Pricing for Customers with Probabilistic Valuations as a Continuous Knapsack Problem

Michael Benisch James Andrews
Norman Sadeh

December 2005
CMU-ISRI-05-137

Institute for Software Research International
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

The research that lead to the development of the software described in this document has been funded by the National Science Foundation under ITR Grant 0205435 and under IGERT grant 9972762.

Keywords: Electronic Commerce, Multi-Agent Supply/Demand Clearing, Continuous Knapsack, TAC SCM

Abstract

In this paper, we examine the problem of choosing discriminatory prices for customers with probabilistic valuations and a seller with indistinguishable copies of a good. We show that under certain assumptions this problem can be reduced to the continuous knapsack problem (CKP). We present a new fast ϵ-optimal algorithm for solving CKP instances with asymmetric concave reward functions. We also show that our algorithm can be extended beyond the CKP setting to handle pricing problems with overlapping goods (e.g. goods with common components or common resource requirements), rather than indistinguishable goods.

We provide a framework for learning distributions over customer valuations from historical data that are accurate and compatible with our CKP algorithm, and we validate our techniques with experiments on pricing instances derived from the Trading Agent Competition in Supply Chain Management (TAC SCM). Our results confirm that our algorithm converges to an ϵ-optimal solution more quickly in practice than an adaptation of a previously proposed greedy heuristic.

1 Introduction

In this paper we study a ubiquitous pricing problem: a seller with finite, indistinguishable copies of a good attempts to optimize profit in choosing discriminatory, take-it-or-leave-it offers for a set of customers. Each customer draws a valuation from some probability distribution known to the seller, and decides whether or not they will accept the seller's offers (we will refer to this as a *probabilistic pricing problem* for short). This setting characterizes existing electronic markets built around supply chains for goods or services. In such markets, sellers can build probabilistic valuation models for their customers, e.g.to capture uncertainty about prices offered by competitors, or to reflect the demand of their own customers.

We show that this pricing problem is equivalent to a continuous knapsack problem (CKP) (i. e. the pricing problem can be reduced to the knapsack problem and vice versa) under two reasonable assumptions: i.) that probabilistic demand is equivalent to actual demand, and ii.) that the seller does not wish to over promise goods in expectation. The CKP asks: given a knapsack with a weight limit and a set of weighted items – each with its value defined as a function of the fraction possessed – fill the knapsack with fractions of those items to maximize the knapsack's value. In the equivalent pricing problem, the items are the customer demand curves. The weight limit is the supply of the seller. The value of a fraction of an item is the expected value of that customer demand curve. The expected value is defined as the probability with which the customer is expected to accept the corresponding offer times the offer price.

Studies of CKPs in Artificial Intelligence (AI) and Operations Research (OR) most often focus on classes involving only linear and quadratic reward functions [10]. We present a fast algorithm for finding ϵ-optimal solutions to CKPs with arbitrary concave reward functions. The class of pricing problems that reduce to CKPs with concave reward functions involve customers with valuation distributions that satisfy the diminishing returns (DMR) property. We further augment our CKP algorithm by providing a framework for learning accurate customer valuation distributions that satisfy this property from historical pricing data.

We also discuss extending our algorithm to solve pricing problems that involve sellers with distinguishable goods that require some indistinguishable shared resources (for example common components or shared assembly capacity). Such problems more accurately represent the movement from make-to-stock production to assemble-to-order and make-to-order production, but involve constraints that are too complex for traditional CKP algorithms.

The class of pricing problems that reduce to CKPs with concave reward functions involve customers with valuation distributions that satisfy the diminishing returns (DMR) property. Therefore, we augment our CKP algorithm by providing a framework for learning accurate customer valuation distributions that satisfy this property from historical pricing data.

The rest of this paper is structured as follows: In Section 2 we discuss related work on the probabilistic pricing and continuous knapsack problems. In Section 3 we present the pricing problem and its equivalence to continuous knapsack. In Section 4 we present our

ϵ-optimal binary search algorithm for concave CKPs. Section 5 presents the framework for learning customer valuation functions. In Section 6 we validate our algorithm and framework empirically on instances derived from the Trading Agent Competition in Supply Chain Management (TAC SCM).

2 Background

2.1 Related Work on Pricing Problems

The pricing problem we study captures many real world settings, it is also the basis of interactions between customers and agents in the Trading Agent Competition in Supply Chain Management. TAC SCM is an international competition that revolves around a game featuring six competing agents each entered by a different team. In TAC SCM simulated customers submit requests for quotes (RFQs) which include a PC type, a quantity, a delivery date, a reserve price, and a tardiness penalty incurred for missing the requested delivery date. Agents can respond to RFQs with price quotes, or bids, and the agent that offers the lowest bid on an RFQ is rewarded with a contractual order (the reader is referred to [3] for the full game specification).

Other entrants from TAC SCM have published techniques that can be adapted to the setting we study. Pardoe and Stone proposed a heuristic algorithm with motivations similar to ours [8]. The algorithm greedily allocates resources to customers with the largest increase in price per additional unit sold. Benisch *et. al.* suggested discretizing the space of prices and using Mixed Integer Programming to determine offers [1], however this technique requires a fairly coarse discretization on large-scale problems.

Sandholm and Suri provide research on the closely related setting of demand curve pricing. The work in [11] investigates the problem of a limited supply seller choosing discriminatory prices with respect to a set of demand curves. Under the assumptions we make, the optimal polynomial time pricing algorithm presented in [11] translates directly to the case when all customers have uniform valuation distributions. Additionally, the result that non-continuous demand functions are $\mathcal{NP}$-Complete to price optimally in [11], implies the same is true of non-continuous valuation distributions.

Additionally there have been several algorithms developed for solving certain classes of continuous knapsack problems. When rewards are linear functions of the included fractions of items, it is well known that a greedy algorithm provides an optimal solution in polynomial time[1]. CKP instances with concave quadratic reward functions can be solved with standard quadratic programming solvers [10], or the algorithm provided by Sandholm and Suri. The only technique that generalizes beyond quadratic reward functions was presented by Mel-

[1]Linear reward functions for CKP would result from a pricing problem where all customers have fixed valuations.

man and Rabinowitz in [7]. The technique in that paper provides a numerical solution to *symmetric* CKP instances where all reward functions are concave and identical[2]. However, this technique involves solving a difficult root finding problem, and its computational costs have not been fully explored.

2.2 Related Work on Learning Valuations

The second group of relevant work involves learning techniques for distributions over customer valuations. Relevant work on automated valuation profiling has focused primarily on first price sealed bid (FPSB) reverse auction settings. Reverse auctions refer to scenarios where several sellers are bidding for the business of a single customer. In the FPSB variant customers collect bids from all potential sellers and pay the price associated with the lowest bid to the lowest bidder. Predicting the winning bid in a first price reverse auction amounts to finding the largest price a seller could have offered the customer and still won. From the point of view of a seller, this price is equivalent to the customer's valuation for the good.

Pardoe and Stone provide a technique for learning distributions over FPSB reverse auctions in TAC SCM [8]. The technique involves discretizing the range of possible customer valuations, and training a regression from historical data at each discrete valuation. The regression is used to predict the probability that a customer's valuation is less than or equal to the discrete point it is associated with. Similar techniques have been used to predict FPSB auction prices for IBM PCs [6], PDA's on eBay [5], and airline tickets [4].

3 Market Model

3.1 P3ID

We define the Probabilistic Pricing Problem with Indistinguishable Goods (P3ID) as follows: A seller has k indistinguishable units of a good to sell. There are n customers that demand different quantities of the good. Each customer has a private valuation for the entirety of her demand, and the seller has a probabilistic model of this valuation. Formally the seller has the following inputs:

- $\boldsymbol{k}$: the number of indistinguishable goods available to sell.
- $\boldsymbol{n}$: the number of customers that have expressed demand for the good.
- $\boldsymbol{q_i}$: the number of units demanded by the $\boldsymbol{i}$th customer.

[2]Identical reward functions for CKP would result from a pricing problem where all customers draw valuations from the same distribution.

- $\boldsymbol{G_i(v_i)}$: a cumulative density function indicating the probability that the $\boldsymbol{i}$th customer draws a valuation below $\boldsymbol{v_i}$. Consequently, $\boldsymbol{1 - G_i(p)}$ is the probability that the customer will be willing to purchase her demand at price $\boldsymbol{p}$.

The seller wishes to make optimal discriminatory take-it-or-leave-it offers to all customers simultaneously. We make the following two assumptions as part of the P3ID to simplify the problem of choosing prices:

- **Continuous Probabilistic Demand (CPD) Assumption**: For markets involving a large number of customers, we can assume that the customer cumulative probability curves can be treated as continuous demand curves. In other words if a customer draws a valuation greater than or equal to \$1000 with probability $\frac{1}{2}$, we assume the customer demands $\frac{1}{2}$ of her actual demand at that price. This is formally modeled by the probabilistic demand of customer i at price p, $q_i * (1 - G_i(p))$.
- **Expected Supply (ESY) Assumption**: We assume that the seller maintains a strict policy against over-offering supply in expectation by limiting the number of goods sold to k (the supply). Note that k is not necessarily the entirety of the seller's inventory.

Under these assumptions, the goal of the seller is to choose a price to offer each customer, p_i, that maximizes the expected total revenue function, $F(p)$:

$$F(p) \;=\; \sum_i (1 - G_i(p_i)) * q_i * p_i \tag{1}$$

Subject to the ESY constraint that supply is not exceeded in expectation:

$$\sum_i (1 - G_i(p_i)) * q_i \;\leq\; k \tag{2}$$

3.2 P3ID and CKP Equivalence

To demonstrate the equivalence between the P3ID and CKP we will show that an instance of either can easily be reduced to an instance of the other. CKP instances involve a knapsack with a finite capacity, k, and a set of n items. Each item has a reward function, $f_i(x)$, and a weight w_i. Including a fraction x_i of item i in the knapsack yields a reward of $f_i(x_i)$ and consumes $w_i * x_i$ of the capacity.

We can easily reduce a P3ID instance to a CKP instance using the following conversion:

- Set the knapsack capacity to the seller's capacity in the P3ID instance.

$$k^{\texttt{CKP}} = k^{\texttt{P3ID}}$$

- Include one item in the CKP instance for each of the n customers in the P3ID instance.
- Set the weight of the ith item to the customer's demanded quantity in the P3ID instances.
$$w_i = q_i$$
- Set the reward function of the ith item to be the inverse of the seller's expected revenue from customer i.
$$f_i(x) = G_i^{-1}(1-x) * x * q_i$$

The fraction of each item included in the optimal solution to this CKP instance, x_i^*, can be converted to an optimal price in the P3ID instance, p_i^*, using the inverse of the CDF function over customer valuations,

$$p_i^* = G_i^{-1}(1 - x_i^*)$$

To reduce a CKP instance to a P3ID instance we can reverse this reduction. The CDF function for the new P3ID instance is defined as,

$$G_i(p) = 1 - \frac{f_i^{-1}(p)}{p * q_i}$$

Once found, the optimal price for a customer, p_i^*, can be translated to the optimal fraction to include, x_i^*, using this CDF function,

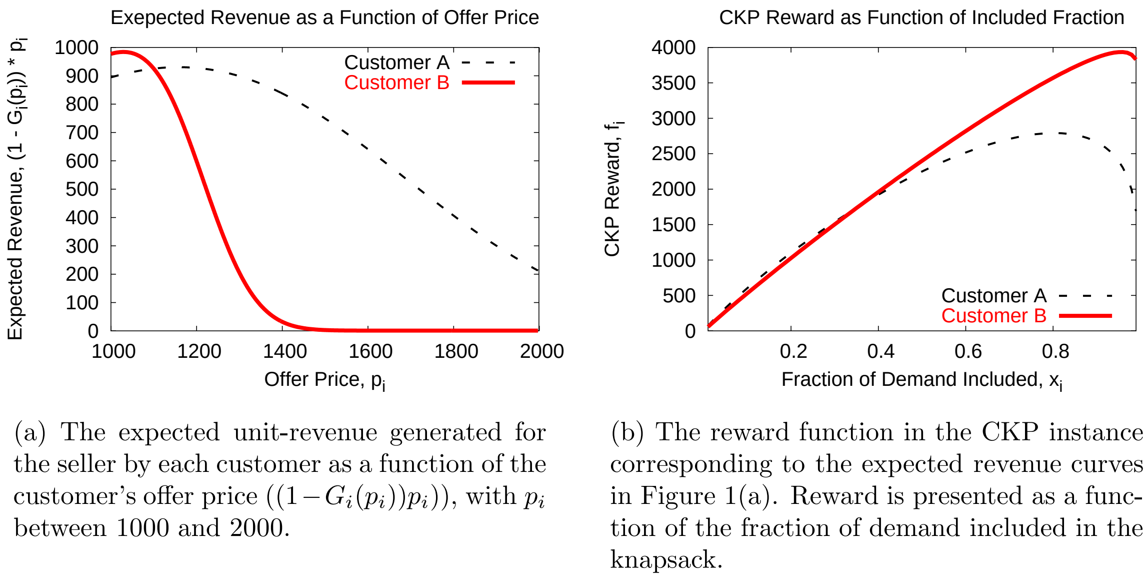
$$x_i^* = G_i(p_i^*)$$

This equivalence does not hold if either the CDF over customer valuations in the P3ID instance, or the reward function in the CKP instance is not invertible. However, if the inverse exists but is difficult to compute numerically, it can be approximated to arbitrary precision by precomputing a mapping from inputs to outputs.

3.3 Example Problem

We provide this simple example to illustrate the kind of pricing problem we address in this paper, and its reduction to a CKP instance. Our example involves a PC Manufacturer with $k = 5$ finished PCs of the same type. Two customers have submitted requests for prices on different quantities of PCs. Customer A demands 3 PCs and Customer B demands 4 PCs. Each customer has a private valuation, if the manufacturer's offer price is less than or equal to this valuation the customer will purchase the PCs.

Based on public attributes that the Customers have revealed, the seller is able to determine that Customer A has a normal unit-valuation (price per unit) distribution with a mean of \$1500 and a standard deviation of \$300, $g_A = \mathcal{N}(1500, 300)$, and Customer B has a normal unit-valuation distribution with mean of \$1200, and a standard deviation of \$100, $g_B = \mathcal{N}(1200, 100)$. Figure 1(a) shows the expected revenue gained by the seller from each customer as a function of the offer price according to these valuation distributions. Figure 1(b) shows the reward functions for the corresponding CKP instance as a function of the fraction of the customer's demand included in the knapsack.



(a) The expected unit-revenue generated for the seller by each customer as a function of the customer's offer price ($(1 - G_i(p_i))p_i)$), with p_i between 1000 and 2000.

(b) The reward function in the CKP instance corresponding to the expected revenue curves in Figure 1(a). Reward is presented as a function of the fraction of demand included in the knapsack.

Figure 1: The expected customer revenue and corresponding reward for the example problem in Section 3.3

Note that in this example, as the price offered to Customer A (or Customer B) increases the probability (or Customer B) accepting it decreases, and hence so does the expected number of PCs sold to that customer. The manufacturer wishes to choose prices to offer each customer to maximize his overall expected revenue, and sell less than or equal to 5 PCs in expectation. In this example it can be shown that the optimal solution is for the manufacturer to offer a unit price of \$1413 to Customer A, which has about a 58% chance of being accepted, and a price of \$1112 to Customer B which has about an 81% chance of being accepted. The total expected revenue of this solution is about \$1212 per unit and it sells exactly 5 units in expectation.

4 Asymmetric Concave CKPs

4.1 Characterizing an Optimal Solution

The main idea behind our algorithm for solving asymmetric CKPs is to add items to the knapsack according to the rate, or first derivative, of their reward functions. We will show that, if all reward functions are concave[3], they share a unique first derivative value in an optimal solution. Finding the optimal solution amounts to searching for this first derivative value. To formalize and prove this proposition we introduce the following notations,

- Let $\boldsymbol{\phi_i(x) = f_i'(x)\frac{1}{w_i}}$, be the first derivative of the $\boldsymbol{i}$'th item's *unit* reward function. Item $\boldsymbol{i}$'s unit reward function is its reward per weight unit.
- Let $\boldsymbol{\phi_i^{-1}(\Delta)}$, be the inverse of the first derivative of $\boldsymbol{i}$'th item's unit reward function. In other words, it returns the fraction of the $\boldsymbol{i}$'th item where its unit reward is changing at the rate $\boldsymbol{\Delta}$.

Proposition 1. *Given a CKP instance, K, if all f_i in K are concave over the interval* $[0, 1]$*, then there exists a unique Δ^* such that, $x_i^* = \phi_i^{-1}(\Delta^*)$, where x_i^* is the fraction of the i'th item in an optimal solution to K.*

Proof. First we will prove that $\phi_i(x)$ is invertible, and that $\phi_i^{-1}(\Delta^*)$ is unique for all i. The reward functions and unit reward functions (since these are simply scaled versions of the originals) in the CKP instance are concave on the interval $[0, 1]$, by the predicate of our proposition. In other words, the first derivative of each unit reward function, $\phi_i(x)$, is decreasing and unique on the interval $[0, 1]$. Because each unit reward function's first derivative is continuous, decreasing, and unique, it is invertible, and its inverse, $\phi_i^{-1}(\Delta)$, is unique[4].

We will now prove that the unit reward functions of any two items, i and j, must share the same first derivative value in the optimal solution. To do this we introduce the following Lemma,

Lemma 1. *If f_i is concave over the interval* $[0, 1]$*, $\phi_i^{-1}(\Delta)$ increases as Δ decreases from $\phi_i(0)$ to* 0*.*

Essentially the Lemma states that as the derivative of item i's unit reward function increases, the fraction of the item included in the knapsack shrinks. This is true because, as we have shown, the derivative is decreasing and unique.

[3]Section 5.1 explains why we can reasonably restrict our consideration to concave reward functions in reductions from P3ID instances.

[4]This inverse may be difficult to characterize numerically. However, the precomputation technique suggested for approximating the inverse of G_i or f_i applies to ϕ_i as well.

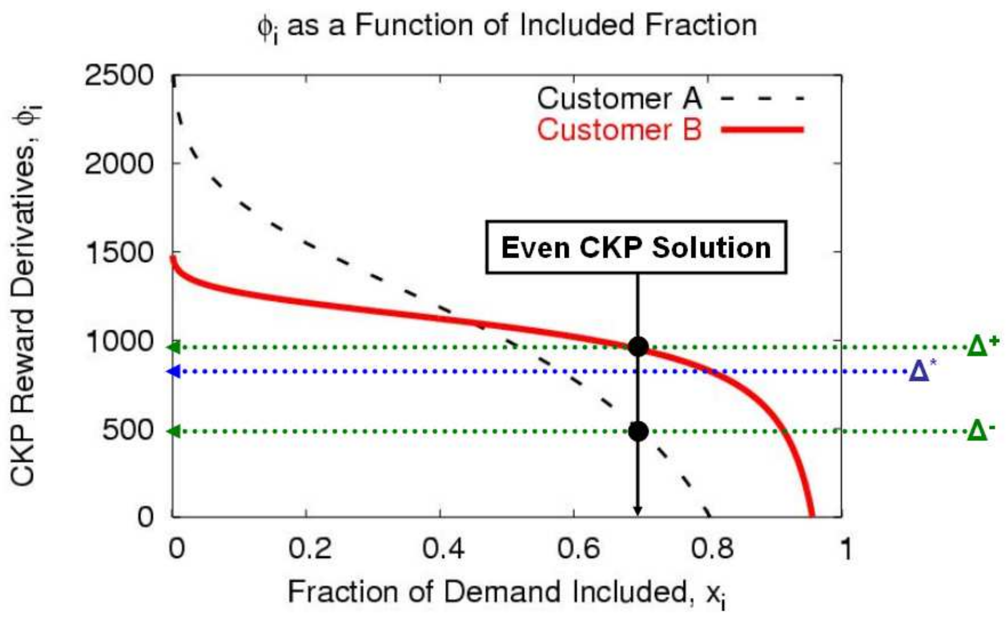


Figure 2: Initial values for Δ^+ and Δ^- are computed from the `even_CKP` solution for the example problem in Section 3.3.

For the remainder of the proof there are two cases we must consider:

Case 1: *the knapsack is not full in the optimal solution.* In this case the unit reward functions will all have derivatives of 0, since every item is included up to the point where its reward begins to decrease[5].

Case 2: *the knapsack is full in the optimal solution.* In this case we will assume that f_i and f_j do not share the same derivative value, and show this assumption leads to a contradiction. Specifically, we can assume, *without loss of generality*, that the reward function of item i has a larger first derivative than j, i.e. $\phi_i(x_i^*) > \phi_j(x_j^*)$. Therefore, there must exist some ϵ, such that adding it to item j's unit reward derivative maintains the inequality, $\phi_i(x_i^*) > \phi_j(x_j^*) + \epsilon$. We can then construct an alternative solution to K as follows:

- Set x_j in our alternative solution to be the fraction of item j that provides its original derivative plus ϵ,

$$x'_j = \phi_j^{-1}(\phi_j(x_j^*) + \epsilon)$$

[5]We assume that all reward functions have derivatives ≤ 0 when an item is entirely included in the knapsack, since the item cannot possibly provide any additional reward.

- By Lemma 1 we know that $x'_j < x^*_j$, which provides some excess space, α, in the knapsack, $\alpha = w_j(x^*_j - x'_j)$. We can fill the empty space with item i, up to the point where the knapsack is full, or its derivative decreases by ϵ,

$$x'_i = \min\left(x^*_i + \frac{\alpha}{w_i}, \phi_i^{-1}(\phi_i(x^*_i) - \epsilon)\right)$$

  It must be that $x'_i > x^*_i$. Either all of the knapsack space from item j was added, in which case the fraction of item i clearly increased. Otherwise, its derivative value decreased by ϵ, which, by Lemma 1, must have increased its included fraction. If $\phi_i(x'_i)$ decreased by ϵ before the knapsack filled up, we can reallocate the excess space to j,

$$x'_j = (k - x_i)\,\frac{1}{w_j}$$

Notice that we have constructed our alternate solution by moving the same number of knapsack units from item j to item i. In our construction we guaranteed that item i was gaining more reward per unit during the entire transfer. Therefore, the knapsack space is more valuable in the alternate solution. This contradicts our assumption that x^*_i and x^*_j were part of an optimal solution.

We have shown that any two unit reward functions must share the same derivative value, Δ^*, in an optimal solution. This implies that *all* unit reward functions must share the derivative value in an optimal solution (since no two can differ). $\square$

4.2 Finding Δ^*

In our proof of Proposition 1 we showed that $\Delta^* \geq 0$. We also showed that as Δ increases, the fraction of each item in the knapsack decreases. Thus, one method for finding Δ^* would be to begin with $\Delta = 0$ and increment by ϵ until the resulting solution is feasible (fits in the knapsack). However, much of this search effort can be reduced by employing a binary search technique.

Figure 3 presents pseudo-code for a binary search algorithm that finds solutions provably within ϵ of an optimal reward value. The algorithm recursively refines its upper and lower bounds on Δ^*, Δ^+ and Δ^-, until the reward difference between solutions defined by the bounds is less than or equal to ϵ.

The initial bounds, shown in Figure 2, are derived from a simple feasible solution where the same fraction of each item is included in the knapsack (see `even_CKP` in Figure 3). The largest derivative value in this solution provides the upper bound, Δ^+. This is because we can reduce the included fractions of each item to the point where all of their derivatives equal Δ^+, and guarantee the solution is still feasible. By the same reasoning, the smallest

```
procedure $\epsilon$-opt_CKP($K$)
  $x \leftarrow$ even_CKP($K$)
  $\Delta^+ \leftarrow \max_i \phi_i^{-1}(x_i)$
  $\Delta^- \leftarrow \min_i \phi_i^{-1}(x_i)$
  return binary_search($\Delta^+, \Delta^-, K$)

procedure binary_search($\Delta^+, \Delta^-, K$)
  if converged($\Delta^+, \Delta^-, K$) then
    $x^+ \leftarrow \{\phi_1^{-1}(\Delta^+), \dots, \phi_n^{-1}(\Delta^+)\}$
    return $x^+$
  end if
  $\delta \leftarrow \frac{\Delta^+ - \Delta^-}{2}$
  if feasible($\{\phi_1^{-1}(\delta), \dots, \phi_n^{-1}(\delta)\}$, $K$) then
    return binary_search($\delta, \Delta^-, K$)
  else
    return binary_search($\Delta^+, \delta, K$)
  end if

procedure even_CKP($K$)
  $\hat{w} \leftarrow \sum_i w_i$
  return $\{\frac{k}{\hat{w}}, \dots, \frac{k}{\hat{w}}\}$

procedure feasible($x, K$)
  return $\sum_i w_i x_i \leq k$

procedure converged($\Delta^+, \Delta^-, K$)
  $x^+ \leftarrow \{\phi_1^{-1}(\Delta^+), \dots, \phi_n^{-1}(\Delta^+)\}$
  $x^- \leftarrow \{\phi_1^{-1}(\Delta^-), \dots, \phi_n^{-1}(\Delta^-)\}$
  return $\sum_i f_i(x^+) - f_i(x^-) \leq \epsilon$
```

Figure 3: Pseudo-code for an ϵ-optimal concave CKP binary search algorithm.

derivative value in the simple solution provides a lower bound Δ^-. Figure 2 shows how initial values of Δ^+ and Δ^- are computed from the even solution on the Example problem from Section 3.3.

During each iteration, a new candidate bound, δ, is computed by halving the space between the prior bounds. The process continues recursively: if the new bound defines a feasible solution it replaces the old upper bound, otherwise (if it is not a valid upper bound), it replaces the old *lower* bound.

When the algorithm converges the solution defined by Δ^- is guaranteed to be feasible and within ϵ of the optimal solution. Convergence is guaranteed since we have proved that Δ^* exists, and the bounds get tighter after each iteration. It is difficult to provide theoretical guarantees about the number of iterations, since convergence is defined in terms of the instance-specific reward functions. However, the empirical results in Section 6 show that the algorithm typically converges exponentially fast in the number of feasibility checks.

4.3 Shared Resource Extension

Our ϵ-optimal binary search algorithm can be extended to solve problems involving more complex resource constraints than typically associated with CKPs. In particular, the algorithm can be generalized to solve reductions of Probabilistic Pricing Problems with Shared Resources (P3SR). P3SR instances involve sellers with multiple *distinguishable* goods for sale. Each good in a P3SR consumes some amount of finite shared resources, such as components or assembly time. This model allows for techniques capable of supporting the movement from make-to-stock practices to assemble-to-order or make-to-order practices.

By applying the reduction described in Section 3.2, a P3SR instance can be converted to a problem similar to a CKP instance. However, the resource constraint in the resulting problem is more complex than ensuring that a knapsack contains less than its weight limit. It could involve determining the feasibility of a potentially $\mathcal{NP}$-Hard scheduling problem, in the case of a shared assembly line and customer demands with deadlines. Clearly, this would require, among other things, changing the feasibility checking procedure (see `feasible()` in Figure 3), and could make each check substantially more expensive.

5 Customer Valuations

5.1 Diminishing Returns Property

Our algorithm was designed to solve CKP reductions of P3ID instances. Recall that it applies only when the reward functions are concave over the interval $[0, 1]$. This is not a particularly restrictive requirement. In fact, this is what economists typically refer to as the Diminishing Returns[6] (DMR) property. This property is generally accepted as characterizing many real-world economic processes [2].

Definition: *The DMR property is satisfied for a P3ID instance when, for a given increase in any customer's filled demand, the increase in the seller's expected revenue is less per unit than it was for any previous increase in satisfaction that customer's demand.*

Note that our market model also captures the setting where customer valuations are determined by bids from competing sellers. In this setting normally distributed competing bid prices can also be shown to result in concave reward functions. This situation is representative of environments where market transparency leads sellers to submit bids that hover around a common price.

5.2 Normal Distribution Trees

We consider a technique which a seller may use to model a customer's valuation distribution. It will use a normal distribution to ensure our model satisfies the desired DMR property. We assume that customers have some public attributes, and the seller has historical data associating attributes vectors with valuations.

[6]This is also occasionally referred to as the Decreasing Marginal Returns property.

Our technique trains a regression tree to predict a customer's valuation from the historical pricing data. A regression tree splits attributes at internal nodes, and builds a linear regression that best fits the training data at each leaf. When a valuation distribution for a new customer needs to be created, the customer is associated with a leaf node by traversing the tree according to her attributes. The prediction from the linear model at the leaf node is used as the mean of a normal valuation distribution, and the standard deviation of the distribution is taken from training data that generated the leaf.

Formally the regression tree learning algorithm receives as input,

- $\boldsymbol{n}$: the number of training examples.
- $\boldsymbol{a_i}$: the attribute vector of the $\boldsymbol{i}$'th training example.
- $\boldsymbol{v_i}$: the valuation associated with the $\boldsymbol{i}$'th training example.

A regression tree learning algorithm, such as the M5 algorithm [9], can be used to learn a tree, T, from the training examples. After the construction of T, the j'th leaf of the tree contains a linear regression over attributes, $y_j(a)$. The regression is constructed to best fit the training data associated with the leaf. The leaf also contains the average error over this data, s_j.

The regression tree, T, is converted to a distribution tree by replacing the regression at each node with a normal distribution. The mean of the normal distribution at the j'th leaf is set to the prediction of the regression, $\mu_j = y_j(a)$. The standard deviation of the distribution at the j'th leaf is set to the average error over training examples at the leaf, $\sigma_j = s_j$. Figure 4 shows an example of this kind of normal distribution tree.

5.3 Learning Customer Valuations in TAC

TAC SCM provides an ideal setting to evaluate the distribution tree technique described in the previous section. Each customer request in TAC SCM can be associated with several attributes. The attributes include characterizations of the request, such as its due date, PC type, and quantity. The attributes also include high and low selling prices for the requested PC type from previous simulation days. Upon the completion of a game, the price at which each customer request was filled is made available to agents. This data can be used with the technique described in the previous section to train a normal distribution tree. The tree can then be used in subsequent games to construct valuation distributions from request attributes.

Figure 5 shows the accuracy curve of a normal distribution tree trained on historical data with an M5 learning algorithm. Training instances were drawn randomly from customer requests in the 2005 Semi-Final round of TAC SCM and testing instances were drawn from the Finals. The attributes selected to characterize each request included: the due date, PC

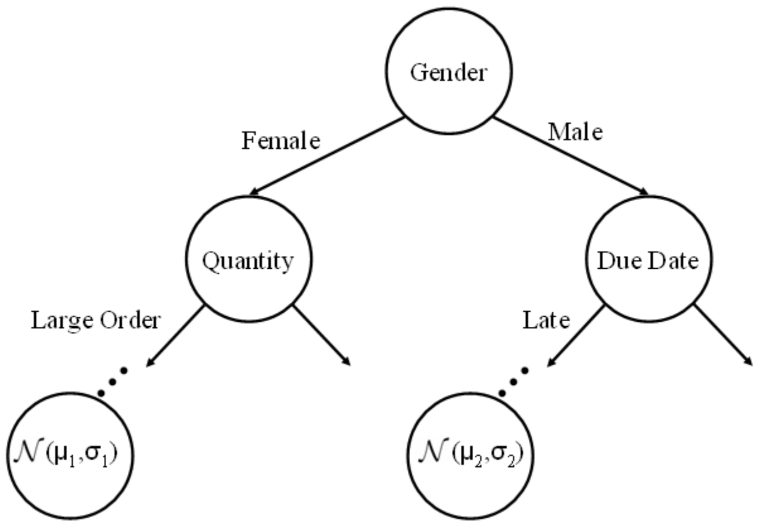


Figure 4: An example Normal Distribution Tree

type, quantity, reserve price, penalty, day on which the request was placed, and the high and low selling prices of the requested PC type from the previous 5 game days.

The error of the distribution was measured in the following way: starting at $p = .1$, and increasing to $p = .9$, the trained distribution was asked to supply a price for all test instances that would fall below the actual closing price (be a winning bid) with probability p. The average absolute difference between p and the actual percentage of test instances won was considered the error of the distribution. The experiments were repeated with 10 different training and testing sets. The results show that normal distribution trees can be used to predict distributions over customer valuations in TAC SCM with about 95%, accuracy after about 25,000 training examples.

6 Empirical Evaluation

6.1 Empirical Setup

Our experiments were designed to investigate the convergence rate of the ϵ-optimal binary search algorithm. We generated 100 CKP instances from P3ID instances based on the pricing problem faced by agents in TAC SCM. The P3ID instances were generated by randomly selecting customer requests from the final round of the 2005 TAC SCM. Each customer request in TAC SCM has a quantity randomly chosen uniformly between 1 and 20 units. Normal probability distributions were generated to approximate the customer valuations

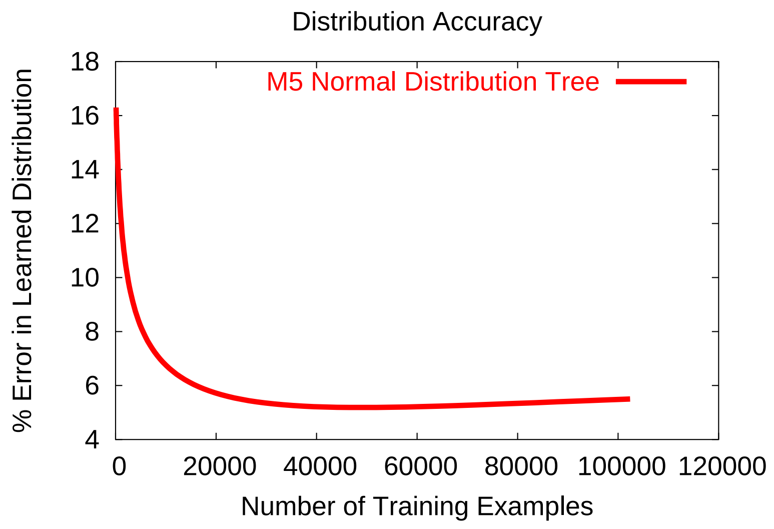


Figure 5: The accuracy curve of an M5 normal distribution tree as the number of training instances increases.

of each customer using the technique described in Section 5 with an M5 Regression Tree learning algorithm. The learning algorithm was given 50,000 training instances from the 2005 TAC SCM Semi-Final rounds.

We tested our algorithm against the even solution, which allocates equal resources to each customer, and the greedy heuristic algorithm used by the first place agent, TacTex [8]. Figure 6 provides pseudo-code adapting the TacTex algorithm to solve the P3ID reductions. It greedily adds fractions of items to the knapsack that result in the largest increases in expected unit-revenue.

We performed three sets of experiments. The first set of experiments provided each algorithm with 20 PCs to sell in expectation, and the same 200 customer requests (this represents a pricing instance of a TAC SCM agent operating under a make-to-stock policy). Figure 7(a) shows each algorithm's percentage of an optimal expected revenue after each feasibility check. For the second set of experiments, the algorithms were given 200 customer requests, and their PC supply was varied by 10 from $k = 10$, to $k = 100$. Figure 7(b) shows the number of feasibility checks needed by the binary search and greedy algorithms to reach solutions within 1% of optimal. The last set of experiments fixed $k = 20$ and varied n by 100 from $n = 200$ to $n = 1000$. Figure 7(c) shows the number of feasibility checks needed by each algorithm to reach a solution within 1% of optimal as n increased.

```
procedure greedy_CKP($K$)
  converged $\leftarrow \perp$
  while $\neg$converged and $\sum_i x_i < n$ do
    $i^* \leftarrow \text{argmax}_i$ unit_reward_increase$(i, x_i, K)$
    $\delta^* \leftarrow$ best_increase$(i^*, x_{i^*}, K)$
    if feasible$(\{x_1, \ldots, x_{i^*} + \delta^*, \ldots, x_n\}, K)$ then
      $x_{i^*} \leftarrow x_{i^*} + \delta^*$
    else
      $x_{i^*} \leftarrow x_{i^*} + \frac{1}{w_i}(k - \sum_i x_i w_i)$
      converged $\leftarrow \top$
    end if
  end while
  return $\{x_1, \ldots, x_n\}$

procedure unit_reward_increase$(i, x_i, \delta, K)$
  $\delta^* \leftarrow$ best_increase$(i, x_i, K)$
  return $\frac{1}{w_i \delta^*}(f_i(x_i + \delta^*) - f_i(x_i))$

procedure best_increase$(i, x_i, K)$
  return $\text{argmax}_\delta \frac{f_i(x_i+\delta) - f_i(x_i)}{\delta}$
```

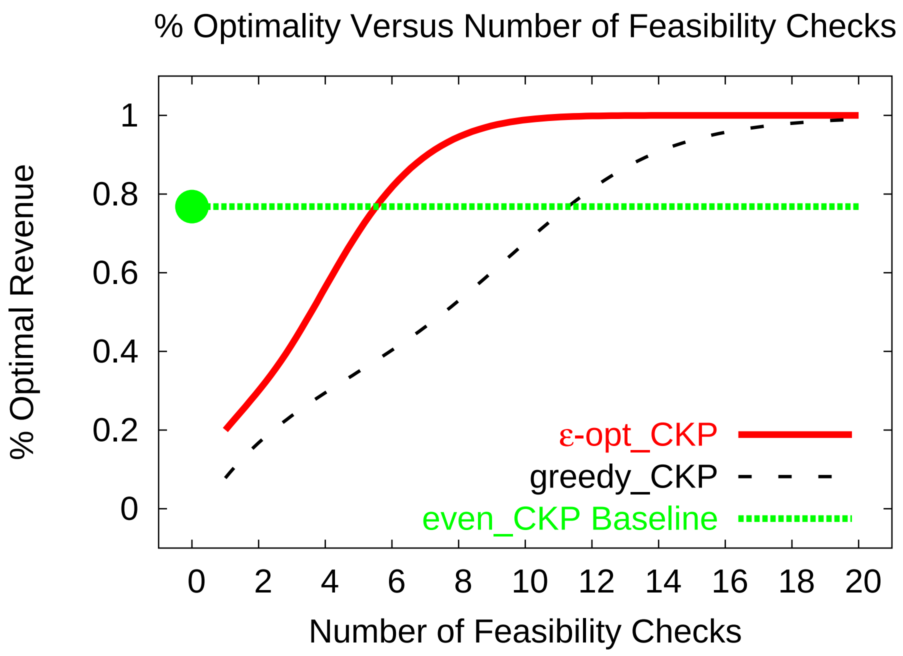
Figure 6: Pseudo-code for the greedy heuristic algorithm used by the 2005 first placed agent, TacTex.

6.2 Empirical Results

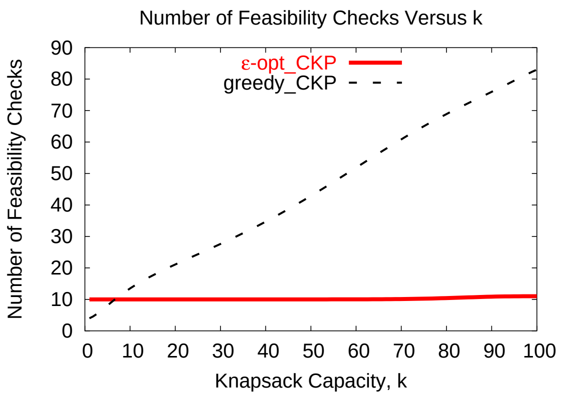
The results presented in Figure 7 compare the optimality of the CKP algorithms to the number of feasibility checks performed. This comparison is important to investigate for two reasons, i.) because it captures the convergence rate of the algorithms, and ii.) because these algorithms are designed to be extended to shared resource settings discussed in Section 4.3 where each feasibility check involves solving (or approximating) an $\mathcal{NP}$-Hard scheduling problem.

The first set of results, shown in Figure 7(a), confirms that the ϵ-optimal binary search algorithm converges exponentially fast in the number of consistency checks. In addition, the results confirm the intuition of Pardoe and Stone in [8] that the greedy heuristic finds near optimal solutions on CKP instances generated from TAC SCM. However, the results also show that it has a linear, rather than exponential, convergence rate in terms of consistency checks. This indicates that our binary search algorithm scales much better than the greedy technique. Finally, the first set of results shows that the even solution, which does not use consistency checks, provides solutions to TAC SCM instances that are about 80% optimal on average.

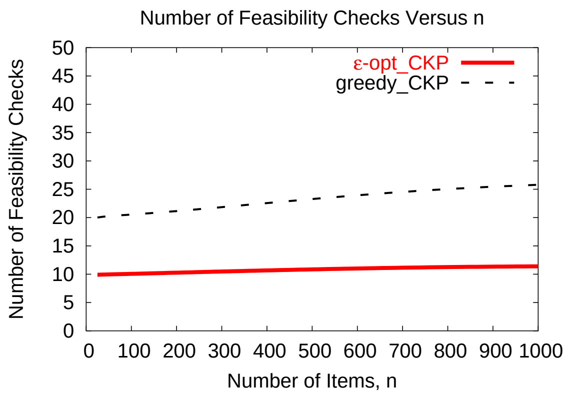
Figures 7(b) and 7(c) investigate how the number of feasibility checks needed to find near (within 99% of) optimal solutions changes as the supply and number of customers increase. The even solution is not included in these results because it does not produce near optimal solutions. The results shown in Figure 7(b) show that the number of consistency checks



(a) This graph shows how the optimality of each algorithm improves with each feasibility check it uses.



(b) The number of feasibility checks needed to reach a solution within 1% of optimal as k increases.



(c) The number of feasibility checks needed to reach a solution withing 1% of optimal as n increases.

Figure 7: Performance of CKP algorithms on instances reduced from TAC SCM pricing problems. Unless otherwise specified, results are averaged over 100 CKP instances with $n = 200$ and $k = 20$.

used by the greedy algorithm increases linearly with the size of the knapsack, whereas the convergence rate of the binary search algorithm does not change. The results shown in Figure 7(c) show that the number of consistency checks used by both algorithms does not significantly increase with the number of customers.

7 Conclusion

In this paper we presented a model for the problems faced by sellers that have multiples copies of an indistinguishable good to sell to multiple customers. We have modeled this problem as a Probabilistic Pricing Problem with Indistinguishable Goods (P3ID) and formally shown its equivalence the Continuous Knapsack Problem (CKP). We showed that P3ID instances with customer valuation distributions that satisfy the DMR property reduce to CKP instances with arbitrary concave reward functions. Prior work had not addressed CKP instances with asymmetric *nonlinear* concave reward functions. To address this gap, we provided a new ϵ-optimal algorithm for such CKP instances. We showed that this algorithm converges exponentially fast in practice. We also provide a technique for learning normal distributions of customer valuations from historical data, by extending existing regression tree learning algorithms. We validated our distribution learning technique and our binary search technique for the P3ID on data from 2005 TAC SCM. Our results showed that our learning technique achieves about 95% accuracy in this setting, indicating that TAC SCM is a good environment in which to apply our P3ID model. Our results further showed that our binary search algorithm for the P3ID scales substantially better than a technique adapted from the winner of the 2005 TAC SCM competition.

8 Acknowledgments

The research reported in this paper has been funded by the National Science Foundation under ITR Grant 0205435.

References

[1] M. Benisch, A. Greenwald, I. Grypari, R. Lederman, V. Naroditskiy, and M. C. Tschantz. Botticelli: A supply chain management agent. In *Proceedings of AAMAS '04*, pages 1174–1181, New York, July 2004.

[2] K. E. Case and R. C. Fair. *Principles of Economics (5th ed.)*. Prentice-Hall, 1999.

[3] J. Collins, R. Arunachalam, N. Sadeh, J. Eriksson, N. Finne, and S. Janson. The supply chain management game for the 2005 trading agent competition. Technical Report CMU-ISRI-04-139, Carnegie Mellon University, 2005.

[4] O. Etzioni, R. Tuchinda, C. A. Knoblock, and A. Yates. To buy or not to buy: mining airfare data to minimize ticket purchase price. In *Proceedings of KDD'03*, pages 119–128, New York, NY, USA, 2003. ACM Press.

[5] R. Ghani. Price prediction and insurance for online auctions. In *Proceedigns of KDD'05*, pages 411–418, New York, NY, USA, 2005. ACM Press.

[6] D. Lawrence. A machine-learning approach to optimal bid pricing. In *Proceedings of INFORMS'03*, 2003.

[7] A. Melman and G. Rabinowitz. An efficient method for a class of continuous knapsack problems. *Society for Industrial and Applied Mathematics Review*, 42(3):440–448, 2000.

[8] D. Pardoe and P. Stone. Bidding for customer orders in TAC SCM. In *Proceedings of AAMAS-04 Workshop on Agent-Mediated Electronic Commerce*, 2004.

[9] J. R. Quinlan. Learning with Continuous Classes. In *5th Australian Joint Conference on Artificial Intelligence*, pages 343–348, 1992.

[10] A. G. Robinson, N. Jiang, and C. S. Lerme. On the continuous quadratic knapsack problem. *Math. Program.*, 55(1-6):99–108, 1992.

[11] T. Sandholm and S. Suri. Market clearability. In *Proceedings of IJCAI'01*, pages 1145–1151, 2001.