Contents lists available at ScienceDirect

Computers & Operations Research

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

An agent-based stochastic ruler approach for a stochastic knapsack problem with sequential competition

Matthew R. Gibson[a], Jeffrey W. Ohlmann[b,*], Michael J. Fry[c]

[a]Department of Computer Science, University of Iowa, 14 MacLean Hall, Iowa City, IA 52242-1419, USA
[b]Department of Management Sciences, University of Iowa, 108 John Pappajohn Business Building, Iowa City, IA 52242-1994, USA
[c]Department of Quantitative Analysis and Operations Management, University of Cincinnati, 532 Carl H. Lindner Hall, Cincinnati, OH 45221-0130, USA

ARTICLE INFO

Keywords:
Knapsack problem
Sequential allocation
Discrete stochastic optimization
Stochastic ruler
Agent-based simulation
Beam search

ABSTRACT

We examine a situation in which a decision-maker executes a sequence of resource allocation decisions over time, but the availability of the indivisible resources at future epochs is uncertain due to actions of competitors. We cast this problem as a specialized type of stochastic knapsack problem in which the uncertainty of item (resource) availability is induced by competitors concurrently filling their own respective knapsacks. Utilizing a multi-period bounded multiple-choice knapsack framework, we introduce a general discrete stochastic optimization model that allows a nonlinear objective function, cardinality constraints, and a knapsack capacity constraint. Utilizing a set of greedy selection rules and agent-based modeling to simulate the competitors' actions, we solve the problem with a stochastic ruler approach that incorporates beam search to determine item selection of the types specified by the solution representation. We illustrate the computational effectiveness of our approach on instances motivated by a sports league draft as well as generic problem instances based on the knapsack literature.

1. Introduction

We consider a sequential decision-making problem in which a decision-maker (DM) and a collection of competitors take turns (according to a prespecified order) allocating indivisible resources. Due to the actions of its competitors, the DM faces uncertainty regarding the availability of resources at future decision epochs, thereby possibly impacting the decision at the current epoch. Instances of this problem arise in applications in which participants make one-at-a-time allocation decisions which affect the future choices of the other competitors. Such a situation arises in sports league drafts [4,13], political cabinet appointments [6], and the distribution of assets from an estate [28].

In a professional sports league draft, competing franchises sequentially select athletes from a common pool in order to upgrade their respective rosters. Once an athlete is selected by a franchise, this player is no longer eligible for selection in the remainder of the draft process. To apportion cabinet ministries among constituent political parties, parliamentary democracies have used sequential appointment of various posts [5]. The selection ordering of the political parties is typically based on a characteristic such as party size so that the larger political parties have the opportunity to select preferred ministry positions (and perhaps more positions overall). Similarly, the distribution of an estate may be administered by having heirs select, in a sequence based on age, the specific items to fulfill their inheritance. We generalize the application-specific considerations (constraints, objective, etc.) of resource allocation subject to sequential competition by casting this problem as a specialized type of stochastic knapsack problem in which the uncertainty of item (resource) availability is induced by competitors concurrently filling their own respective knapsacks.

The notion of sequential selection under competition-induced uncertainty characterizes a *sequential game*. A sequential game (also called a *stochastic game* or a *Markov game*) is a decision process with multiple participants in which each participant makes a decision at discrete points of time called decision epochs [16]. In a general sequential game, the participants simultaneously determine their decisions at each epoch; these actions are not revealed to the group until the end of the epoch after all participants have executed their actions. In our treatment, we consider a special case in which the participants execute their selections sequentially according to a priority ordering (previously established by a bidding process or some other mechanism external to the actual selection process) so that each participant is aware of all previous selections.

* Corresponding author. Tel.: +1 319 335 0837; fax: +1 319 335 0297.
E-mail addresses: mrgibson@cs.uiowa.edu (M.R. Gibson), jeffrey-ohlmann@uiowa.edu (J.W. Ohlmann), mike.fry@uc.edu (M.J. Fry).

0305-0548/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2009.02.028

Sequential games conceptually generalize *Markov decision processes* [24] and *game theory* [14], and consequently suffer the similar fate of general intractability as the number of participants and possible actions increase. To solve our specialized stochastic knapsack problem, we present a discrete stochastic optimization approach utilizing thestochastic ruler algorithm [30,1] and an agent-based simulation of the competitors' actions [3] to overcome the computational challenges. This paper makes a contribution with respect to heuristic methodology by presenting the first metaheuristic hybridizing a stochastic ruler algorithm, beam search, and agent-based modeling within a simulation–optimization framework. This stochastic optimization approach is also the first to explicitly consider the competition-induced uncertainty in a resource allocation problem with sequential competition.

The remainder of this paper is outlined as follows. In Section 2, we present a stochastic knapsack formulation of our problem and we provide details of our solution approach. We provide computational experience in Section 3 to demonstrate our approach. We summarize the paper in Section 4 with concluding remarks and a discussion of future work.

2. Discrete stochastic optimization model

As a classical combinatorial optimization problem with application in many disciplines, the *knapsack problem* and its variants are the subject of extensive research literature [20,17]. The objective of a knapsack problem is to select items from a given set in order to maximize the total reward subject to constraints on knapsack capacity dimensions. The *generalized multiple-choice* knapsack problem further partitions the set of candidate items into classes and enforces constraints on the number of items which can be selected from each item class [17]. Of particular relevance is the *bounded multiple-choice* knapsack problem, a special case of the generalized multiple-choice knapsack problem which places upper and lower bounds on the number of items which can be selected from each item class. In this paper, we consider a *multi-period* bounded multiple-choice knapsack problem in which a decision-maker must make selections at decision epochs over time and the availability of items at future epochs is uncertain due to the actions of competitors who sequentially make selections to fill their own respective knapsacks. For brevity, we will refer to this problem as the knapsack problem with sequential competition (KPSC).

With regard to the consideration of uncertainty, the research literature contains work on two distinct *stochastic knapsack problems* (SKP), the "dynamic" and the "static" versions. In a dynamic SKP, items stochastically arrive over time and must be accepted or rejected upon arrival without knowledge of the items that will be available for consideration in the future [26,9,18]. The KPSC differs fundamentally from the dynamic SKP. In contrast to the dynamic SKP, the KPSC assumes that the DM has full knowledge of the set of items available for selection and all items are available at the initial decision epoch.

Henig [15], Carraway et al. [7], and Morton and Wood [22] address static integer SKP formulations in which the reward of each item is an independent normal random variable, the item weights are deterministic, and the objective is to maximize the probability of meeting a total return threshold. Morton and Wood [22] also consider a Monte Carlo approximation procedure to solve static SKPs with general distributions on the random rewards. It is possible to formulate the KPSC as a special case of a static multi-period SKP with random rewards in which the likelihood of an item being unavailable at a decision epoch is reflected in the probability of a zero reward. Solving such a formulation, however, requires the ability to evaluate an exponential number of scenarios (and the corresponding probabilities of these scenarios). In this paper, we present an alternative formulation that facilitates the application of a simulation-based heuristic approach.

The KPSC uniquely combines the features of uncertainty and multiple-choice constraints in a multi-period setting. To formally present the KPSC,we introduce some notation. Let $\mathscr{M}=\{1,\dots,M\}$ be the set of participants, where participant 1 denotes the decision-maker. For convenience, let $\mathscr{C}=\mathscr{M}\backslash\{1\}$ be the set of the DM's competitors. Define $\mathscr{K}$ as the set of distinct item types and $\mathscr{L}_k$ as the set of items of type $k\in\mathscr{K}$. For $k\in\mathscr{K}$ and $l\in\mathscr{L}_k$, denote the lth item of type k as i_{kl}. The individual reward associated with i_{kl} from the perspective of participant $m\in\mathscr{M}$ is r^m_{kl}. For each $k\in\mathscr{K}$, we order the items $i_{k1},i_{k2},\dots,i_{k|\mathscr{L}_k|}$ such that $r^1_{k1}\geqslant r^1_{k2}\geqslant\cdots\geqslant r^1_{k|\mathscr{L}_k|}$. The capacity of participant m's knapsack is given by the scalar b^m, and item i_{kl} consumes a_{kl} units of the knapsack capacity if selected. In addition, there are upper and lower bounds on the allowed number of items of type k in participant m's knapsack, u^m_k and ℓ^m_k, respectively.

Within the context of a sports draft, the distinct item types correspond to the various player positions or skills. Sports teams are composed of specific player positions that demand specific skill sets or player types. For instance, a baseball team is made up of pitchers, outfielders, shortstops, etc; American football teams require quarterbacks, offensive linemen, wide receivers, etc. With few exceptions, the player skills required to play one position do not translate well to other positions. Because sports teams are constantly trying to fill vacancies in their rosters (due to player retirements, free agent departures, injuries, etc.), each team tends to enter the draft with a specific set of positions for which it would like to acquire players in the draft. The upper and lower bounds on the number ofplayers of position k to be selected by team m, u^m_k, and ℓ^m_k, represent team m's preferences. The salary demands of a selected player may also be an important factor, especially in sports leagues (such as Major League Baseball) which do not level the playing field between small- and large-market franchises by enforcing a league-wide salary cap [19]. We express financial considerations by assuming that each team m has a fiscal budget of b^m units and the selection of player i_{kl} requires an expenditure of a_{kl} units.

The stochastic nature of the KPSC is due to the DM's lack of knowledge regarding the future selections of its competitors and the corresponding uncertainty about the availability of items at future decision epochs. Let T be the finite number of decision epochs at which the participants execute item selections for their respective knapsacks. At each decision epoch $t\in\{1,\dots,T\}$, the DM and competitors sequentially select items, in a prespecified order, to insert into their respective knapsacks. Without loss of generality, we assume that the DM makes the first selection at each decision epoch. We assume that the number and order of competitors making a selection at each decision epoch t is known, but these parameters may vary across decision epochs. Denote the ordered set of competitors making a selection at decision epoch t as $\mathscr{R}_t\subseteq\mathscr{M}$. For $t\in\{1,2,\dots,T\}$ and $d\in\{1,2,\dots,|\mathscr{R}_t|\}$, we denote the selection of item i_{kl} (for some $k\in\mathscr{K}$ and $l\in\mathscr{L}_k$) by the dth competitor in the sequence at decision epoch t as $y_{td}=i_{kl}$. As each competitor's actions are typically not known a priori by the DM (or the other competitors) with certainty, y_{td} is a discrete random variable whose distribution is dependent on the stochastic process, $\tilde{\xi}_{td}=\{y_{11},y_{12},\dots,y_{1|\mathscr{R}_1|},y_{21},y_{22},\dots,y_{2|\mathscr{R}_2|},\dots,y_{t1},y_{t2},\dots,y_{t,d-1}\}$, describing all of the item selections prior to the dth selection at decision epoch t. Note that y_{t1} corresponds to the DM's selection at decision epoch t, for $t=1,\dots,T$. Furthermore, we denote a realization of $\tilde{\xi}_{td}$ as ξ_{td}.

Determining an optimal selection strategy is difficult due to the large number of possible action choices and scenarios facing the DM. To help overcome this complexity, rather than specifying the particular item to select at each decision epoch, we define the vector $\mathbf{x}=(x_1,\dots,x_T)$ to be the DM's *selection policy*, where $x_t\in\mathscr{K}$ specifies the *item type* that the DM selects at decision epoch t. Setting $x_t=k\in$

$\mathscr{K}$ does not specify the selection of a specific item, i_{kl} for some $l \in \mathscr{L}_k$, but rather only outlines a strategy of selecting an item of type k at decision epoch t; the actual item of type k the DM selects at decision epoch t will depend on the observed realization of $\tilde{\xi}_{t1}$. Given a realized selection sequence, ξ_{t1}, and the DM's current item type specification, x_t, the function $g(\xi_{t1}, x_t)$ identifies the DM's item selection, $i_{x_t,\ell}$, at epoch t. The evolution of the selection process depends on $\mathbf{x}$; specifically $\tilde{\xi}_{t1}$ is a function of $g(\tilde{\xi}_{11}, x_1), g(\tilde{\xi}_{21}, x_2), \ldots, g(\tilde{\xi}_{t-1,1}, x_{t-1})$. We notate the entire stochastic process encompassing all decision epochs as $\tilde{\xi}_T(\mathbf{x})$ and correspondingly let $\xi_T(\mathbf{x})$ be a realization of an entire selection process, i.e., all $\sum_{t=1}^{T} |\mathscr{R}_t|$ total selections.

We define $v^1(\mathbf{i})$ to be a deterministic, real-valued function which computes the value of the vector of items, $\mathbf{i}$, inserted in the DM's knapsack. The only assumption that we enforce on the form of v^1 is that it is non-increasing with increasing $l \in \mathscr{L}_k$. That is, if $l < n$ (and therefore $r^1_{kl} \geqslant r^1_{kn}$), then $v^1(i_1, i_2, \ldots, i_{kl}, \ldots, i_T) \geqslant v^1(i_1, i_2, \ldots, i_{kn}, \ldots, i_T)$. We define $\mathscr{A}(k, \xi_{td})$ as the set of items of type k available at the dth selection of epoch t in realization ξ_{td}. At the onset of the problem, we assume that all items are available to the DM and the competitors; item unavailability in the future is due to the item being previously selected by a participant. We formally state the KPSC as

$$\max \quad E_{\tilde{\xi}[v^1(\mathbf{g}(\tilde{\xi}_T(\mathbf{x})))]} \tag{1}$$

$$\text{s.t.} \quad \sum_{t=1}^{T} I_{\{x_t = k\}} \leqslant u^1_k \quad \forall k \in \mathscr{K}, \tag{2}$$

$$\sum_{t=1}^{T} I_{\{x_t = k\}} \geqslant \ell^1_k \quad \forall k \in \mathscr{K}, \tag{3}$$

$$\sum_{t=1}^{T} a_{x_t, h(g(\xi_{t1}, x_t))} \leqslant b^1 \quad \forall \xi_T(\mathbf{x}), \tag{4}$$

$$g(\xi_{t1}, x_t) \in \mathscr{A}(x_t, \xi_{t1}), \quad t = 1, \ldots, T, \quad \forall \xi_T(\mathbf{x}). \tag{5}$$

In (1), we define the objective function for our discrete stochastic optimization problem, where $\mathbf{g}(\tilde{\xi}_T(\mathbf{x}))$ is the vector corresponding to $(g(\tilde{\xi}_{11}, x_1), g(\tilde{\xi}_{21}, x_2), \ldots, g(\tilde{\xi}_{T1}, x_T))$. The general formulation of (1) allows for the maximization of expected value as well as the maximization of the probability of exceeding a specified threshold value. We enforce the cardinality restrictions on the selection policy via constraints (2) and (3); we express these restrictions solely as a function of $\mathbf{x}$ and therefore can assure their satisfaction without consideration of $\tilde{\xi}_T(\mathbf{x})$. Constraint (4) enforces the capacity limitation of the knapsack, where h is a function which returns an item's index, i.e., $h(i_{kl}) = l \in \mathscr{L}_k$. Constraint (5) only allows the selection of available items. To guarantee the feasibility of a selection policy $\mathbf{x}$ with respect to (4) and (5) for all possible realizations of $\tilde{\xi}_T(\mathbf{x})$, we assume that there is a sufficient supply of items of each type that will fit into the knapsack. The existence of a sufficient number of items with zero or negative reward and zero consumption of knapsack capacity guarantees feasibility. For the applications and instances that we consider, the practical implications of thismodeling assumption are negligible.

2.1. Modeling competitor behavior

Capturing the nature of the uncertainty in item availability is a critical aspect in the modeling and solution of the KPSC. We hypothesize that each competitor $m \in \mathscr{M}$ considers its reward structure $\{r^m_{kl} : k \in \mathscr{K}, l \in \mathscr{L}_k\}$ and the knapsack constraints (b^m, u^m_k, and ℓ^m_k $\forall k \in \mathscr{K}$) to devise a selection policy, π^m, that prescribes a selection for every possible scenario. We assume that the DM has no information regarding the reward structure of the competitors, but possesses full knowledge of each competitor's knapsack capacity parameters. The DM's lack of information regarding $\{r^m_{kl} : \forall m \in \mathscr{C}, \forall k \in \mathscr{K}, \forall l \in \mathscr{L}_k\}$ induces uncertainty regarding π^m for $m \in \mathscr{C}$. That is, the DM is uncertain about each competitor's item valuation and inferred selection policy.

In our application to professional sports drafts, competing franchises select players that they believe will maximize their chances of winning (or maximize their revenue). The DM typically is aware of its competitors' rosters and therefore knows the needs of its competitors, but does not possess perfect knowledge of the competitors' player valuations. In addition, each franchise's budget tends to be common knowledge to all teams. We note that even if all participants possess perfect knowledge of each others' reward structures, anticipating competitors' selection strategies is a difficult problem when there is more than two teams [4].

A "top-down" approach to modeling this uncertainty relies on the construction of estimates of conditional probability mass functions, $p_{td}(i|h) = P(y_{td} = i | \xi_{td} = h)$, describing the behavior of the competitors. We characterize the evolution of the selection process up to the dth selection during decision epoch t as a product of these conditional mass functions:

$$\begin{aligned} P(\xi_{td} = \{y_{11} = j_1, y_{12} = j_2, \ldots, y_{t,d-1} = j_{t\times(d-1)}\}) \\ = p_{11}(j_1|\{\}) \cdot p_{12}(j_2|\{j_1\}) \cdot p_{13}(j_3|\{j_1, j_2\}) \\ \ldots p_{t,d-1}(j_{t\times(d-1)}|\{j_1, \ldots, j_{t\times(d-2)}\}). \end{aligned}$$

The dependence of the uncertainty in the competitors' future selections on the past selections greatly complicates the decision-making problem. The construction of these probability mass functions quickly becomes an obstacle as the number of possible scenarios over which they are defined combinatorially explodes, requiring increasing amounts of data to reliably estimate these parameters or forcing parameter estimation to rely upon subjective impressions.

As an alternative to explicitly defining probability mass functions in a "top-down" modeling approach, we propose the use of agent-based modeling (ABM) to directly simulate the selection policies of each competitor in a "bottom-up" modeling approach. ABM is an artificial intelligence paradigm that incorporates autonomous agents to simulate complex systems that arise in a variety of applications [29,8]. In particular, for problems in which uncertainty is caused by the actions of various constituents, agent-based simulation provides the advantage of allowing direct imitation of behaviors which may be hard to replicate solely through probability mass functions over the range of aggregate outcomes. In addition, ABM allows the flexible testing of the effect of varying the assumptions of individual agent behavior on the emergent aggregate trends. Admittedly, a disadvantage of ABM is the potential sensitivity to modeling assumptions on individual agent behavior and the difficulty in validating these assumptions by means other than empirical testing.

To implement our agent-based simulation, we assume that the DM possesses a forecast on the *aggregate* behavior of the set of its competitors, $\mathscr{C}$. We assume that this aggregate forecast describes each item's reward with a measure of central tendency, $\bar{r}_{kl}$, and a measure of dispersion, v_{kl}, for a probability distribution. This forecast information, $\mathscr{F} = \{\bar{r}_{kl}, v_{kl} : \forall k \in \mathscr{K}, \forall l \in \mathscr{L}_k\}$, forms the basis for the DM's beliefs on how the selection process will evolve. We utilize $\mathscr{F}$ to generate the DM's estimates of each individual competitor's reward structure, i.e., $\{\hat{r}^m_{kl} : \forall m \in \mathscr{C}, \forall k \in \mathscr{K}, \forall l \in \mathscr{L}_k\}$, that will subsequently factor into the DM's forecasted competitor selections.

To generate the DM's forecasted selec-tions, we model each competitor m as an agent that selects items according to a set of decision criteria that considers the item weights ($\{a_{kl} : \forall k \in \mathscr{K}, \forall l \in \mathscr{L}_k\}$), estimated item valuations ($\{\hat{r}^m_{kl} : \forall k \in \mathscr{K}, \forall l \in \mathscr{L}_k\}$) and the knapsack constraints (b^m, u^m_k, and ℓ^m_k $\forall k \in \mathscr{K}$). In general, the decision criteria are based on rules that reflect the competitors' selection rationale. Agent-based modeling allows the flexibility to tailor decision rules for a particular application, e.g., if the DM is aware of tendencies of a competitor, this knowledge can be incorporated.

Algorithm 1 outlines the implementation of our agent-based simulation to forecast the evolution of the selection process. Algorithm 1 relies on the Select procedure, which contains the application-specific logic for applying decision criteria to simulate the choices of the competitors. In this paper, we consider *greedy* decision rules based on various criterion to simulate competitor behavior; we provide the specific definitions of Select for each of our applications later in this paper.

Algorithm 1. Agent-based simulation of competition.

```
Input: A partial realization of the selection process, $\xi_{td}$, up to the $d$th selection of epoch $t$.
Output: A simulated realization of the entire selection process, $\xi_T$.
Initialization:
 $\xi_T \leftarrow \xi_{td}$.
for $\tau = t, \ldots, T$ do
  for $s = d, \ldots, |\mathscr{R}_\tau|$ do
    if $s = 1$ then
      Call Select to set $y_{\tau s} = i_{x_\tau, l} \in \mathscr{A}(x_\tau, \xi_{\tau s})$ for some $l \in \mathscr{L}_{x_\tau}$.
      $\mathscr{A}(x_\tau, \xi_{\tau, s+1}) \leftarrow \mathscr{A}(x_\tau, \xi_{\tau s}) \backslash \{y_{\tau s}\}$
    else
      Call Select to set $y_{\tau s} = i_{kl} \in \mathscr{A}(k, \xi_{ts})$ for some $k \in \mathscr{K}, l \in \mathscr{L}_k$.
      $\mathscr{A}(k, \xi_{\tau, s+1}) \leftarrow \mathscr{A}(k, \xi_{ts}) \backslash \{y_{\tau s}\}$
    end if
    $\xi_T \leftarrow \xi_T \cup \{y_{\tau s}\}$.
  end for
  Set $d = 1$.
end for
```

Using the logic for simulating competitor behavior to determine the availability of items, we can evaluate the DM's selection policy vector $\mathbf{x}$. To evaluate $\mathbf{x}$, we repeatedly simulate the selection process. In each replication, we execute the DM's selection policy $\mathbf{x} = (x_1, \ldots, x_T)$ by selecting an available item of type x_t at each decision epoch t. Therefore, a critical step is appropriately translating x_t, the item type specified for decision epoch t, into a specific item selection of type x_t, i.e., we need a suitable definition of $g(\xi_{t1}, x_t)$. To accurately evaluate $\mathbf{x}$, ideally $g(\xi_{t1}, x_t)$ would identify the item of type x_t for $t = 1, \ldots, T$, so that collectively $\mathbf{g}(\xi_T(\mathbf{x}))$ maximizes $v^1(\mathbf{g}(\xi_T(\mathbf{x})))$. Determining $g(\xi_{t1}, x_t)$ is difficult, however, as the appropriate item choice depends on the DM's item valuations, the competitors' item valuations, and the competitors' selection policies. In the evaluation of various candidate items of type x_t for $g(\xi_{t1}, x_t)$, we borrow concepts from *beam search* to bound the computational expense of evaluating the effectiveness of $\mathbf{x}$.

Beam search is a heuristic approach that adapts the exact branch-and-bound method by maintaining only the most promising nodes and permanently pruning the rest to cap the computational effort required to search vast enumeration trees. Because its use of a tree structure naturally lends itself to addressing combinatorial problems which consider sequences of decisions, beam search applications include a variety of scheduling problems and sequencing problems in both deterministic [27] and stochastic [11] settings.

Our approach can be viewed as a δ-depth beam search of the enumeration tree of candidate solutions corresponding to $\mathbf{x}$. Beam search pairs a breadth-first search of this enumeration tree with a pruning mechanism governed by a combination of *local* and *global* evaluations of $\mathbf{x}$. A pair of parameters, α and β, called the *filter width* and *beam width*, control the extent that we search the enumeration tree via the appropriately defined local and global evaluations, respectively. Via α and β, we systematically build upon a small number of partial solutions in parallel in search of a good complete solution.

Algorithm 2 outlines the implementation of our beam search to evaluate the value of a DM's selection policy. Algorithm 2 begins by calling the Local procedure to identify the best α candidate items for x_1 with respect to the local evaluation. In general, the local evaluation evaluates candidate items according to a myopic criterion such as item reward, item weight, etc. We provide application-specific definitions of the Local procedure later in the paper when we describe our applications. Each of the α items identified by the Local procedure is assigned to a partial solution, creating α candidate solutions. For each of these α partial solutions, we execute a global evaluation by simulating the remainder of the selection process using Algorithm 1. Based on the completed realizations of these α solutions, we select the best β solutions with respect to $v^1(\cdot)$. Starting with these β solutions for the next iteration, we repeat the process $\delta - 1$ more times, i.e., we probe the enumeration tree to a depth of δ epochs resulting in consideration of $\alpha + (\delta - 1)\alpha\beta$ candidate solutions for $\mathbf{x}$. For example if $\delta = 2$, for the β active solutions after the first iteration, we fix the corresponding realization of $\tilde{\xi}_{21}$ up to the DM's next selection. For each of these β solutions, we then identify α item candidates for x_2. We simulate the remainder of the selection process for each of these $\alpha\beta$ solutions to identify the best β solutions with respect to $v^1(\cdot)$.

Algorithm 2. Beam search for evaluating DM's selection policy.

```
Input: A selection ordering for a pool of competitors, $\mathscr{M}$, and a selection policy, $\mathbf{x}$, for the DM.
Output: An estimate of $v^1(\mathbf{g}(\tilde{\xi}_T(\mathbf{x})))$.
Initialization:
  Initialize $\xi^1_{11} = \{\}$ as an empty realization.
  Let $check = 1$.
for $\tau = 1 \ldots \delta$ do
  if $\tau > 1$ then
    Set $check = \infty$.
  end if
  for $b = 1, \ldots, \min(check, \beta)$ do
    Call Local to identify set of $\alpha$ candidate items $i_{x_\tau l_1}, \ldots, i_{x_\tau l_\alpha} \in \mathscr{A}(x_\tau, \xi^b_{\tau 1})$.
    for $a = 1 \ldots \alpha$ do
      $\xi^{ba}_{\tau 2} \leftarrow \xi^b_{\tau 1} \cup \{i_{x_\tau l_a}\}$.
      Simulate the remainder of $\xi^{ba}_{\tau 2}$ using Algorithm 1to obtain $\xi^{ba}_T$.
      Compute the value of DM's selections in $\xi^{ba}_T$, $v^1(\mathbf{g}(\xi^{ba}_T(\mathbf{x})))$.
    end for
  end for
  Identify the realizations $\xi^1_T, \ldots, \xi^\beta_T$ with $\beta$ largest values of $v^1(\mathbf{g}(\xi^{ba}_T(\mathbf{x})))$ for $b = 1, \ldots, \min(check, \beta)$ and $a = 1, \ldots, \alpha$.
  for $j = 1, \ldots, \beta$ do
    $\xi^j_{\tau+1,1} \leftarrow \xi^j_T \backslash \xi^j_{\tau+1,2}$
  end for
end for
From $\beta$ completed scenarios, set estimate of $v^1(\mathbf{g}(\tilde{\xi}_T(\mathbf{x}))) = \max_{j=1,\ldots,\beta}\{v^1(\mathbf{g}(\xi^j_T(\mathbf{x})))\}$.
```

2.2. *Stochastic ruler approach*

The stochastic ruler method is a stochastic search algorithm that replaces the maximization of the original objective function in (1), and instead seeks to minimize

$$P\{v^1(\mathbf{g}(\tilde{\xi}_T(\mathbf{x}))) \leqslant \Theta(a,b)\},$$

where $\Theta(a,b)$ is a uniform random variable over the range (a,b) and a and b are bounds on the range of observed values for $v^1(\mathbf{g}(\tilde{\xi}_T(\mathbf{x})))$.

To evaluate different selection policies for the DM, we implement the best average estimate variant [2] of the stochastic ruler algorithm. Algorithm 3 describes this procedure in the context of the KPSC. Let $C(\mathbf{x})$ be the number of times that the search algorithm visits $\mathbf{x}$ and let $V(\mathbf{x})$ be the cumulative value of $\mathbf{x}$ summed over all the visits to $\mathbf{x}$.

From the current selection policy at iteration $\imath$, $\mathbf{x}^\imath$, the stochastic ruler algorithm generates a candidate selection policy, $\mathbf{z}$, from $\mathscr{N}(\mathbf{x}^\imath)$, the neighborhood consisting of the set of selection policies obtained by applying a specified set of pertubation operators to $\mathbf{x}^\imath$. In this paper, we randomly generate neighbors of a selection policy $\mathbf{x} = (x_1, x_2, \ldots, x_T)$ from a compound exchange neighborhood consisting of two 1-exchanges. To describe the first 1-exchange, we define

$$\mathscr{U}(\mathbf{x}) = \left\{ k \in \mathscr{K} : u_k^1 - \sum_{t=1}^{T} I_{\{x_t = k\}} > 0 \right\}$$

and

$$\mathscr{O}(\mathbf{x}) = \left\{ k \in \mathscr{K} : \sum_{t=1}^{T} I_{\{x_t = k\}} - \ell_k^1 > 0 \right\}.$$

By definition, $\mathscr{U}(\mathbf{x})$ is the set of item types which are eligible for exchange into the solution policy and $\mathscr{O}(\mathbf{x})$ is the set of item types eligible for exchange out of the solution policy. For a selection policy $\mathbf{x} = (x_1, x_2, \ldots, x_T)$, restricting the first 1-exchange move to take advantage of the slack in constraints (2) and (3) allows the exchange of an item type $j \in \mathscr{U}(\mathbf{x})$ with $x_t \in \mathscr{O}(\mathbf{x})$ for some $1 \leqslant t \leqslant T$, resulting in $\mathbf{x}' = (x_1, \ldots, x_{t-1}, j, x_{t+1}, \ldots x_T)$ that is feasible with respect to the cardinality constraints in (2) and (3). To complete the compound move and obtain the neighbor policy $\mathbf{z}$, we exchange the positions of two elements of $\mathbf{x}'$ with probability q.

Recall that we ensure feasibility with respect to (4) and (5) for all possible realizations of $\tilde{\xi}_T(\mathbf{x})$ by assuming $|\mathscr{L}_k|$ is sufficiently large for all $k \in \mathscr{K}$ so that a competitor m has a full array of item types at every decision epoch; note that we can satisfy this requirement by simply adding items with $r_{kl}^m \leqslant 0$ and $a_{kl}^m = 0$ if necessary. Thus, we can construct an initial selection policy, x^0, that is feasible with respect to all constraints, and maintain this feasibility throughout the local search.

Algorithm 3. Stochastic ruler algorithm.

```
Input: Data for a KPSC instance.
Output: A KPSC selection policy, $\mathbf{x}^\star$.
Initialization:
  Use forecast, $\mathscr{F}$, to generate reward estimates, $\hat{r}_{kl}^m \ \forall m \in \mathscr{M}, k \in \mathscr{K}, l \in \mathscr{L}_k$.
  Specify initial selection policy, $\mathbf{x}^0$, feasible with respect with respect to (4) and (5).
  Let $V(\mathbf{x}^0) = 0$ and $C(\mathbf{x}^0) = 0$. Let $k = 0$ and $\mathbf{x}^\star = \mathbf{x}^0$.
for maxIter iterations do
  Generate a neighbor policy $\mathbf{z} \in N(\mathbf{x}^k)$.
  Let sampleCount = 1.
  while sampleCount $\leqslant M$ do
    Compute an estimate of the value of the selection policy, $\vartheta = v^1(\mathbf{g}(\tilde{\xi}_T(\mathbf{z})))$, via Algorithm 2.
    $V(\mathbf{z}) \leftarrow V(\mathbf{z}) + \vartheta$.
    $C(\mathbf{z}) \leftarrow C(\mathbf{z}) + 1$.
    Generate a uniform random variable $\Theta(a, b)$.
    if $\vartheta < \Theta$ then
      Break while loop. Let $\mathbf{x}^{k+1} = \mathbf{x}^k$.
    end if
    sampleCount $\leftarrow$ sampleCount + 1.
  end while
  if sampleCount = $M + 1$ then
    Accept move to $\mathbf{z}$; let $\mathbf{x}^{k+1} = \mathbf{z}$.
  end if
  $k \leftarrow k + 1$.
end if
Let $\mathbf{x}^\star = \mathrm{argmax}_{\mathbf{x}} V(\mathbf{x})/C(\mathbf{x})$.
```

3. Computational results

To demonstrate the potential of our algorithm, we perform computational experiments on data sets with varying structure. The first data set is based on the sports draft problem in Fry et al. [13]. We benchmark our results on instances of this data set, in which the knapsack capacity constraint is non-binding and all participants value players identically, against the deterministic dynamic programming approximation of Fry et al. [13]. We also consider generic data sets (with binding knapsack capacity constraints) by utilizing the framework of Pisinger [23] to generate instances with two different assumptions on the reward structure. Except for slight differences in the implementation of the Select and Local procedures to account for the knapsack capacity constraints, we implement our algorithm under the same conditions. To minimize the impact of selection order, we assume a serpentine selection order for the instances of all data set types, i.e., the selection order of the competitors is reversed every other epoch. For the beam search of Algorithm 2, we set $\delta = 1$, $\alpha = 2$, and $\beta = 1$. For the stochastic ruler of Algorithm 3, we set $M = 3$, $maxIter = 100 \times T$, and $q = 1.0$.

To analyze the impact of the DM's forecast accuracy, we examine algorithm performance with varying levels of uncertainty in the competitors' valuation of items. We consider four different levels of uncertainty by varying v_{kl} from 0 to 30 in increments of 10. In our testing, we assume that $\mathscr{F}$ provides the mean, $\bar{r}_{kl}$, of a uniform random variable with a range of v_{kl} for all $k \in \mathscr{K}, l \in \mathscr{L}_k$.

As Algorithm 3 mentions, the DM uses the forecast, $\mathscr{F}$, to generate estimates of the competitors' valuations, $\hat{r}_{kl}^m$ for all $m \in \mathscr{M}, k \in \mathscr{K}, l \in \mathscr{L}_k$. For purposes of creating a realization of an actual scenario, we generate the *true* valuations of the competitors, $r_{kl}^m$for all $m \in \mathscr{M}, k \in \mathscr{K}, l \in \mathscr{L}_k$ also by sampling from $\mathscr{F}$. That is, for $v_{kl} > 0$, the DM's estimates, $\{\hat{r}_{kl}^m\}$, will generally vary from the true valuations, $\{r_{kl}^m\}$, but will be drawn from the same uniform distribution.

To obtain our computational results, we implement a rolling horizon approach. We apply Algorithm 3 to determine the item type, $x_1^\star$, that the DM should select at the current epoch. Then, we execute the selection of the specific item, $i_{x_1^\star, l^\star}$ (presumably stored during the algorithm's implementation). Next, we "roll" the problem forward by simulating an "actual" realization of the draft up to the DM's next decision epoch by having each competitor m execute a selection (via an appropriately defined Select procedure) according to its true valuations $\{r_{kl}^m\}$. Then, we re-apply our heuristic approach to obtain the DM's next selection given this observed realization of the draft.

3.1. Sports league draft data set

We first consider KPSC instances explicitly motivated by a sports league draft. A sports draft is a process in which franchises comprising a sports league select, in a pre-determined order, eligible athletes to complement their existing rosters. The draft system is utilized predominantly by North American professional sports leagues to distribute athletes to franchises in the sports of baseball, football, basketball, and hockey. In general, a sports draft is organized into several rounds (corresponding to KPSC decision epochs) with the selection order of each round typically based on each team's performance from the previous season. Barring special circumstances such as trades and compensatory picks, each team makes a single selection in each round.

Recall from Section 2 that we represent the different player positions via the set of different item types, $\mathscr{K}$. Furthermore, the positional needs of each team m's roster is reflected through the lower cardinality and upper cardinality requirements, ℓ_k^m and u_k^m. Prior to the teams actually selecting players in a draft, teams determine values for each player based on the players' skills as measured through physical and mental tests as well as players' past performances in the sport. Teams often value players differently due to inherent subjectivity in scouts' evaluations as well as differences in the emphasis that teams place on particular player traits. In terms of our model, the value assigned to player l of position k by team m is r^m_{kl}.

Using the sports league draft data set from Fry et al. [13], we evaluate our algorithm's performance against their benchmark results. Fry et al. [13] reduce a Markov decision process formulation of a sports draft to a deterministic dynamic program (DP) in order to attain tractability. The authors solve the deterministic DP through a linear program to determine the DM's best draft strategy. While the effects of uncertainty in draft forecasts are briefly considered by the authors, these stochastic effects are not explicitly modeled.

Fry et al. [13] also examine common rules-of-thumb draft strategies. Because of the overwhelming complexities involved in draft-day decision making, sports teams often advocate simple rules-of-thumb to execute draft strategies based on player valuations and the positional needs of their team. As in Fry et al. [13], we consider the following draft strategies:

I: Team drafts player i_{kl} with the highest ordinal draft ranking, o_{kl}.[1]

II: Team drafts player i_{kl} with largest r_{kl} among positions which the lower cardinality requirement has not been satisfied.

III: Team drafts player i_{kl} with largest r_{kl} among positions which its lower cardinality requirement has not been satisfied unless this player's draft ranking is more than 10 places below another player $i_{k'l'}$ at a position k' which the upper cardinality limit has not been met, i.e., unless $o_{kl} - o_{k'l'} > 10$.

IV: Team drafts player i_{kl} with largest r_{kl} among positions which the upper cardinality limit has not been met.

For the implementation of our algorithm on these instances, we use these draft strategies to simulate competitor behavior via the Select procedure as defined in Algorithm 4. We define Select procedure in Algorithm 4 so as to make an equitable comparison to the approach in Fry et al. [13]. This Select procedure executes each competitor's selection by randomly selecting one of the four draft strategies, i.e., a competitor is equally likely to employ any of the four draft strategies to guide its selection, while the DM selects the player at the specified position x_t with the largest reward.

Algorithm 4. Select procedure for simulating competitor selection in Fry et al. [13] instances.

```
Input: A partial realization of the selection process, $\xi_{td}$, up to the dth selection of epoch t.
Output: An item for the dth selection of epoch t.
if $d = 1$ then
    Return item $i_{x_t,l} = \text{argmax}_{l \in \mathscr{L}_{x_t}} \{r_{x_t,l}\}$.
else
    Randomly select a strategy from the set $\{I, II, III, IV\}$.
    Return item $i_{kl}$ according to the selected strategy.
end if
```

For these instances, Algorithm 5 defines the local evaluation procedure utilized by Algorithm 2. At decision epoch t, Algorithm 5 allows the DM to evaluate various items of the specified type x_t by considering both the individual item reward, $r^1_{x_t,l}$, as well as the ordinal draft ranking, $o_{x_t,l}$, which is effectively a projection of how the draft process will evolve. As the "best" choice of an item of type x_t at decision epoch t may not be the item with the largest $r^1_{x_t,l}$ (due to the uncertainty in how competitors will select items in the future), the logic of Algorithm 5 identifies a set of α items that have large reward values and/or are anticipated to be selected soon.

[1] Such rankings may be specific to each team or may be generic for all opponents. As discussed in Fry et al. [13], there are scores of "draft experts" that compile rankings of all players that are eligible for a sports draft, and who then attempt to predict draft selections for each team. Here we use the same draft ranking system as in Fry et al. [13] drawn from one such third-party "draft expert."

Table 1
Upper and lower cardinal requirements and number of available items for each item type $k \in \mathscr{K}$ in the instances from Fry et al. [13].

k	1	2	3	4	5	6
u^m_k	2	4	4	2	2	2
ℓ^m_k	1	2	2	1	1	1
$\mathscr{L}_k$	45	92	115	48	32	32

Algorithm 5. Local procedure for Fry et al. [13] instances.

```
Input: A set of available items, $\mathscr{A}(k, \xi_{t1})$
Output: A set of items $i_{kl_1}, \ldots, i_{kl_\alpha} \in \mathscr{A}(k, \xi_{t1})$
Initialization:
    Let $\mathscr{L}'_k = \{l \in \mathscr{L}_k : i_{kl} \in \mathscr{A}(k, \xi_{t1})\}$.
    Let $u' = u^1_k - \sum_{s=1}^{t-1} I_{\{x_s = k\}}$.
for $a = 1 \ldots \alpha$ do
    if $(a \bmod 2) \neq 0$ or $u' \leqslant 1$ then
        $i_{kl_a} \leftarrow i_{kl'}$ such that $l' = \text{argmax}_{l \in \mathscr{L}'_k} \{r^1_{kl}\}$.
    else
        Let $\mathscr{L}''_k = \{l \in \mathscr{L}'_k : i_{kl}$ corresponds to an item with one of the $u'$ largest $r_{kl}$ values $\}$
        $i_{kl_a} \leftarrow i_{kl'}$ such that $l' = \text{argmax}_{l \in \mathscr{L}''_k} \{o_{kl}\}$.
    end if
    $\mathscr{L}'_k \leftarrow \mathscr{L}'_k \backslash i_{kl_a}$.
    $u' \leftarrow u' - 1$.
end for
```

For the sports league instances, we set $|\mathscr{M}| = 10$, $|\mathscr{K}| = 6$, $T = 16$, $b^m = 16\ \forall m \in \mathscr{M}$, and $a_{kl} = 1\ \forall k \in \mathscr{K}, \forall l \in \mathscr{L}_k$. Table 1 supplies the upper and lower cardinality requirements for each position. For each player selected to fulfill a lower cardinality requirement of position k, the contribution to the DM's overall value function is $1 \times r^1_{kl}$ while each player selected beyond the lower cardinality requirement for position k contributes $0.6 \times r^1_{kl}$. This valuation serves to differentiate between players that are expected to be major contributors to the team in terms of playing time versus those that will serve as backups or limited-use players.

Fry et al. [13] test their deterministic dynamic programming approximation approach using varying levels of information regarding the DM's knowledge of its competitor's selection strategies. In the *naïve* case, the authors assume that the DM has no knowledge of its competitor's selection strategies and simply assumes that they will select according to strategy III. In the *imperfect* information case, Fry et al. [13] assume that the DM does not know the strategy of individual competitors, but possesses some knowledge of the distribution of strategies implemented by the competitors in aggregate. In the *perfect* information case, they assume that the DM knows the selection strategy of each individual competitor. As Table 2 presents, the approach of Fry et al. [13] is likely to outperform the myopic draft strategies, especially as the accuracy of information about the competitor's draft strategies improves. We note that in these five instances, we assume that the forecast, $\mathscr{F} = \{\bar{r}_{kl}, v_{kl} : \forall k \in \mathscr{K}, \forall l \in \mathscr{L}_k\}$, is defined such that $\bar{r}_{kl} = r^1_{kl}$ and $v_{kl} = 0 \forall k \in \mathscr{K}, \forall l \in \mathscr{L}_k$. That is, each team values the players the same as the DM, and the DM is aware of this.

Table 2 compares the results from Fry et al. [13] to the results from our stochastic ruler algorithm under varying amounts of information availability. Note that results for our stochastic ruler algorithm display average solution values and standard deviations over 10 runs.

Table 2
The stochastic ruler offers improvement over alternative heuristics on the five instances from Fry et al. [13].

DM selection approach	Instance				
	1	2	3	4	5
I	2108.5	2112.6	2165.9	2075.3	2132.4
II	2135.9	2135.9	2175.1	2129.3	2158.7
III	2129.8	2124.4	2172.5	2119.4	2146.2
IV	2027.1	2046.8	2068.9	2027.9	2009.5
Fry et al. [13]—naïve	2148.2	2164.5	2170.4	2158.0	2157.6
Fry et al. [13]—imperfect	2173.4	2184.8	2176.7	2162.0	2171.7
Stochastic ruler	2175.9 (4.7)	2189.5 (4.5)	2210.3 (8.7)	2189 (7.2)	2185.4 (5.0)
Fry et al. [13]—perfect	2187.4	2195.9	2240.5	2184.5	2192.1
stochastic ruler—perfect	2252.2 (29.4)	2275 (2.3)	2280.6 (7.6)	2289.0 (3.9)	2285.1 (7.7)
SR % improve. Fry et al. [13]–naïve %	1.3	1.1	1.8	1.4	1.3
SR % improve. Fry et al. [13]–imperfect %	0.1	0.2	1.5	1.2	0.6
SR—perfect % improve. Fry et al. [13]—perfect %	3.0	3.6	1.8	4.8	4.2

Comparing the performance of the stochastic ruler to the naïve version of Fry et al.'s deterministic approach, we observe the benefit of using agent-based modeling to simulate competitor behavior versus simply assuming all competitors follow a single fixed strategy. In the imperfect case, the stochastic ruler algorithm achieves an average solution value that exceeds the value from the deterministic approach of Fry et al. [13]. Comparing the stochastic ruler algorithm to the deterministic approach of [13] when both algorithms have perfect information regarding the competitors' selection strategies shows that the stochastic ruler algorithm does a better job of taking advantage of the improved information. All differences are significant at the 99% confidence level except for the difference between the stochastic ruler and Fry et al. [13]-imperfect on instance I, which is significant at the 90% confidence level. The stochastic ruler algorithm also generates solutions quite quickly for this sports draft problem. The time required to make the first selection never exceeds 58 CPU s, and the time to make all T selections averages 2.7 CPU min.

3.2. Generalized data sets

To demonstrate the flexibility of our approach and specifically the agent-based modeling component, we consider a pair of generalized data sets. These data sets ostensibly represent instances of sports drafts in which a budgetary constraint is represented by the knapsack capacity. We utilize the framework of Pisinger [23] to create data sets with two different reward structures. In the generalized data sets, the DM does not have perfect information regarding its competitors' reward structures (unlike the data set from Fry et al. [13] in which all participants share the same valuations).

To model the selections of the competitors in the presence of a binding knapsack capacity constraint, we consider a set of decision rules based on the criterion from various greedy algorithms for the multidimensional 0–1 knapsack problem [12]. The first step in simulating y_{td}, the selection of competitor m at the dth selection in epoch t, is to identify a subset of item types, $\mathscr{K}^m_{td} \subseteq \mathscr{K}$, for consideration. We hypothesize that the criterion for identifying appealing item types for item selection may consider the cardinality constraints on these types. We introduce a random element in the construction of $\mathscr{K}^m_{td} \subseteq \mathscr{K}$ by defining two binary random variables, Z_ℓ and Z_m, that shape the rationale of the competitor's selection. If $Z_\ell = 1$, the competitor focuses on item types for which the lower bound requirement has not been satisfied; otherwise, the competitor considers all item types for which the upper bound limit has not been met. If the competitor indeed focuses on item types for which the lower bound requirement has not been satisfied, the indicator $Z_m \in [0, Z_\ell]$ further refines whether or not item selection should consider the *number* of items of type k that still need to be selected. Competitor m chooses item $i_{kl} \in \mathscr{A}(k, \xi_{td})$ for $k \in \mathscr{K}^m_{td}$ in the dth selection of epoch t that maximizes its utility function, ϕ^m_{kl}. In Appendix A, we provide detailed specifications for $\mathscr{K}^m_{td}$ and ϕ^m_{kl} as well as the definitions of the Select and Local procedures that utilize them.

3.2.1. Uniform data set

Utilizing the structure of the *weakly correlated* instances in Pisinger [23], we consider a set of uniformly generated KPSC data sets. We randomly generate integer reward values, $\bar{r}_{kl}$, from the interval $[1, 100]$, for all $k \in \mathscr{K}$, $l \in \mathscr{L}_k$. We randomly generate the corresponding capacity consumption, a_{kl}, from the interval $[\bar{r}_{kl} - 10, \bar{r}_{kl} + 10]$, for all $k \in \mathscr{K}$, $l \in \mathscr{L}_k$. We assume $|\mathscr{K}| = 10$, $|\mathscr{L}_k| = 100$ for all $k \in \mathscr{K}$, and $|\mathscr{M}| = 30$. Integer cardinality bounds are randomly generated such that $u^m_k \in [1, 5]$ and $\ell^m_k \in [1, u^m_k]$ for all $m \in \mathscr{M}, k \in \mathscr{K}$. To calibrate the length of the selection process with the number of available items, we set $T = \max_{m \in \mathscr{M}}\{\sum_{k \in \mathscr{K}} \ell^m_k\}$. We also set $b^m = \lfloor \bar{r} T \rfloor$ for all $m \in \mathscr{M}$, where $\bar{r}$ is the average $\bar{r}_{kl}$ value over $k \in \mathscr{K}, l \in \mathscr{L}_k$.

For each of the four levels of forecast uncertainty, we randomly generate five instances corresponding to different realizations of competitors' true valuations, r^m_{kl}, and the strategies that they implement at each selection. Note that the instances with no variability, $v_{kl} = 0$ for all $k \in \mathscr{K}$, $l \in \mathscr{L}_k$, still have a random element because the selection strategy implemented by a competitor at each selection is determined by randomly generating $Z_\ell \in [0, 1]$ and $Z_m \in [0, Z_\ell]$.

Randomly selecting one of the participants to serve as the DM, we implement the stochastic ruler algorithm five times on each instance and present the average and standard deviation of these five runs. To benchmark the performance, we implement four combinations of the greedy heuristics defined by the criteria given by (6) and (7) in Appendix A. The first three greedy approaches fix the values Z_ℓ and Z_m in various combinations. The fourth greedy approach assumes that Z_ℓ and Z_m are randomly generated in the respective ranges, i.e., the DM determines a selection strategy in the same manner as the competitors.

Tables 3–6 demonstrate the performance of our algorithm at varying levels of forecast uncertainty. For each instance, we denote the best average solution in boldface font. The average solution obtained by the stochastic ruler approaches dominates the alternatives for all instances. The column titled *% Improve* provides the percent improvement provided by the stochastic ruler over the next best alternative. In addition, the stochastic ruler approach is consistent as reflected by the relatively low σ_p value, which is the pooled estimate of variability [21] over the entire set of 25 runs at each level of forecast accuracy. Furthermore, Tables 3–6 show that the stochastic ruler approach is increasingly more effective relative to the greedy approaches as forecast accuracy wanes.

Table 3
Five instances of uniformly generated data set with $v_{kl} = 0\ \forall k \in \mathscr{K}, l \in \mathscr{L}_k$.

Random-I instance	Stochastic ruler		(Z_ℓ, Z_m)			Randomized		Improve
	μ	σ	(1,1)	(1,0)	(0,0)	μ	σ	
1	**1061.6**	7.2	1030.0	1039.0	1039.0	1041.4	5.5	2.1
2	**1058.8**	8.6	1040.0	1040.0	1040.0	1041.4	2.7	1.8
3	**1062.2**	5.6	1038.0	1051.0	1051.0	1054.0	3.7	1.1
4	**1057.4**	4.8	1046.0	1038.0	1038.0	1035.2	4.1	1.8
5	**1053.2**	6.8	1047.0	1050.0	1050.0	1044.4	7.3	0.3
$\bar{\mu}$	**1058.6**		1040.2	1043.6	1043.6	1043.3		1.4
σ_p	6.7		6.9	6.3	6.3	4.9		

Table 4
Five instances of uniformly generated data set with $v_{kl} = 10\ \forall k \in \mathscr{K}, l \in \mathscr{L}_k$.

Random-II instance	Stochastic ruler		(Z_ℓ, Z_m)			Randomized		% Improve
	μ	σ	(1,1)	(1,0)	(0,0)	μ	σ	
1	**1079.2**	3.8	1049.0	1042.0	1042.0	1034.4	9.8	2.8
2	**1078.4**	15.0	1024.0	1023.0	1023.0	1023.2	1.6	5.0
3	**1071.8**	8.8	1024.0	1048.0	1048.0	1046.4	8.8	2.2
4	**1083.0**	6.5	1043.0	1040.0	1040.0	1039.0	11.0	3.7
5	**1080.2**	10.3	1030.0	1057.0	1057.0	1044.8	10.9	2.1
$\bar{\mu}$	**1078.5**		1034.0	1042.0	1042.0	1037.6		3.4
σ_p	9.6		11.4	12.5	12.5	9.1		

Table 5
Five instances of uniformly generated data set with $v_{kl} = 20\ \forall k \in \mathscr{K}, l \in \mathscr{L}_k$.

Random-III instance	Stochastic ruler		(Z_ℓ, Z_m)			Randomized		% Improve
	μ	σ	(1,1)	(1,0)	(0,0)	μ	σ	
1	**1097.2**	13.4	1002.0	1021.0	1021.0	1026.2	8.7	6.9
2	**1107.6**	5.5	1049.0	1039.0	1039.0	1034.4	12.9	5.3
3	**1118.2**	6.1	1013.0	1020.0	1020.0	1023.4	12.6	8.8
4	**1108.4**	8.1	1025.0	1062.0	1062.0	1070.4	12.3	4.2
5	**1114.8**	6.6	1029.0	1041.0	1041.0	1030.4	9.3	6.6
$\bar{\mu}$	**1109.2**		1023.6	1036.6	1036.6	1037.0		6.5
σ_p	8.4		17.7	17.2	17.2	11.3		

Table 6
Five instances of uniformly generated data set with $v_{kl} = 30\ \forall k \in \mathscr{K}, l \in \mathscr{L}_k$.

Random-IV instance	Stochastic ruler		(Z_ℓ, Z_m)			Randomized		% Improve
	μ	σ	(1,1)	(1,0)	(0,0)	μ	σ	
1	**1132.6**	6.3	1013.0	1043.0	1043.0	1025.2	10.0	7.9
2	**1131.4**	10.3	1040.0	1051.0	1051.0	1061.0	20.3	8.1
3	**1124.2**	23.2	1051.0	1038.0	1038.0	1033.2	8.6	7.7
4	**1124.8**	11.5	1048.0	1056.0	1056.0	1049.2	11.9	6.1
5	**1127.0**	7.5	1034.0	1057.0	1057.0	1028.8	10.1	6.2
$\bar{\mu}$	**1128.0**		1037.2	1049.0	1049.0	1039.5		7.0
σ_p	13.2		15.1	8.3	8.3	12.9		

As previously mentioned, the solution values presented in Tables 3–6 correspond to solutions obtained in a rolling horizon fashion. The time to determine the DM's first selection takes the longest as the stochastic ruler must search over a solution policy vector of full length T. However, the time to make the first selection never exceeds 10 CPU min for the uniformly generated data set, and the time to make all T selections averages 23.8 CPU min. Thus, over 40 percent of the computation time is spent determining the first decision. The amount of time to determine subsequent decisions decreases quickly as the length of the solution policy decreases.

3.2.2. *Heterogeneous item type data set*

In the second set of instances, we consider $|\mathscr{K}| = 6$, $|\mathscr{M}| = 10$, and $T = 16$. In contrast to the data set in Section 3.2.1, we vary, by item type, the distribution from whence we generate $\bar{r}_{kl}$ for all $k \in \mathscr{K}$, $l \in \mathscr{L}_k$. Table 7 describes the beta distribution parameters from whence we generate the $\bar{r}_{kl}$ values for the respective $k \in \mathscr{K}$. Upon generating the $\bar{r}_{kl}$ values, we set $a_{kl} = (\bar{r}_{kl}/10) + 5 + \varepsilon$, where $\varepsilon \in [-5, 5]$. We set $b^m = 300$ for all $m \in \mathscr{M}$. Table 8 contains the upper and lower cardinality requirements as well as the number of available items of each type $k \in \mathscr{K}$.

For each of the four levels of forecast uncertainty, we randomly generate 10 instances corresponding to different realizations of competitors' true valuations, r_{kl}^{m}, and the strategies that they implement at each selection. As in our first set of instances, we randomly select one of the participants to serve as the DM, and we implement the stochastic ruler algorithm five times on each instance. We benchmark the performance by comparing the stochastic ruler solution values to the same set of greedy heuristics used in Section 3.2.1.

Table 7
Distribution for generating $\bar{r}_{kl}$ for heterogeneous item type data set.

k	Beta distribution parameters			
	Shape 1	Shape 2	Lower bound	Upper bound
1	0.95	1.3	10	360
2	0.65	2.0	20	350
3	1.10	1.4	10	220
4	1.40	2.6	0	180
5	1.10	1.4	75	160
6	1.2	2.0	75	165

Table 8
Upper and lower cardinal requirements and number of available items for each item type $k \in \mathscr{K}$ in the heterogeneous item type data set.

k	1	2	3	4	5	6
u_k^m	4	6	8	3	3	3
ℓ_k^m	1	2	2	1	1	1
$\mathscr{L}_k$	50	100	120	50	40	40

Tables 9–12 demonstrate the performance of our algorithm on this data set at the varying levels of forecast uncertainty. For each instance, we denote the best average solution in boldface font. In Table 9, where the DM has no uncertainty regarding the competitors' item valuations, the stochastic ruler's performance is very close to that of the greedy strategies characterized by $(Z_\ell, Z_m) = (1,1)$ or $(1,0)$. However, even in this case, the stochastic ruler solution averaged over all 10 instances is superior to the next best alternative. In addition, the stochastic ruler approach consistently finds good solutions, as evidenced by the relatively low σ_p value. The combination of good average solutions and consistent performance across instances suggests that the stochastic ruler is much more robust than the greedy alternatives. Similar to the experiments on the uniformly generated data set, Tables 9–12 show that the stochastic ruler approach is increasingly more effective relative to the greedy approaches as forecast accuracy decreases. We note that the percentage improvement is not as large as in the uniformly generated data set as the variability introduced (as a percentage of the $\bar{r}_{kl}$ values) is smaller in the heterogeneous data set. Noting that the average item reward in the heterogeneous item type data set is about three times as large as the average item reward in the uniformly generated data set, the results for *Random-II* and *Hetero-IV* are most comparable in terms of percent variability. The similarity in the stochastic ruler's overall percent improvement in these two cases (Tables 4 and 12) suggest the robustness of the stochastic ruler algorithm.

As with our previous results, the solution values presented in Tables 9–12 are obtained in a rolling horizon fashion. The time required to make the first selection never exceeds 54 CPU s for the heterogeneous item type data set, and the time to make all T selections averages 3.4 CPU min.

Table 9
Heterogeneous item type data set with $v_{kl} = 0\ \forall k \in \mathscr{K}, l \in \mathscr{L}_k$.

Hetero-I instance	Stochastic ruler		(Z_ℓ, Z_m)			Randomized		% Improve
	μ	σ	(1,1)	(1,0)	(0,0)	μ	σ	
1	**1874.88**	16.85	1851.30	1851.30	1638.10	1839.62	66.34	1.3
2	1834.40	20.80	**1853.00**	**1853.00**	1598.90	1833.18	25.81	−1.0
3	1872.84	20.35	1862.60	1847.50	1653.30	**1875.80**	38.71	−0.2
4	1845.00	20.18	**1847.60**	**1847.60**	1606.10	**1847.60**	52.82	−0.1
5	**1851.28**	26.74	1845.60	1845.60	1595.90	1785.56	65.70	0.3
6	**1882.64**	13.36	1867.00	1867.00	1680.10	1859.20	7.74	0.8
7	**1873.12**	5.51	1842.80	1842.80	1671.50	1859.92	15.05	1.6
8	**1851.88**	24.62	1808.50	1808.50	1601.20	1746.56	19.96	2.4
9	**1882.44**	5.15	1866.60	1866.60	1678.90	1856.44	14.83	0.8
10	1877.48	30.15	**1878.40**	**1878.40**	1607.30	1834.60	52.82	−0.0
$\bar{\mu}$	**1864.60**		1852.34	1850.83	1633.13	1833.85		0.7
σ_p	18.06		19.11	18.80	35.25	39.11		

Table 10
Heterogeneous item type data set with $v_{kl} = 10\ \forall k \in \mathscr{K}, l \in \mathscr{L}_k$.

Hetero-II instance	Stochastic ruler		(Z_ℓ, Z_m)			Randomized		% Improve
	μ	σ	(1,1)	(1,0)	(0,0)	μ	σ	
1	**1894.88**	15.63	1862.30	1862.30	1685.30	1780.86	50.18	1.7
2	**1873.84**	7.84	1836.20	1836.20	1637.90	1822.56	38.01	2.0
3	**1914.24**	10.44	1839.00	1872.70	1645.60	1854.74	37.90	2.2
4	1936.44	7.87	**1947.00**	**1947.00**	1702.30	1881.20	18.56	−0.5
5	1860.96	5.96	**1863.80**	**1863.80**	1623.00	1825.58	33.04	−0.2
6	**1873.16**	18.40	1866.80	1866.80	1600.70	1782.68	63.91	0.3
7	**1892.12**	27.69	1844.20	1891.80	1700.10	1838.24	31.34	0.0
8	**1903.24**	7.83	1882.20	1875.60	1658.50	1868.88	15.60	0.6
9	1891.68	12.82	**1922.20**	**1922.20**	1676.80	1909.70	13.44	−1.6
10	**1886.24**	22.58	1833.10	1833.30	1596.90	1804.08	36.72	2.9
$\bar{\mu}$	**1892.68**		1869.68	1877.17	1652.71	1836.85		0.8
σ_p	15.34		38.06	35.36	38.52	36.98		

Table 11
Heterogeneous item type data set with $v_{kl}=20\ \forall k\in\mathcal{K}, l\in\mathcal{L}_k$.

Hetero-III instance	Stochastic ruler		(Z_ℓ, Z_m)			Randomized		% Improve
	μ	σ	(1,1)	(1,0)	(0,0)	μ	σ	
1	**1925.36**	14.14	1844.40	1844.40	1661.30	1786.08	39.35	4.4
2	**1915.56**	8.59	1844.00	1889.60	1644.00	1754.50	95.39	1.4
3	**1933.52**	20.26	1861.40	1911.60	1665.70	1825.44	51.57	1.1
4	**1865.56**	14.82	1794.80	1794.80	1668.60	1818.08	14.37	3.9
5	**1904.72**	24.86	1896.00	1896.00	1633.30	1863.18	22.21	0.5
6	**1903.44**	11.36	1871.00	1873.40	1663.10	1830.94	32.39	1.6
7	**1952.12**	9.23	1901.40	1901.40	1664.10	1837.16	57.20	2.7
8	**1905.24**	15.29	1869.00	1869.00	1658.70	1855.28	32.67	1.9
9	**1921.20**	17.52	1864.90	1864.90	1654.40	1812.48	89.46	3.0
10	**1907.98**	28.00	1803.10	1803.10	1618.10	1787.44	45.38	5.8
$\bar{\mu}$	**1913.47**		1855	1864.82	1653.13	1817.06		2.6
$\sigma_{p\cdot}$	17.50		34.92	39.91	16.40	54.27		

Table 12
Heterogeneous item type data set with $v_{kl}=30\ \forall k\in\mathcal{K}, l\in\mathcal{L}_k$.

Hetero-IV instance	Stochastic ruler		(Z_ℓ, Z_m)			Randomized		% Improve
	μ	σ	(1,1)	(1,0)	(0,0)	μ	σ	
1	**1969.88**	14.10	1794.40	1794.40	1644.80	1792.84	7.73	9.8
2	**1930.80**	17.46	1897.40	1897.40	1661.30	1820.78	77.71	1.8
3	**1925.92**	4.41	1851.90	1851.90	1597.40	1742.94	63.56	4.0
4	**1893.36**	30.02	1787.30	1807.90	1659.70	1766.16	70.96	4.7
5	**1941.96**	16.62	1852.40	1852.40	1628.90	1847.24	71.14	4.8
6	**1973.68**	14.80	1919.60	1919.60	1678.40	1899.76	35.36	2.8
7	**1891.40**	6.05	1806.80	1806.80	1567.50	1804.74	34.55	4.7
8	**1933.20**	2.58	1846.50	1846.50	1636.00	1775.82	70.21	4.7
9	**1915.32**	14.72	1841.80	1841.80	1646.80	1784.56	75.55	4.0
10	**1942.60**	16.43	1912.60	1912.60	1653.90	1892.44	16.59	1.6
$\bar{\mu}$	**1927.86**		1853.8	1861.38	1639.23	1819.89		3.6
$\sigma_{p\cdot}$	19.84		47.95	51.27	30.90	59.20		

4. Conclusions and future work

We present a hybrid metaheuristic for addressing a multi-period stochastic knapsack problem in which the availability of items in future epochs is uncertain due to the presence of competitors. Representing a solution as a vector identifying the *item type* to select at each upcoming decision epoch (rather than a specific item) reduces the dimension of the solution space that is computationally intractable via a direct, naïve state space representation [13]. This solution representation facilitates the development of an efficient search heuristic. For a given item-type solution vector, we utilize beam search principles to evaluate item candidates of the specified types. The beam search in turn relies upon an agent-based model that incorporates application-specific decision rules to simulate the actions of competitors, which are unknown to the decision-maker. Using the solution evaluations provided by the beam search algorithm, a stochastic ruler algorithm governs the local search process which generates neighbor solutions with a compound exchange neighborhood. We demonstrate the effectiveness of our approach on instances of three different types of data sets. Our approach compares favorably to greedy heuristics, particularly as the decision-maker's knowledge of the competitors' item valuations deteriorates.

To the knowledge of the authors, this is the first treatment of a multi-period bounded multiple-choice knapsack problem in which there is uncertainty of item availability induced by competitors. The problem formulation and our proposed solution approach are flexible enough to be applied to a variety of resource allocation problems under sequence-based competition. Future work includes exploring the use of our solution framework on related problems by considering different objective function and constraint formulations. In addition, we believe that there is potential for adaptive learning techniques to be integrated into the solution approach, particularly for the decision-maker to better anticipate the actions of the competitors.

Appendix A.

In this appendix, we describe the constructs necessary for definition of the SELECT and LOCAL procedures for the generalized data sets of Section 3.2. Specifically, we outline the method for determining $\mathcal{K}^m_{td}$, the subset of item types considered by competitor m at the dth selection of epoch t, and we define the functional form of the opponents' utility functions, ϕ^m_{kl}.

Let N^m_{td} be the number of selection opportunities for competitor m from the dth selection in epoch t to the end of the entire selection process. Let $\mathbf{i}^m_{td}$ be the vector of items selected by competitor m up to the dth selection in epoch t. Define $c_k(\mathbf{i}^m_{td})$ as a function providing the number of items of type k that competitor m has selected up to the dth selection in epoch t. While various functional forms are possible, we compose the subset $\mathcal{K}^m_{td}$ as the set of item types $k'\in\mathcal{K}$ such that

$$k' = \underset{k\in\mathcal{K}}{\operatorname{argmax}}\left\{\left(Z_\ell \vee I_{\left\{N^m_{td}=\sum_{j\in\mathcal{K}}(\ell^m_j - c_j(\mathbf{i}^m_{td}))\right\}}\right) I_{\{\ell^m_k > c_k(\mathbf{i}^m_{td})\}} + Z_m(\ell^m_k - c_k(\mathbf{i}^m_{td}))^+ - I_{\{c_k(\mathbf{i}^m_{td})=u^m_k\}}\right\}. \quad (6)$$

The first summand of (6) will be one if the competitor has not selected the minimum number of type k items and it has been determined that the competitor will select an item type for which the lower bound has not been met (either because $Z_\ell=1$ or because the competitor must select such an item type in order to achieve feasibility with respect to the lower cardinality requirement). The second summand will be positive only if the competitor has not selected the minimum number of type k items and it has been determined that the competitor is focusing on the degree to which the lower bound has not been achieved. The third summand will be one for item type k if the competitor has reached the upper bound on the number of items of type k.

Before stating the criterion for determining competitor m's selection from $\mathscr{K}_{td}^m$, we define some additional parameters. Let $b^m(\mathbf{i}_{td}^m)$ be the remaining capacity of competitor m's knapsack at the dth selection in epoch t. Let S_{td}^m be the number of selections until competitor m's next selection after its current selection, y_{td}. We estimate competitor m's opportunity cost of an item selection through the construction of $\mathscr{G}(k,\xi_{td})$, the set of items of type k estimated to be selected before competitor m's next selection (and therefore unavailable to competitor m at future epochs). The method for constructing $\mathscr{G}(k,\xi_{td})\ \forall k\in\mathscr{K}$ must be simple and easily computed as this procedure is implemented to forecast each competitor's selection. We construct $\mathscr{G}(k,\xi_{td})\ \forall k\in\mathscr{K}$ by considering items to be selected before competitor m's next selection in decreasing order of $\hat{r}_{kl}^m$. As $\hat{r}_{kl}^m$ is static, this list of values only has to be sorted once in a pre-processing step before the selection process begins. Furthermore, an item is only considered a candidate for potential selection if it would fit into competitor m's knapsack. We also assume that competitors will first consider their lower cardinality bounds. After these requirements are fulfilled, the competitors will then consider items with large reward values at positions which have not met their upper cardinality bounds. Algorithm 6 outlines this procedure.

```
Algorithm 6. Heuristic for constructing $\mathscr{G}(k,\xi_{td})\forall k\in\mathscr{K}$.
Input: A partial realization of the selection process, $\xi_{td}$, up to the $d$th selection of epoch $t$.
Output: $\mathscr{G}(k,\xi_{td})\ \forall k\in\mathscr{K}$, collectively representing the items expected to be selected before competitor $m$'s next selection.
Initialization:
  Let $\mathscr{G}(k,\xi_{td})=\emptyset\ \forall k\in\mathscr{K}$.
  Let $L=\{i_{k_1l_1}, i_{k_2l_2},\ldots\}\ \forall i_{k_jl_j}\in\mathscr{A}(k_j,\xi_{td})$ for some $k_j\in\mathscr{K}$, ordered such that $\hat{r}_{k_jl_j}^m\geqslant\hat{r}_{k_{j+1}l_{j+1}}^m$.
  Let $j=1$.
while $\sum_{k\in\mathscr{K}}|\mathscr{G}(k,\xi_{td})|<S_{td}^m$ do
  if $a_{k_jl_j}\leqslant b^m(\mathbf{i}_{td}^m)$ then
    if $|\mathscr{G}(k_j,\xi_{td})|<\sum_{p\in\mathscr{M},p\neq m}[\ell_{k_j}^p-c_{k_j}(i_{td}^p)]$ then
      $\mathscr{G}(k_j,\xi_{td})\leftarrow\mathscr{G}(k_j,\xi_{td})\cup\{i_{k_jl_j}\}$.
    else
      if $\sum_{k\in\mathscr{K}}|\mathscr{G}(k,\xi_{td})|+\sum_{k\in\mathscr{K}}\sum_{p\in\mathscr{M},p\neq m}[\ell_k^p-c_k(i_{td}^p)]<S_{td}^m$ and $\sum_{p\in\mathscr{M},p\neq m}[u_k^p-c_k(i_{td}^p)]>0$ then
        $\mathscr{G}(k_j,\xi_{td})\leftarrow\mathscr{G}(k_j,\xi_{td})\cup\{i_{k_jl_j}\}$.
      end if
    end if
  end if
  $j\leftarrow j+1$.
end while
```

After constructing $\mathscr{K}_{td}^m$ to limit the item types considered, we determine the specific item selection of competitor m at the dth selection in epoch t by modifying the greedy knapsack heuristics of Dobson [10] and Rinnooy Kan et al. [25] to account for the uncertain and dynamic nature of the KPSC. We assume that competitor m selects item $i_{kl}\in\mathscr{A}(k,\xi_{td})$ for $k\in\mathscr{K}_{td}^m$ that maximizes ϕ_{kl}^m, where

$$\phi_{kl}^m=I_{\{a_{kl}\leqslant b^m(\mathbf{i}_{td}^m)\}}\left[\frac{\hat{r}_{kl}^m-\max\limits_{n\in\mathscr{A}(k,\xi_{td})\setminus\mathscr{G}(k,\xi_{td})}\hat{r}_{kn}^m}{(a_{kl}/b^m(\mathbf{i}_{td}^m))+(1/N_{td}^m)+(1/(u_k^m-c_k(\mathbf{i}_{td}^m)))}\right]. \quad (7)$$

In (7), $I_{\{a_{kl}\leqslant b^m(\mathbf{i}_{td}^m)\}}$ prevents the selection of an item which does not obey the knapsack capacity constraint. The numerator of the fraction in (7) calculates the opportunity cost of bypassing the selection of item i_{kl}. The denominator of the fraction in (7) is a weighted average of (i) the percentage of remaining item capacity that item i_{kl} would consume, (ii) the inverse ratio of the remaining number of selections, and (iii) the inverse ratio of the remaining number of type k items that the competitor can still feasibly select. Algorithm 7 summarizes the Select procedure which we utilize to simulate competitors' selections.

```
Algorithm 7. Select procedure for simulating competitor selection.
Input: A partial realization of the selection process, $\xi_{td}$, up to the $d$th selection of epoch $t$.
Output: An item for the $d$th selection of epoch $t$.
Initialization:
  Randomly generate $Z_\ell\in[0,1]$ and $Z_m\in[0,Z_\ell]$.
Let $\mathscr{K}_{td}^m=\{k'\in\mathscr{K}: k'$ satisfies (6) $\}$.
Return $i_{kl}\in\mathscr{A}(k,\xi_{td})$ for $k\in\mathscr{K}_{td}^m$ that maximizes $\phi_{kl}$ as defined by (7).
```

To assist the beam search in Algorithm 2 evaluate alternatives for DM's selection of an item of type x_t at decision epoch t, we outline the Local procedure with Algorithm 8. As the "best" choice of an item of type x_t at decision epoch t may not be the item with the largest $r_{x_t,l}^1$, Algorithm 8 allows the consideration of items (that can be feasibly inserted into the knapsack) by considering both the individual item reward and the marginal item reward per unit consumption of remaining knapsack capacity.

```
Algorithm 8. Local procedure for generalized data sets.
Input: A set of available items, $\mathscr{A}(k,\xi_{t1})$
Output: A set of items $i_{kl_1},\ldots,i_{kl_\alpha}\in\mathscr{A}(k,\xi_{t1})$
Initialization:
  Let $\mathscr{L}'_k=\{l\in\mathscr{L}_k:\forall i_{kl}\in\mathscr{A}(k,\xi_{t1})\}$.
  Let $b_t^1=b^1-\sum_{s=1}^{t-1}a_{x_s,h(g(\xi_{s1},x_s))}$ be the remaining capacity of DM's knapsack.
for $a=1\ldots\alpha$ do
  if $(a \bmod 2)\neq 0$ then
    $i_{kl_a}\leftarrow i_{kl'}$ such that $a_{kl'}\leqslant b_t^1$and $l'=\operatorname{argmax}_{l\in\mathscr{L}'_k}\{r_{kl}^1\}$.
  else
    $i_{kl_a}\leftarrow i_{kl'}$ such that $a_{kl'}\leqslant b_t^1$ and $l'=\operatorname{argmax}_{l\in\mathscr{L}'_k}\{r_{kl}^1/(a_{kl}/b_t^1)\}$.
  end if
  $\mathscr{L}'_k\leftarrow\mathscr{L}'_k\setminus i_{kl_a}$.
end for
```

References

[1] Alrefaei MH, Andradottir S. A modification of the stochastic ruler method for discrete stochastic optimization. European Journal of Operational Research 2001;133:160–82.

[2] Alrefaei MH, Andradottir S. Discrete stochastic optimization using variants of the stochastic ruler method. Naval Research Logistics 2005;52(4):344–60.

[3] Axelrod R. The complexity of cooperation: agent-based models of competition and collaboration. Princeton, NJ: Princeton University Press; 1997.

[4] Brams SJ, Straffin PD. Prisoners' dilemma and professional sports drafts. The American Mathematical Monthly 1979;86(2):80–8.

[5] Brams SJ. Mathematics and democracy: designing better voting and fair-division procedures. Mathematical and Computer Modelling; 2008.

[6] Brams SJ, Kaplan TR. Dividing the indivisible: procedures for allocating cabinet ministries to political parties in a parliamentary system. Journal of Theoretical Politics 2004;16(2):143.

[7] Carraway RL, Schmidt RL, Weatherford LR. An algorithm for maximizing target achievement in the stochastic knapsack problem with normal returns. Naval Research Logistics 1993;40(2):161–73.

[8] Chen X, Makio J, Weinhardt C. Agent-based simulation on competition of e-auction marketplaces. In: International conference on computational intelligence for modelling, control and automation; 2005. International conference on intelligent agents, web technologies and internet commerce, vol. 2; 2005.

[9] Chiu SY, Lu L, Cox LA. Optimal access control for broadband services: stochastic knapsack with advance information. European Journal of Operational Research 1996;89(1):127–34.

[10] Dobson G. Worst-case analysis of greedy heuristics for integer programming with nonnegative data. Mathematics of Operations Research 1982;7(4): 515–31.
[11] Erel E, Sabuncuoğlu İ, Sekerci H. Stochastic assembly line balancing using beam search. International Journal of Production Research 2005;43(7):1411–26.
[12] Fréville A. The multidimensional 0–1 knapsack problem: an overview. European Journal of Operational Research 2004;155(1):1–21.
[13] Fry MJ, Lundberg AW, Ohlmann JW. A player selection heuristic for a sports league draft. Journal of Quantitative Analysis in Sports 2007;3(2).
[14] Fudenberg D, Tirole J. Game theory. Cambridge, MA: MIT Press; 1991.
[15] Henig MI. Risk criteria in a stochastic knapsack problem. Operations Research 1990;38(5):820–5.
[16] Heyman DP, Sobel MJ. Stochastic models in operations research, vol. II: stochastic optimization. New York: McGraw-Hill; 1984.
[17] Kellerer H, Pferschy U, Pisinger D. Knapsack problems. Berlin: Springer; 2004.
[18] Kleywegt AJ, Papastavrou JD. The dynamic and stochastic knapsack problem with random sized items. Operations Research 2001;49(1):26–41.
[19] Lewis M. Moneyball. W.W. Norton & Company; 2004.
[20] Martello S, Toth P. Knapsack problems: algorithms and computer implementations. New York, NY, USA: Wiley; 1990.
[21] Montgomery DC. Design and analysis of experiments. 5, New York: Wiley; 2001.
[22] Morton DP, Wood RK. On a stochastic knapsack problem and generalizations. In: Advances in computational and stochastic optimization, logic programming, and heuristic search: interfaces in computer science and operations research. Dordrecht: Kluwer Academic Publishers; 1998. p. 149–69.
[23] Pisinger D. Budgeting with bounded multiple-choice constraints. European Journal of Operational Research 2001;129(3):471–80.
[24] Puterman ML. Markov decision processes. Wiley; 1994.
[25] Rinnooy Kan AHG, Stougie L, Vercellis C. A class of generalized greedy algorithms for the multi-knapsack problem. Discrete Applied Mathematics 1993;42(2–3):279–90.
[26] Ross KW, Tsang DHK. The stochastic knapsack problem. IEEE Transactions on Communications 1989;37(7):740–7.
[27] Sabuncuoğlu İ, Bayiz M. Job shop scheduling with beam search. European Journal of Operational Research 1999;118(2):390–412.
[28] Tergesen A. Estates: divvying up the silver. Business Week; May 29, 2008.
[29] Wallace A. Sequential resource allocation utilizing agents. International Journal of Production Research 2003;41(11):2481–99.
[30] Yan D, Mukai H. Stochastic discrete optimization. SIAM Journal on Control and Optimization 1992;30(3):594–612.