v_1 h) - E_2[V_1(r-h-T, v + v_1 h + v_2 T)] \\
&= (v + v_1 h) - E_2[(v + v_1 h + v_2 T) I_{T \leqslant r-h}] \\
&= (v + v_1 h) - \int_0^{r-h} \lambda_2 e^{-\lambda_2 t}(v + v_1 h + v_2 t) dt
\end{aligned}$$

$$\begin{aligned}
H'(h) &= v_1 - \int_0^{r-h} \lambda_2 e^{-\lambda_2 t}(v + v_1 h + v_2 t)_h dt \\
&\quad - (r-h)_h \lambda_2 e^{-\lambda_2 t}(v + v_1 h + v_2 t)|_{t=r-h} \\
&= v_1 - v_1(1 - e^{-\lambda_2(r-h)}) + \lambda_2 e^{-\lambda_2 t}(v + v_1 h + v_2 t)|_{t=r-h} \\
&= v_1 e^{-\lambda_2(r-h)} + \lambda_2 e^{-\lambda_2 t}(v + v_1 h + v_2 t)|_{t=r-h} \geqslant 0.
\end{aligned}$$

Combining Case 2.1 and Case 2.2, we've shown $H(h)$ is always increasing in h when $h \in [0, r]$ under Case 2. □

A.2. Proof of Lemma 3

Case 1: $v \geqslant v_1(r)$.

Similar to the proof of Case 1 in Lemma 2, this part of Lemma 3 can immediately be proved because the state (r, v) is already in stopping domain, i.e., (r, v) is above both $v_1(\cdot)$ and $v_2(\cdot)$.

Case 2: $v < v_1(r)$.

Case 2.1: $h < r - R_{21}(r, v)$. Given

$$V_1(r, v) \geqslant E_2[V_1(r - T, v + v_2 T)],$$

we have:

$$\begin{aligned}
V_1(r, v) &\geqslant e^{-\lambda_2 h} E_2[V_1(r - T, v + v_2 T) | T \\
&\geqslant h] + (1 - e^{-\lambda_2 h}) E_2[V_1(r - T, v + v_2 T) | T < h].
\end{aligned}$$

Because of the memoryless property of exponentially distributed r.v.,

$$E_2[V_1(r - T, v + v_2 T) | T \geqslant h] = E_2[V_1(r - h - T, v + v_2 h + v_2 T)].$$

When $0 \leqslant t \leqslant h < r - R_{21}(r, v)$,

$$V_1(r - t, v + v_2 t) \geqslant V_1(r, v),$$

which implies:

$$E_2[V_1(r - T, v + v_2 T) | T < h] \geqslant V_1(r, v).$$

Therefore,

$$V_1(r, v) \geqslant e^{-\lambda_2 h} E_2[V_1(r - h - T, v + v_2 h + v_2 T)] + (1 - e^{-\lambda_2 h}) V_1(r, v),$$

i.e.,

$$V_1(r, v) \geqslant E_2[V_1(r - h - T, v + v_2 h + v_2 T)].$$

Because

$$V_1(r - h, v + v_2 h) \geqslant V_1(r, v), \forall h < r - R_{21}(r, v),$$

we've shown:

$$V_1(r - h, v + v_2 h) \geqslant E_2[V_1(r - h - T, v + v_2 h + v_2 T)], \quad \forall h < r - R_{21}(r, v).$$

Case 2.2: $r - R_{21}(r, v) \leqslant h \leqslant r$.

As in the proof of Lemma 2, let's define:

$$H(h) := V_1(r - h, v + v_2 h) - E_2[V_1(r - h - T, v + v_2 h + v_2 T)].$$

It's easy to see $H(h)$ is a continuous function of h, $\forall h \leqslant r$. When $r - R_{21}(r, v) \leqslant h \leqslant r$, we have:

$$\begin{aligned}
H(h) &= V_1(r - h, v + v_2 h) - E_2[V_1(r - h - T, v + v_2 h + v_2 T)] \\
&= (v + v_2 h) - \int_0^{r-h} \lambda_2 e^{-\lambda_2 t}(v + v_2 h + v_2 t) dt.
\end{aligned}$$

Differentiation yields:

$$\begin{aligned}
H'(h) &= v_2 - \int_0^{r-h} \lambda_2 e^{-\lambda_2 t}(v + v_2 h + v_2 t)_h dt \\
&\quad - (r-h)_h \lambda_2 e^{-\lambda_2 t}(v + v_2 h + v_2 t)|_{t=r-h} \\
&= v_2 - v_2(1 - e^{-\lambda_2(r-h)}) + \lambda_2 e^{-\lambda_2(r-h)}(v + v_2 r) \\
&= v_2 e^{-\lambda_2(r-h)} + \lambda_2 e^{-\lambda_2(r-h)}(v + v_2 r) > 0.
\end{aligned}$$

From Case 2.1,

$$H(h)|_{h=r-R_{21}(r,v)} \geqslant 0.$$

Therefore,

$$H(h) \geqslant 0, \quad \forall r - R_{21}(r, v) \leqslant h \leqslant r,$$

which concludes the proof. □

A.3. Proof of Corollary 1

Given

$$V_1(r, v) \geqslant E_2[V_1(r - T, v + v_2 T)],$$

by Lemmas 2 and 3, for $i = 1, 2$, we have

$$V_1(r - h, v + v_i h) \geqslant E_2[V_1(r - h - T, v + v_i h + v_2 T)], \quad \forall h \in [0, r],$$

which implies:

$$V_2(r - h, v + v_i h) = V_1(r - h, v + v_i h), \quad \forall h \in [0, r].$$

Therefore,

$$LHS = V_2(r, v) = V_1(r, v),$$

and

$$\begin{aligned}
RHS &= \max\left\{v, \max_{i=1,2}\left\{\int_0^r \lambda_i e^{-\lambda_i t} V_2(r - t, v + v_i t) dt\right\}\right\} \\
&= \max\left\{v, \max_{i=1,2}\left\{\int_0^r \lambda_i e^{-\lambda_i t} V_1(r - t, v + v_i t) dt\right\}\right\} \\
&= \max\left\{v, \max_{i=1,2}\{E_i[V_1(r - T, v + v_i T)]\}\right\}.
\end{aligned}$$

Because

$$V_1(r, v) = \max\{v, E_1[V_1(r - T, v + v_1 T)]\},$$

and

$$V_1(r, v) \geqslant E_2[V_1(r - T, v + v_2 T)],$$

we have:

$$RHS = \max\left\{v, \max_{i=1,2}\{E_i[V_1(r - T, v + v_i T)]\}\right\} = V_1(r, v) = LHS,$$

which concludes the proof. □

A.4. Proof of Lemma 4

Given

$$V_1(r,v) \leqslant E_2[V_1(r-T, v+v_2T)], \tag{b.1}$$

we want to prove:

$$V_2(r,v) = \max\left\{v, \max_{i=1,2}\{E_i[V_2(r-T, v+v_iT)]\}\right\}. \tag{b.2}$$

We consider two different cases to proceed the proof on whether the current state (r,v) is above the curve $v_1(\cdot)$ or not (see Fig. 4).

Case 1: $V_1(r,v) = v$.

We have $v \geqslant v_1(r)$ and

$$V_1(r-t, v+v_it) = v + v_it, \quad \forall t \in [0,r], \quad i = 1,2.$$

From the above observation, we know that *Policy Two* for the system starting at state (r,v) is equivalent to the one-stage look-ahead stopping rule as if only type 2 items are available. From Proposition 2, when $v \geqslant v_1(r)$, it is never optimal to put in type 1 items at state (r,v) and the following states. Therefore, *Policy Two* is the optimal policy under this case, which implies the equation (b.2).

Case 2: $V_1(r,v) > v$.

We first make two definitions:

Good Condition: if state (r,v) satisfies $V_1(r,v) \geqslant E_2[V_1(r-T, v+v_2T)]$, we say the state (r,v) meets *Good Condition*.

Critical Condition: if state (r,v) satisfies $V_1(r,v) = E_2[V_1(r-T, v+v_2T)]$, we say the state (r,v) meets *Critical Condition* and (r,v) is a **Critical Point**.

On the line $\{(r-t, v+v_2t) : 0 \leqslant t \leqslant r\}$, denote (r',v') as the rightmost point that satisfies *Good Condition*. It is easy to see that there must exist such (r',v') given the inequality (b.1). According to Lemma 3, we know that the state $(r'-h, v'+v_2h)$ satisfies *Good Condition* for all $0 \leqslant h \leqslant r'$; and because (r',v') is the rightmost point on the line $\{(r-t, v+v_2t) : 0 \leqslant t \leqslant r\}$, any state $(r'+h, v'-v_2h)$, where $h > 0$, cannot satisfy *Good Condition*. The continuity of all the functions involved here implies that (r',v') is a *Critical Point*. To complete the proof, we consider two subcases.

Subcase 2.1: $r' \geqslant R_{21}(r,v)$

Starting from state (r,v), note $r' < r$, by *Policy Two*, we will keep inserting items of type 2 until the remaining capacity is less than r'. By the memoryless property of exponential r.v.s, we have:

$$\begin{aligned} E_2[V_2(r-T, v+v_2T)] &= E_2[V_2(r'-T, v'+v_2T)] \\ &= E_2[V_1(r'-T, v'+v_2T)] = V_1(r',v') \\ &= v_1(R_{11}(r',v')), \end{aligned}$$

where $R_{11}(r',v')$ is well defined since $r' \geqslant R_{21}(r,v)$.

Because $(R_{11}(r',v'), v_1(R_{11}(r',v')))$ must satisfy *Good Condition* by Lemma 2, this point must be on the left side of $v_2(\cdot)$, i.e.,

$$v_2(R_{11}(r',v')) \leqslant v_1(R_{11}(r',v')).$$

We first want to show: for all points (r_h, v_h) on the line segment $[(R_{11}(r,v), v_1(R_{11}(r,v))), (r,v)]$,[2] we always have $V_2(r_h, v_h) \leqslant v_1(R_{11}(r',v'))$.

Because all points on $[(R_{11}(r',v'), v_1(R_{11}(r',v'))), (r',v')]$ satisfy *Good Condition*, for all the lines of slope $-v_2$ which intersect with this line segment, the *Critical Points* on those lines must be below $[(R_{11}(r',v'), v_1(R_{11}(r',v'))), (r',v')]$. For any point $(r-h, v+v_1h)$, where $0 \leqslant h \leqslant (r'-R_{11}(r',v'))$, either it satisfies *Good Condition*, or there exists one point which is on the line $\{(r-h-t, v+v_1h+v_2t) : 0 \leqslant t \leqslant r-h\}$ but below $[(R_{11}(r',v'), v_1(R_{11}(r',v'))), (r',v')]$, and this point satisfies *Critical Condition*. For both cases

$$V_2(r-h, v+v_1h) \leqslant v_1(R_{11}(r',v')), \quad \forall 0 \leqslant h \leqslant (r'-R_{11}(r',v')).$$

Given a point (r_h, v_h) which is on $[(R_{11}(r,v), v_1(R_{11}(r,v))), (r-(r'-R_{11}(r',v')), v+(r'-R_{11}(r',v'))v_1)]$, if (r_h, v_h) satisfies *Good Condition*, then

$$V_2(r_h, v_h) = V_1(r_h, v_h) = v_1(R_{11}(r,v)) \leqslant v_1(R_{11}(r',v')).$$

If (r_h, v_h) does not satisfy *Good Condition*, let's denote (r'_h, v'_h) as the *Critical Point* on the line $\{(r_h-t, v_h+v_2t) : 0 \leqslant t \leqslant r-h\}$. Then (r'_h, v'_h) can be either on right side of $v_1(\cdot)$, or on left side of $v_1(\cdot)$. On the former case,

$$V_2(r'_h, v'_h) = v_1(R_{11}(r'_h, v'_h)) \leqslant v_1(R_{11}(r',v'));$$

on the latter case, because (r'_h, v'_h) must be on $v_2(\cdot)$,

$$V_2(r'_h, v'_h) = v'_h = v_2(r'_h) \leqslant v_2(R_{11}(r',v')) \leqslant v_1(R_{11}(r',v')).$$

Therefore, if (r_h, v_h) does not satisfy *Good Condition*,

$$V_2(r_h, v_h) = V_2(r'_h, v'_h) \leqslant v_1(R_{11}(r',v')).$$

Thus for any (r_h, v_h) on $[(R_{11}(r,v), v_1(R_{11}(r,v))), (r-(r'-R_{11}(r',v')), v+(r'-R_{11}(r',v'))v_1)]$,

$$V_2(r_h, v_h) \leqslant v_1(R_{11}(r',v'))$$

In all the discussions so far, we've proved that for all points (r_h, v_h) on the line interval $[(R_{11}(r,v), v_1(R_{11}(r,v))), (r,v)]$,

$$V_2(r_h, v_h) \leqslant v_1(R_{11}(r',v')).$$

Now we'll use this result to prove the rest of the lemma.

If (r,v) is above the curve $v_2(\cdot)$, then the proof is trivial. So in the following we always assume that (r,v) is below $v_2(\cdot)$ so that $R_{12}(r,v)$ is well defined.

If $R_{12}(r,v) < R_{11}(r,v)$, let's denote (r'',v'') be the *Critical Point* on the line $\{(r-t, v+v_1t) : 0 \leqslant t \leqslant r\}$, then we must have $r'' = R_{12}(r,v)$.

Because

$$r'' < R_{11}(r,v) < R_{11}(r',v'),$$

by the monotonicity of $v_2(\cdot)$,

$$V_2(r'',v'') = V_1(r'',v'') = v'' = v_2(r'') < v_2(R_{11}(r',v')) \leqslant v_1(R_{11}(r',v')).$$

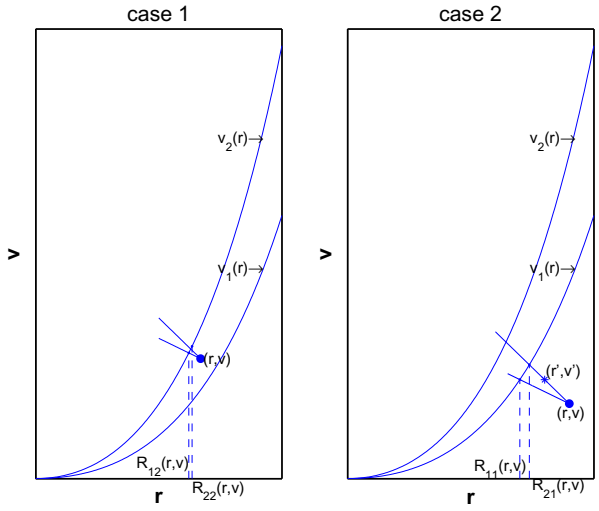

Fig. 4. Given $V_1(r,v) \leqslant E_2[V_1(r-T, v+v_2T)]$, Case 1 considers $V_1(r,v) = v$, i.e., $v \geqslant v_1(r)$; Case 2 considers $V_1(r,v) > v$, i.e., $v < v_1(r)$.

[2] For any two points (r_b, v_b) and (r_e, v_e), we denote $[(r_b, v_b), (r_e, v_e)]$ as the line segment which connects the two points.

For all points (r_h, v_h) on the line $[(r'', v''), (R_{11}(r, v), v_1(R_{11}(r, v)))]$, we have

$$V_2(r_h, v_h) = v_2(R_{22}(r_h, v_h)) < v_2(R_{11}(r', v')) \leqslant v_1(R_{11}(r', v')).$$

Therefore:

$$\begin{aligned}
&E_1[V_2(r - T, v + v_1T)] = P_1(T > r - r'')\\
&E_1[V_2(r - T, v + v_1T)|T > r - r''] + P_1(T \leqslant r - r'')\\
&E_1[V_2(r - T, v + v_1T)|T \leqslant r - r''] \leqslant P_1(T > r - r'')\\
&E_1[(v'' + v_1T)I_{T\leqslant r''}] + P_1(T \leqslant r - r'')v_1(R_{11}(r', v'))\\
&\quad \leqslant P_1(T > r - r'')v'' + P_1(T \leqslant r - r'')v_1(R_{11}(r', v')) \leqslant v_1(R_{11}(r', v'))
\end{aligned}$$

If $R_{12}(r, v) \geqslant R_{11}(r, v)$, denote $(r'', v'') = (R_{11}(r, v), v_1(R_{11}(r, v)))$, using the exact same procedure as the above, we can see:

$$E_1[V_2(r - T, v + v_1T)] \leqslant v_1(R_{11}(r', v')).$$

So we've proved under *Subcase 2.1*,

$$E_1[V_2(r - T, v + v_1T)] \leqslant v_1(R_{11}(r', v')).$$

Combining the above result with

$$E_2[V_2(r - T, v + v_2T)] = v_1(R_{11}(r', v')),$$

and

$$E_2[V_2(r - T, v + v_2T)] > v,$$

we have:

$$\begin{aligned}
V_2(r, v) &= E_2[V_2(r - T, v + v_2T)]\\
&= \max\left\{v, \max_{i=1,2}\{E_i[V_2(r - T, v + v_iT)]\}\right\}.
\end{aligned}$$

Subcase 2.2: $r' < R_{21}(r, v)$.

It's easy to see that (r', v') is above the curve $v_1(\cdot)$. Because we assume that (r', v') is a *Critical Point*, (r', v') must be on the curve $v_2(\cdot)$ and we have:

$$V_2(r, v) = E_2[V_2(r - T, v + v_2T)] = v'.$$

Let (r'', v'') be the *Critical Point* on the line $\{(r - t, v + v_1t) : 0 \leqslant t \leqslant r\}$.

If $r'' < R_{11}(r, v)$, then we must have $r'' < r'$, and

$$V_2(r'', v'') = v'' = v_2(r'') < v_2(r') = v'.$$

If $r'' \geqslant R_{11}(r, v)$, then

$$V_2(r'', v'') = v_1(R_{11}(r, v)) \leqslant v_1(R_{21}(r, v)) < v'.$$

We can use similar arguments as for *Subcase 2.1* to show that for all the points (r_h, v_h) on the line interval $[(r'', v''), (r, v)]$, we have:

$$V_2(r_h, v_h) \leqslant v',$$

which implies:

$$E_1[V_2(r - T, v + v_1T)] \leqslant v' = E_2[V_2(r - T, v + v_2T)] = V_2(r, v).$$

Therefore under *Subcase 2.2* we can prove:

$$V_2(r, v) = \max\left\{v, \max_{i=1,2}\{E_i[V_2(r - T, v + v_iT)]\}\right\}.$$

Above all, Lemma 4 is proved. □

A.5. Proof of Proposition 3

Let's define:

$$f(r) = v_1(r) - v_2(r),$$

then

$$f'(r) = v_1(e^{\lambda_1 r} - 1) - v_2(e^{\lambda_2 r} - 1),$$

and

$$f''(r) = v_1\lambda_1 e^{\lambda_1 r} - v_2\lambda_2 e^{\lambda_2 r}.$$

We also have:

$$f(0) = f'(0) = 0, \quad f''(0) = v_1\lambda_1 - v_2\lambda_2.$$

Let's first assume $v_1\lambda_1 \geqslant v_2\lambda_2$.

Because $\lambda_1 > \lambda_2$, $\forall r > 0$,

$$\begin{aligned}
f''(r) = v_1\lambda_1 e^{\lambda_1 r} - v_2\lambda_2 e^{\lambda_2 r} \geqslant v_2\lambda_2(e^{\lambda_1 r} - e^{\lambda_2 r}) > 0 \Rightarrow f'(r) > f'(0)\\
= 0 \Rightarrow f(r) > f(0) = 0,
\end{aligned}$$

which proves the first case in the proposition.

Now let's assume $v_1\lambda_1 < v_2\lambda_2$.

We have

$$f''(0) = v_1\lambda_1 - v_2\lambda_2 < 0,$$

and given $\lambda_1 > \lambda_2$, we also know

$$f''(+\infty) > 0.$$

The continuity of $f''(r)$ implies that there must exist some points $r, r > 0$, such that $f''(r) = 0$. Let's define $r_2 = \inf\{r : f''(r) \geqslant 0\}$. $\forall \delta > 0$, we have:

$$f''(r_2 + \delta) = v_1\lambda_1 e^{\lambda_1 r_2} e^{\lambda_1 \delta} - v_2\lambda_2 e^{\lambda_2 r_2} e^{\lambda_2 \delta} > e^{\lambda_2 \delta} f''(r_2) = 0.$$

Therefore,

$$f''(r) < 0, \forall 0 \leqslant r < r_2; \quad f''(r) > 0, \forall r > r_2.$$

Because $f'(0) = 0$ and $f'(+\infty) > 0$, the above result implies that there exists r_1, where $r_1 > r_2$, such that:

$$f'(r) < 0, \forall 0 < r < r_1 \quad ; f'(r) > 0, \forall r > r_1.$$

With the facts that $f(0) = 0$ and $f(+\infty) > 0$, we know from the above observation that there exists r_0, where $r_0 > r_1$, such that:

$$f(r) < 0, \forall 0 < r < r_0; \quad f(r) > 0, \forall r > r_0.$$

This proves the second case in the proposition. □

References

Chen, K., & Ross, S. M. (2013). An adaptive stochastic knapsack problem with exponential capacity (work on progress).

Chen, K., & Ross, S. M. (2013). Static stochastic knapsack problems (work on progress).

Cohn, A. M., & Mit, C. B. (1998). The stochastic knapsack problem with random weights: A heuristic approach to robust transportation planning. In *Proceedings of the Triennial Symposium on Transportation Analysis. Citeseer*.

Dean, B. C., Goemans, M. X., & Vondrdk, J. (2004). Approximating the stochastic knapsack problem: The benefit of adaptivity. In *Proceedings. 45th Annual IEEE Symposium on Foundations of Computer Science, 2004* (pp. 208–217). IEEE.

Derman, C., Lieberman, G. J., & Ross, S. M. (1978). A renewal decision problem. *Management Science, 24*(5), 554–561.

Ferguson, T. S. (2012). Optimal stopping and applications.

Iravani, S. M. R., Ilhan, T., & Daskin, M. S. (2011). The adaptive knapsack problem with stochastic rewards. *Operations Research, 59*(1), 242–248.

Kellerer, H., Pferschy, U., & Pisinger, D. (2004). *Knapsack problems*. Berlin, Heidelberg: Springer.

Kleywegt, A. J., & Papastavrou, J. D. (1998). The dynamic and stochastic knapsack problem. *Operations Research, 46*, 17–35.

Kleywegt, A. J., & Papastavrou, J. D. (2001). The dynamic and stochastic knapsack problem with random sized items. *Operations Research, 49*(1), 26–41.

Kosuch, S., & Lisser, A. (2010). Upper bounds for the 0–1 stochastic knapsack problem and a branch-and-bound algorithm. *Annals of Operations Research, 176*, 77–93. http://dx.doi.org/10.1007/s10479-009-0577-5.

Kosuch, S., & Lisser, A. (2011). On two-stage stochastic knapsack problems. *Discrete Applied Mathematics, 159*(16), 1827–1841.

Lee, T.-E., & Oh, G. T. (1997). The asymptotic value-to-capacity ratio for the multi-class stochastic knapsack problem. *European Journal of Operational Research, 103*(3), 584–594.

Lin, G. Y., Lu, Y., & Yao, D. D. (2008). The stochastic knapsack revisited: Switch-over policies and dynamic pricing. *Operations Research, 56*(4), 945–957.

Lu, L. L., Chiu, S. Y., & Cox, L. A. Jr., (1999). Optimal project selection: Stochastic knapsack with finite time horizon. *The Journal of the Operational Research Society, 50*(6), 645–650.

Martello, S., Pisinger, D., & Toth, P. (2000). New trends in exact algorithms for the 0–1 knapsack problem. *European Journal of Operational Research, 123*(2), 325–332.

Merzifonluoğlu, Y., Geunes, J., & Romeijn, H. E. (2012). The static stochastic knapsack problem with normally distributed item sizes. *Mathematical Programming, 134*(2), 459–489.

Ross, K. W., & Tsang, D. H. K. (1989). The stochastic knapsack problem. *IEEE Transactions on Communications, 37*(7), 740–747.

Schilling, K. (1994). Random knapsacks with many constraints. *Discrete Applied Mathematics, 48*(2), 163–174.

Smith, D. R. (1978). Note on 'a renewal decision problem'. *Management Science, 24*(5).

Van Slyke, R., & Young, Y. (2000). Finite horizon stochastic knapsacks with applications to yield management. *Operations Research, 48*(1), 155–172.