Monte Carlo Algorithms for Expected Utility Estimation in Dynamic Purchasing

Buffett, Scott

National Research
Council Canada

Conseil national de
recherches Canada

National Research
Council Canada

Institute for
Information Technology

Conseil national
de recherches Canada

Institut de technologie
de l'information

*Monte Carlo Algorithms for Expected Utility Estimation in Dynamic Purchasing **

Buffett, S.

March 2004

* published as PhD Thesis. Faculty of Computer Science, University of New Brunswick. Fredericton, New Brunswick, Canada. March 2004. 191 Pages. NRC 46557.

Monte Carlo Algorithms for Expected Utility Estimation in Dynamic Purchasing

by

Scott Buffett

BCS — University of New Brunswick 1996

MCS — University of New Brunswick 1998

A Thesis Submitted in Partial Fulfillment of

the Requirements for the Degree of

Doctor of Philosophy

in the

Graduate Academic Unit of Computer Science

Supervisors:	Bruce Spencer, Ph.D., NRC and Faculty of Computer Science J. D. Horton, Ph.D., Faculty of Computer Science
Examining Board:	Ebrahim Roumi, Ph.D., Faculty of Business, UNBSJ (Chair) George Stoica, Ph.D., Dept of Mathematical Sciences, UNBSJ Huajie Zhang, Ph.D., Faculty of Computer Science
External Examiner:	Prof. Dr. Michael M. Richter, Department of Computer Science, University of Kaiserslautern

This thesis is accepted.

Dean of Graduate Studies

THE UNIVERSITY OF NEW BRUNSWICK

February, 2004

Dedication

To my grandparents, Harold and Julia Buffett, Lefty and Regina Wright.

Abstract

This thesis describes a theory for decision-making in a dynamic purchasing environment where one of possibly many bundles of items must be purchased from possibly many suppliers. The *online combinatorial purchasing problem* is defined as the problem with which a purchase agent in such an environment is faced when deciding which combination of items, from whom and at what time to buy in order to maximize overall satisfaction. Expected utility maximization is used as the criterion on which decision-making is based. To facilitate the exchange of probabilistic and temporal information between suppliers and purchasers, the PQR protocol is defined. The theory considers two distinct purchasing models, one in which only complete bundle purchases can be made at any time, and one in which partial bundle purchases are allowed. In the first model, a decision procedure that exploits future time intervals where several options will be available is developed that provably yields higher expected utility than simply pursuing the bundle with highest expected utility. For the second model, the QR-tree is defined as a decision tree that can be exponentially smaller than conventional decision trees when used to model the same system of decisions. Efficient Monte Carlo methods are developed that solve the QR-tree in linear time on the number of nodes in the tree. Results show that these methods estimate expected utility as much as 25 times more accurately than a greedy method that always pursues the bundle with the current highest expected utility.

Acknowledgements

I would like to thank my supervisor, Dr. Bruce Spencer, for all of the help, support and encouragement he offered during my research for this thesis. Through thick and thin, his confidence in the progress and direction of my work was unwavering. Also deserving of thanks is Dr. Jane Fritz, who offered financial support by employing me in the Faculty of Computer Science, and contributed to my success in ways too numerous to mention.

Many thanks also go to the National Research Council Canada for funding me as a guest worker, Dr. Joe Horton for his help as co-supervisor, the NRC-UNB Business Rules for Web Services Group, NRC's Internet Logic Group and all of my colleagues at NRC IIT - e-Business.

Last but not least, my most heartfelt gratitude goes to my parents Hal and Donna, who gave me every chance in life, and my wife Kathy, whose patience and encouragement I needed the most throughout the entire endeavour.

Contents

List of Tables

List of Figures

List of Symbols

Symbol	Meaning
A_n	set of above values of node n (monetary amounts that could possibly be spent before n is encountered)
A'_n	a small subset of A_n
$\mathcal{B}$	set of bundles
$\mathcal{B}_v$	set of valid bundles
b	bundle
$ci(CS)$	comparison interval of CS
CS	comparison set
$csc(\mathcal{B})$	comparison set cover of $\mathcal{B}$
CT_d	classification tree for decision node d
$ct_d(a)$	expected utility of d given above value $a \in A_d$ as computed using CT_d
D	set of decision nodes
$E[\tilde{x}]$	expected value of $\tilde{x}$
$E_{gr}(n)$	the true (possibly unknown) expected utility if the greedy method is used from n until the end
E_{im}	the expected utility of the improved decision procedure
E_{na}	the expected utility of the naïve decision procedure

Symbol	Meaning
$E_{rq}(n)$	the true (possibly unknown) expected utility if the restricted q-horizon method is used from n until the end
I	set of items
i	item
j	the highest expected utility over all bundles (chapter 6)
K	set of outcomes for prices of items in a QR-tree
k	the highest expected utility over all comparison sets (chapter 6)
k_{bz}	scaling constant for $u_b(b)u_z(z)$ in the two-attribute utility function
k_b	scaling constant for $u_b(b)$ in the two-attribute utility function
k_z	scaling constant for $u_z(z)$ in the two-attribute utility function
p	probability distribution or density function
P	set of purchase nodes
$pi(b)$	purchase interval of bundle b
$qh(d)$	q-horizon of decision node d
$qs(d)$	q-set of decision node d
$qsh(d)$	q-subhorizon of decision node d
$qss(d)$	q-subset of decision node d
$qssc(d)$	q-subset-complement of decision node d
T_{pp}	purchase procedure tree
T_{qr}	QR-tree
$t_p(x)$	prequote time of item, node or purchase interval x
$t_q(x)$	quote time of item, node or purchase interval x
$t_r(x)$	rescind time of item, node or purchase interval x
$\tilde{u}(b)$	the unknown outcome of utility of purchasing b
$u(b, z)$	utility of bundle b at cost z
$u_b(b)$	utility of bundle b

Symbol	Meaning
$u_z(z)$	utility of cost z
$up_n(a)$	the utility projection function for node n (gives the expected utility at n given that an above value $a \in A_n$ is spent before n is encountered)
$up'_n(x)$	same as $up_n(a)$ except A'_n values (rather that A_n values) are mapped to expected utilities
$\tilde{x}$	the unknown outcome of random variable x
Z	set of monetary units
μ	mean
μ_r	relative mean
σ	standard deviation
σ_r	relative standard deviation
θ_{ct}	estimate of expected utility as given by the classification tree method
θ_{gr}	estimate of expected utility as given by the greedy method
$\theta_{gr-rq}(n)$	the estimated expected utility at n if the greedy method is used throughout n's q-region, and the restricted q-horizon method is used afterward
θ_{PT}	estimate of expected utility as given by the q-subset discretization method
θ_{rq}	estimate of expected utility as given by the restricted q-horizon method
$\theta'_{rq}(n)$	the expected utility at n estimated by the restricted q-horizon method if all q-horizons in n's q-region are eliminated (i.e. for any decision node d in the q-region, $qsh(d')$ is subtracted from $qh(d)$ for every descendent d')
$\theta_{rq}(n)$	the expected utility at n estimated by the restricted q-horizon method

Chapter 1

Introduction

As early as the 1960's, when the idea of computer connectivity became realistic, people in the business world envisioned the possibility of performing transactions electronically. As technology has improved and electronic commerce has become more common, strategic purchasing tools that can cleverly ascertain the true value of many different options over possibly many different markets are becoming extremely important. More and more businesses are turning to Web-based pricing tools that sift through large volumes of data on product revenues, inventory levels and consumer activity to determine how much to charge for certain items during certain periods of time [Kee03]. This "perfect pricing" translates into higher profits for sellers, mostly at the expense of the buyer. To combat this trend, buyers need the decision analysis technology that can properly assess not only the current purchasing options, but also the positive or negative potential of future opportunities.

This thesis presents a theory for decision-making in a dynamic electronic marketplace where the buyer needs to purchase a set of items (referred to as a *bundle*). In this framework, there may be alternative bundles (two or more items may be interchangeable, for example) from which to choose. Each bundle has a particular preference level based on the quality of the items, the compatibility of the items, the reputations of the suppliers or any other factors that may make one bundle of items more desirable than another. The dynamic element implies that, at any given time, some items may be currently available at a set price for a fixed length of time, while other items may be available during future periods at prices that are currently unknown. The buyer must therefore make purchasing decisions based on incomplete information, in hopes of ultimately completing the best set of purchases in terms of item preference and cost.

While purchase decision-making is certainly not a novel concept, applying existing theories to e-commerce, where potential offers from countless suppliers can be considered at once, is often impractical. *Expected utility theory* [vNM47] was proposed by von Neumann and Morgenstern as a means for computing a numeric preference level (called a utility) for each choice in a decision problem for which perhaps only a probability measure is known for the consequences, based on the decision-maker's attitude towards risk. However, with expected utility theory alone, it is difficult to model a sequence of decisions where the consequences of some choices are learned before subsequent decisions are made. This modeling is necessary in bundle purchasing. Consider an example where the buyer needs to decide whether or not to buy some item A, and there are two bundles containing A: A and B or A and C. Perhaps the prices of B and C will be known before the decision on which of the two to buy needs to be made, but not until after the decision on whether to buy A must be made, making it difficult to determine the expected utility of buying A. This sequence of decisions can be modeled by a *decision tree* [Mag64]. A technique referred to as *rollback solution* can be applied to the model in reverse order with respect to time in order to determine the expected utility of each choice at the original decision. In a decision tree there is a branch for each choice at each decision, as well as for each possible consequence of each choice. Thus decision trees can easily get unmanageably large, especially if there are many consequences (i.e. price outcomes) for choices. Hespos and Strassman [HS65] proposed the use of Monte Carlo simulation [MU49] in decision trees for investment decision-making. This was done with the purpose of eliminating the high branching factor at chance nodes by instead simply taking random samples from probability distributions. However, their technique only considered consequences that would occur after a decision was made (which is common in investment decision-making). It was not designed to handle the computational complexity that arises when solving a future decision during rollback where some consequences associated with the decision (i.e. outcomes of some item costs) occur *before* the decision is made. This thesis attempts to solve these problems and present a solid theory of how such decisions can be made in computationally feasible ways.

After formalizing the problem of determining utility functions for bundles and money and computing the expected utility of bundle purchases, a simple method for making decisions where all items in a bundle must be purchased at once is presented. The decision problem is much less complicated in this case since there are no subsequent decisions to be made after the buyer chooses to purchase something. The theory is then extended to the general case where the buyer makes decisions at the item level. The QR-tree is proposed as a structure to efficiently model the system of subsequent decisions that will follow a choice in a decision. This tree is shown to be exponentially smaller than a conventional decision tree if used to model the same system of decisions. Three Monte Carlo techniques for solving the QR-tree to determine the expected utility of each choice are developed. Since the time complexity of each technique grows linearly with the size of the QR-tree, then each of these techniques is exponentially faster than decision tree rollback solution, as well as the method proposed by Hespos and Strassman.

The main results of the thesis are as follows: Initially the problem of bundle purchasing in this model, termed the *Online Combinatorial Purchasing Problem*, is formally defined. A decision procedure is proposed for the simple domain in which only complete bundles are chosen, and is proven to always yield expected utility greater than or equal to that of a naïve procedure that simply instructs the buyer to buy a bundle if it has the highest expected utility over all bundles. In the more general domain where the buyer makes decisions at the item level, the *QR-tree* is defined as a decision tree that can be exponentially smaller than a conventional decision tree. A solution method for the QR-tree that considers all possible discrete price outcomes is developed, as well as three Monte Carlo algorithms: the restricted q-horizon method, the classification tree method and the q-subset discretization method. The expected utility of the restricted q-horizon method is proven to be at least as high as the expected utility of using the greedy method of always pursuing the bundle with the highest expected utility. The classification tree method is shown to provide an underestimate of the true utility when a reasonable restriction is placed on the execution of the purchase procedure. Testing shows that this estimate is also very close to the true expected utility. Finally, the q-subset discretization method is shown to provide the most accurate estimate of all, with a standard error just $\frac{1}{26}$ that of the greedy method that judges the expected utility of a choice to be the maximum expected utility of all bundles that can potentially be procured as a result of that choice. To reduce the required sampling workload, the antithetic variate [HM56] sampling method is shown to be an effective variance reduction technique in Monte Carlo simulation in this domain. For the simple case of finding the expected highest utility of two simultaneous choices, antithetic variate sampling is shown to reduce the variance (and thus the number of simulations needed) to $\frac{1}{6}$ that of crude Monte Carlo.

While much of the required background is given in chapter 2 and other techniques and terminology are introduced when necessary, the reader is expected to have some prior knowledge of basic probability theory and statistics. Knowledge of standard decision analysis tools such as decision trees and Monte Carlo simulation would be an asset, as would a basic understanding of calculus.

The thesis is organized as follows: Chapter 2 outlines the required background on e-commerce, market clearing and utility theory. After introducing the online combinatorial purchasing problem and the required theory for computing utility functions needed to solve the problem in Chapter 3, Chapter 4 presents a decision procedure for use in the restricted domain where only complete bundle purchases are made. Next, Chapter 5 discusses the inherent problems in the purchase procedure when partial bundle purchases are permitted and the expected utility of individual item purchases must be evaluated. The QR-tree is proposed as a structure for modeling the system of decisions in such way that is more convenient for solution techniques, and three such techniques which make partial use of Monte Carlo simulation are developed. Chapter 6 presents both theoretical and experimental results, and Chapter 7 concludes by summarizing the contributions of the thesis, and also discusses plans for future work.

Chapter 2

Background

2.1 E-Commerce

2.1.1 History and Background

In 1984, a significant breakthrough was made toward the goal of enabling business transactions to be performed electronically. During that year, the *Electronic Data Interchange* (EDI), which gave computing systems a reliable means for handling a large number of transactions, became an ASC X12 standard. With this technology in place, all that was needed for online commerce to become a reality was a user-friendly platform that would allow the transactions to be created. The first point-and-click Internet web browser, *Mosaic*, was introduced in 1992 (and subsequently *Netscape* in 1994), giving merchants a new medium for displaying and demonstrating their products, and buyers the easy user interface required for locating goods and making purchases. In the next few years the activity of performing transactions electronically over the Internet, eventually termed *electronic commerce* (or e-commerce), became more common. Finally, the largely successful holiday season of 1998 showed online buyers and sellers alike that e-commerce was the way of the future.

The rapid growth witnessed in the volume of online transactions over the last few years is astounding. In 1997, worldwide e-commerce transactions totaled approximately $5 million (US) [FR01]. The year 2000 saw e-commerce hit the $1 trillion mark [FR01], and it is predicted to reach $4.6 trillion in 2005 [GC01]. In Canada, e-commerce reached $24.8 billion (Cdn) in 2000 and is projected to grow to $220.4 billion in 2005. Of this, $187.3 billion (85%) is expected to come from Business-to-Business (B2B) transactions, and the remainder from Business-to-Consumer (B2C) and Consumer-to-Consumer (C2C) transactions [GC01].

With this growth in sales volume comes an equal growth in the need for capable technology. Unfortunately for businesses and consumers who benefit from e-commerce solutions, the technology has not grown fast enough. Much of the e-commerce-related technology produced today is focused on solutions for helping transactions between businesses work more smoothly and efficiently. However, these solutions are typically just the automation of business practices and procedures that are already in place between companies. What about the business looking for new contracts? Or perhaps one in need of a set of one-time purchases for which it is not necessary to establish long-term business relationships, but rather is more important to find the best items at the lowest cost? Even a regular consumer attempting to acquire a set of parts to build a custom car or computer needs help. Locating the desired items can often be very difficult for a buyer. And once the proper websites are found, they are often slow, uninformative, and difficult to navigate. Recent research has shown that 43% of shoppers who visit an e-commerce enabled site intending to make a purchase leave frustrated, buying nothing. These missed potential sales are believed to represent a $11 billion (US) loss [Cre01].

2.1.2 E-Commerce Purchasing Models

Aside from the development of new business models upon which new e-commerce solutions are built, a lot of work is now being put into developing and improving actual *e-commerce* purchasing models. Each of these models basically provides a setting and a strategy for buyers and sellers to meet and perform transactions. Some are static and serve merely as a place to browse available goods and make purchases on selected items, while others are more interactive and involve both the supplying and purchasing parties to work together to ultimately reach a transaction. A few of the more popular models are now discussed [DDN01].

Storefront

The storefront e-commerce model is the one typically used in most B2C e-commerce transactions. In this model, businesses set up a ”store” on their website. Potential buyers can visit their site, view their products, and make purchases on-line. The storefront model is the most widely used in e-commerce, by businesses of all types and sizes.

Portal

If a buyer is in need of a few items of generally the same nature, it is relatively easy to find a site at which most of his or her needs can be satisfied. However if a broad range of items is needed, the onus is on the potential buyer to locate each of the many sites at which the desired goods or services may be found. This search process can be very time consuming. In the portal model, the buyer needs only to put in a request at a portal website. The portal, typically linked to thousands of sites, will search for the requested product and return a list of matching items it found, along with information as to the supplier, price, ordering information, etc.

Request for Quote

In this model, purchasers visit a particular B2B e-commerce enabling site to request a quote for supplies, projects, etc. These sites are typically vertical and product-specific. When a potential buyer submits a request-for-quote (RFQ), a notice is sent to all suppliers of that item who have signed on with the site. A list of current RFQ’s is also available for perusal. Suppliers send their quotes to the buyer, who then has the opportunity to compare quotes and choose a winner. Competitors' quotes are typically not disclosed. If a transaction is made, a fee is paid by the winning supplier to the enabling site.

Auctions

Just as there are traditionally, several variations of the auctioning model are held electronically. Sequential auctions, where items are auctioned one at a time (in sequence) and awarded to the highest bidder (referred to as an English auction), are common. Silent bid auctions (similar to request-for-quote) are also seen. Another type that is quite common in e-commerce is the reverse auction. Reverse auctions work in much the same manner as sequential auctions, except in the opposite direction. Here, a purchaser requests an item from a set of suppliers, and each of the suppliers compete for the lowest sale price. At any given time in a typical reverse auction, each of the suppliers know what each of its competitors has offered (but possibly without knowing who each of the competitors are), and has the right to submit a better offer. Reverse auctions work differently from traditional sequential auctions in that the purchaser might not accept the best price, but perhaps the one it presumes to be the best offer depending on extra incentives, preference of supplier, etc.

2.2 Market Clearing

While the research presented in this thesis is not directly based on any particular aspect of market clearing research, there is a strong relation and thus a few techniques are discussed for completeness. This section focuses on past and present research on techniques for clearing auctions that require more complex item allocation and winning-bid determination strategies. Specifically, the problem of awarding winning bids becomes much more complex when bids are on quantities of goods or on bundles of different goods. Thus the problem of deciding which bids should be awarded in order to achieve maximum income, profit, liquidity, social welfare, etc. is difficult. For auctions that require offline algorithms, where all bids are submitted before the auctioneer decides any winners, the problem is to compute the optimal allocation of the available goods to the bidders. Auctions or other markets that require online algorithms, where bids are being submitted and the market is being cleared continuously, have the extra problem of also deciding *when* clearing should be done.

The problem of determining optimal allocations has been identified in many problem domains, such as in task allocation in multi-agent systems, and airport take-off and landing time slot allocation [RSB82]. The most recent demand for efficient clearing algorithms has surfaced in e-commerce research, as demonstrated by such market server prototypes as *eMediator* [San00] and *AuctionBot* [WWW98]. For generality, the term *item* is used to represent whatever entity for which buy bids and sell bids are placed in these auctions. First, the problem of awarding bids on quantities of indistinguishable items is presented. The problem is then extended to include bids on bundles of different items, followed by a discussion on optimal auction protocols that encourage bidders to bid their true values for items. Finally, a discussion on recent work on online algorithms for continuous auctions is given.

2.2.1 Multiple Indistinguishable Items

This section considers the setting where there are multiple indistinguishable units of a single item (e.g. stock) for sale. The following formalization is attributed to Sandholm and Suri [SS01], and the reader is referred there for a more thorough treatment including algorithms, proofs, and other special cases.

In this auction, each bidder has a certain desire for different quantities of the item for sale. To express this, each bidder submits a piecewise linear demand curve indicating the quantity $q(p)$ he/she is willing to pay at unit price p. If the bid is cleared at price p, the bidder is awarded $q(p)$ units at price $p \cdot q(p)$. The goal of the auctioneer is to consider the demand curve for each bidder, and clear the bids in such a way that will maximize total income. Different strategies for solving this problem are developed for different auction rules, specifically whether or not *discriminatory pricing* and/or *free dispersal* are allowed.

In a discriminatory price auction, a price p_i is set for each bidder i in such a way as to maximize $\sum_i p_i q_i(p_i)$ subject to the supply constraint $\sum_i q_i(p_i) \leq Q$ (where Q is total quantity available). With free disposal, the auctioneer is not required sell all available units (i.e. the remainder is allowed to be disposed for free). Without free disposal, all units must be sold. Sandholm and Suri show that the problem of determining a revenue-maximizing allocation using discriminatory pricing is NP-complete when free disposal is either allowed (*knapsack*) or not allowed (*subset sum*).

In a non-discriminatory price auction, all bidders pay the same unit price. The optimal price p^* must be determined such that $\sum_i p^* q_i(p_i)$ is maximized. Sandholm and Suri give $O(nk \log(nk))$ algorithms (where k is the maximum number of pieces in any bidder's demand curve) for both with and without free dispersal.

2.2.2 Combinatorial Auctions

Quite often, participants in an auction have preferences over bundles of different items. For example, a buyer may be interested in obtaining two items A and B, where both are needed (one or the other by itself would be unacceptable). It would be most beneficial if the buyer could place a bid on both items, thus eliminating the risk of buying one and subsequently finding that the other is unattainable. Or possibly either A or B by itself is acceptable, but the buyer's value for the two of them together is greater than the sum he/she values for them individually. The buyer would certainly like to express this preference in his/her bidding strategy. These needs are satisfied by the *combinatorial auction* mechanism.

The Winner-Determination Problem (WDP)

The combinatorial auction is believed to be first presented by Rassenti *et al.* [RSB82] as a means for auctioning airport take-off and landing time slots to airlines. In the general setting of the combinatorial auction, an auctioneer has a set I of items available for sale, and accepts a set B of bids where each $b \in B$ is a subset of I and has a bid price $p(b)$. The *winner determination problem* (WDP) is the problem faced by the auctioneer when clearing a combinatorial auction, and is defined as follows (as in [SSGL01])

Definition 2.1 The *winner-determination problem* (WDP) for a combinatorial auction is the problem of deciding how to label the bids as winning or losing so as to maximize the auctioneer's revenue under the constraint that each item can be allocated to at most one bidder.

This problem is *NP*-complete [RPH98], and cannot even be approximated to within a bound $k \leq n^{1-\epsilon}, \epsilon > 0$ in polynomial time, where n is the number of available items and k is a constant such that the revenue yielded by the best allocation is guaranteed to be no more than k times that of the best allocation found by the algorithm. However, polynomial time algorithms exist for instances of the WDP in combinatorial auctions with certain restrictions. The following are discussed in [San02]:

1. If the bids have at most w items each, and Δ is the number of bids with which any given bid can share items, a bound $k = 2(w+1)/3$ can be established in $O(nw^2\Delta^w)$ time [CH99].

2. If Δ is the number of bids with which any given bid can share items, a bound $k = \lceil(\Delta+1)/3\rceil$ can be established in linear time by partitioning the set of bids into $\lceil(\Delta+1)/3\rceil$ subsets such that $\Delta \leq 2$ in each subset, and then using dynamic programming to solve the weighted set packing problem in each subset [Hal98]. Other

bounds to be established for this setting are $k = \Delta/2$ [Hoc83] and $k = (\Delta + 2)/3$ [HL97].

3. If the bids can be coloured with c colours such that no two bids that share items have the same colour, then $k = c/2$ can be established [Hoc83].

4. The set of bids have a K-claw if there is some bid that shares items with K other bids that do not share any items with each other. If the bids are free of K-claws, bounds of $k = K - 2 + \epsilon$ and $k = (4K + 2)/5$ can be established in $n^{O(1/\epsilon)}$ and $n^{O(K)}$ time, respectively [Hal98].

5. If D is the largest d such that there is some set of bids in which every bid shares items with at least d bids in that subset, then $k = (D + 1)/2$ can be established in polynomial time [Hoc83].

Tennenholtz [Ten00] also pointed out practical non-trivial instances of combinatorial auctions for which there are polynomial-time solutions, such as combinatorial network auctions and various sub-additive combinatorial auctions.

An Optimal Solution for Winner-Determination

While the WDP is NP-complete, some computationally manageable algorithms have been developed [FLBS99, ATY00, San02, SSGL01]. See also [dVV00] for a survey. Some of these techniques make use of clever programming strategies, while others take advantage of characteristics of typical problem instances. One such solution, attributed to Sandholm [San02], takes advantage of the fact that, while the number of possible item allocations is exponential on the number of items, in practice the space of allocations that need to be checked is actually quite sparse. This solution is now presented.

Consider a combinatorial auction on a set I of items, upon which there is a set of bids B and each bid $b \in B$ has a bid price $p(b)$. The algorithm generates a tree

T (see Figure 2.1) where each path in T consists of a sequence of disjoint bids. Let $B_p \subseteq B$ be the bids on path p, and let $p(B_p)$ be the total of the bid prices. The algorithm adds bids to a path p until all items in I have been used on p. B_p then corresponds to a feasible allocation. Note that since free dispersal is typically allowed in the combinatorial auction, dummy bids may have to be added to B (with bid price 0) for single items for which no actual bids were placed. This is necessary since the optimal allocation may not include every item in I. This continues for all allocations. The best allocation (i.e. the one with the highest $p(B_p)$) found so far is always stored. Once the search completes, that allocation is optimal.

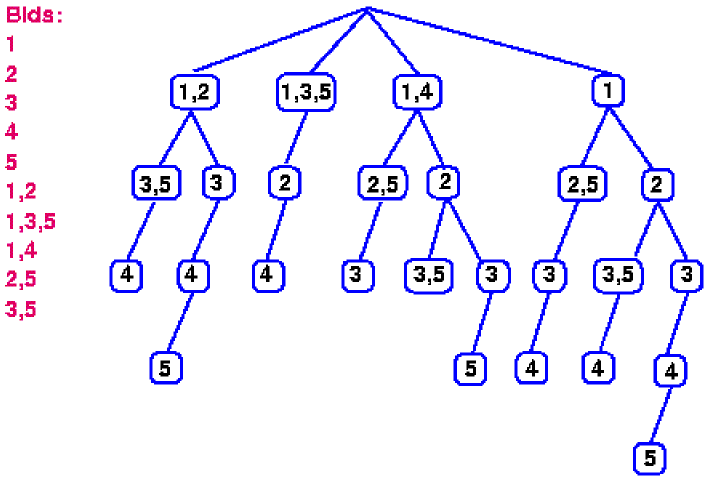


Figure 2.1: Sandholm's search tree for finding optimal allocation.

The naïve method of constructing the tree would include all bids (that do not include items already on the path) as children of each node. This, however, will lead to the consideration of many different orderings of bids representing the same allocation. The branching factor can be significantly reduced by ensuring that each allocation is explored only once. This can be done by assigning an *index* to each item. The tree is then built by generating the children of a node with those bids that

- include the item with the smallest index among items that are not on the path yet, and

- do not include items that are already on the path.

Since the algorithm only generates the populated parts of the search space of allocations, which in practice is typically sparse, it provides a computationally feasible method for solving the WDP. To improve on the performance of the algorithm, Sandholm uses a secondary depth-first search and IDA* [Kor85] with heuristics to find children quickly, Also, several preprocessing techniques such as removing noncompetitive bids and decomposing the problem into parts, are implemented in order to help speed up the overall process.

Many sophisticated algorithms for solving the WDP optimally have been, and continue to be, developed to work much faster than the one presented here. Two examples of front-runners in the field are CPLEX [ATY00] and CABOB [SSGL01]. Layton-Brown *et al.* [LBST00] have developed CAMUS, an algorithm to determine optimal winning bids in *multi-unit* combinatorial auctions. This auction protocol considers the more general setting where there may be more than one unit of each good, and bidders are allowed to bid on bundles consisting of not only several distinct items, but also on quantities of those items.

The problem that remains is that, while the WDP can be solved optimally, it is not necessarily the case that an auctioneer will obtain the maximum possible income. This is because the general combinatorial auction problem itself is not optimal. In this setting, there is not much incentive for bidders to bid their actual worth for a bundle, but rather quite often a lower price (and never higher). An auction protocol that encourages the bidders to bid their full worth on bundles, where the auctioneer can solve the WDP on those bids, would certainly provide an optimal solution. Such protocols are now discussed.

2.2.3 Optimal Combinatorial Auctions

An optimal combinatorial auction is one where bidders, using a reasonable bidding strategy, will bid their actual value for a bundle of items. The idea of an optimal auction protocol was inspired by Vickrey [Vic61], and later generalized to include combinatorial auctions. *Iterative* combinatorial auctions, where prices on bundles are increased incrementally until the highest possible sale prices are achieved, are also known to be optimal. Examples of each of these protocols are now presented.

Generalized Vickrey Auctions

A Vickrey auction is a sealed-bid silent auction where the submitter of the highest bid for an item is the winner. This winner then pays the amount of the second-highest bid. With this mechanism, it is an optimal strategy for each bidder to reveal his/her true value for the item. Consider a Vickrey auction for a single item, where each participant i has a value v_i and places a bid b_i on the item. Let the *payoff* for the winning bidder to be the difference between his/her value and actual price paid for the item. Participant i would therefore have the *expected* payoff

$$Prob(b_i > b_{max}) \cdot (v_i - b_{max})$$

where b_{max} is the highest over all bids other than b_i. If i's value for the item is more than b_{max} (i.e. $v_i > b_{max}$), then he can make the probability of winning equal to 1 by bidding $b_i \geq v_i$. He will then pay b_{max}. Else if $v_i < b_{max}$, then he/she can make the probability of winning 0 by bidding $b_i \leq v_i$. So if $b_i = v_i$ then i will win the auction and pay b_{max} iff his value for the item is at least this amount. Since $Prob(b_i > b_{max})$ is 1 when $v_i - b_{max}$ is positive and 0 when $v_i - b_{max}$ is negative, the expected payoff is maximized. Hence, bidding one's true value for the item in this type of auction is always an optimal strategy.

MacKie-Mason and Varian [MMV95] describe a Generalized Vickrey Auction (GVA), in which bids on more than one type of good, multiple units of goods, and externalities (interests of the bidders other than just their own bundles) are considered. They demonstrate that, even for their extension to the general case, an optimal bidding strategy[1] for the participants is to reveal their true value for desired bundles.

Iterative Combinatorial Auctions

In an iterative combinatorial auction, the idea is to iteratively increase the ask price for each requested bundle through a sequence of rounds, until it is determined any further increase will not increase the total income of any allocation (i.e. the bidders are not willing to go any higher). Note that this works very much like the single-item standard English auction, where the price increased until it is to determined that no bidder will offer anything more. When this final round is reached, the best allocation is found and bids are awarded.

*i*Bundle, attributed to Parkes [Par99, PU00a], is an iterative ascending-price combinatorial auction that is guaranteed to compute optimal bundle allocations with a best-response agent bidding strategy. In particular, *i*Bundle considers agents to be *myopicly rational*, which means that they play a best (utility-maximizing) response to the current ask prices and allocation in the auction. The agents are myopic in the sense that they only consider the current round of the auction. Note that if agents had the foresight to consider future rounds, the auction could be strategically manipulated and therefore become sub-optimal. See [PU00b] for work on preventing this pitfall.

The auction begins with the submission of bids. Bidders are allowed to submit both *OR* bid sets, which contain bids such that winning more than one would be acceptable to the bidder, and *XOR* bid sets, where no more than one win is acceptable. The ask price for each desired bundle is initially set at 0, and is incremented by a pre-set value ϵ after each round. In round t, an agent can submit a bid for a bundle b equal to the ask price of b in round $t-1$ (meaning that it is not interested in

[1] The authors refer to this as a "dominant" bidding strategy.

increasing its bid), or greater than or equal to the new ask price. After all bids are submitted for round t, an algorithm for solving the WDP is used to determine the best allocation. The auction terminates when one of the following occurs:

1) all bidding agents are *happy* in the current winning allocation, or

2) all agents submit the same bids in two consecutive rounds

where an agent is happy iff it wins all bids in its OR bid sets and exactly one bid in each of its XOR bid sets. If either 1) or 2) occur, then it is the case that no agent will ever increase its bids further, and thus the optimal set of bids has been achieved.

2.2.4 Online Market Clearing

In the online market clearing problem, items are bought and sold by multiple buyers and sellers who submit buy and sell bids that arrive and expire at different times. The auctioneer is faced with an online clearing problem of deciding which buy and sell bids to match before seeing all of the bids. Blum *et al.* [BSZ02] study the setting where there is a market for one commodity. Algorithms are also developed to maximize surplus (total profit for the auction) and liquidity (total number of transactions, total buy volume, or total sell volume), and competitive ratios are given for comparison with offline algorithms. Their formalization, as well as two of their algorithms designed to maximize surplus, are presented here.

Temporal Clearing Model

The online market clearing problem faced by an auctioneer where buy and sell bids arrive over time is formalized as follows: When a bid is introduced, the auctioneer learns the bid price and expiration time. A bid that has been introduced but has not yet expired is called *live*. At any point in time, the auctioneer can match a live buy bid with a live sell bid and remove the bids from the system. At such a time, the submitter of the buy bid is committed to buying the item at the buy bid price, and the submitter of the sell bid is committed to selling the item at the sell bid price.

Definition 2.2 (temporal clearing model) A *temporal clearing model* consists of a set B of buy bids and a set S of sell bids. Each bid $v \in B \bigcup S$ has a positive price $p(v)$, is introduced at start time $t_s(v)$, and removed at finish time $t_f(v)$. A bid v is *live* in the interval $[t_s(v), t_f(v)]$. Two bids are said to be *concurrent* if there is some time when they are both live simultaneously.

Definition 2.3 (legal matching) A *legal matching* is a collection of pairs $\{(b_1, s_1), (b_2, s_2), \ldots\}$ of buy and sell bids such that b_j and s_j are concurrent.

Algorithms for Maximizing Surplus

Each matched pair of buy bids and sell bids produces a surplus. This surplus is equal to the buy price minus the sell price (and is not necessarily positive in the general case). The total surplus is the sum of surpluses over all matched pairs, and can be split in any way among the auctioneer, buyers, and sellers.

An offline algorithm receives knowledge of all buy and sell bids up front, while an online algorithm only learns of bids when they are introduced. Both types of algorithms have to produce a legal matching. The performance of an online algorithm can be measured using competitive analysis [BEY98] with the optimal offline solution for a given sequence of bids. Let ALG be an online algorithm whose performance is to be measured where ALG(I) is the surplus achieved by the matching found by ALG on input I, and let OPT be an offline algorithm that computes the matching yielding the optimal result OPT(I). For the purposes of the following algorithms presented here, for a $c \in \Re$ ALG is said to be c-competitive if for any I, $c\cdot$ ALG(I) $\geq$ OPT(I).

A method for maximizing total surplus online is to choose some G to be the minimum surplus required by any matched pair. That is, b and s can only be matched if $p(b) \geq p(s) + G$. In their paper, Blum *et al.* [BSZ02] choose G randomly according to an exponential distribution. Specifically, for all $x \in [1, p_{max} - p_{min}]$, let

$$Prob(G \leq x) = \frac{\ln(x) + 1}{\ln(p_{max} - p_{min}) + 1}$$

where $[p_{min}, p_{max}]$ is the range of possible bid prices. Let $M^*(G)$ be the maximum matching (i.e. the matching with the highest number of matches) given G, $|M^*(G)|$ be the number of matches in $M^*(G)$, and OPT be the optimal achievable surplus. If G is chosen from the above distribution, then

$$E[G \cdot |M^*(G)|] \geq \frac{OPT}{\ln(p_{max} - p_{min}) + 1}$$

where $E[G \cdot |M^*(G)|]$ is the expected minimum surplus. See [BSZ02] for the proof.

Following are two online matching algorithms. The first, which works as a greedy algorithm, corresponds to the strategy "whenever there exists a pair of bids that would produce a surplus of at least G, match them immediately".

Algorithm 2.1 (Greedy) When a buy (sell) bid v_1 is introduced, if there exists a live sell (buy) bid v_2 such that matching v_1 and v_2 would provide a surplus greater or equal to G, then match them.

The Greedy algorithm is $2(ln(p_{max} - p_{min}) + 1)$-competitive [BSZ02]. Note that this algorithm does not need to consider the expiry times of bids.

The second algorithm is a bit more sophisticated, and thus provides a better result. It works by keeping a perfect matching throughout. Each time an unmatched bid is about to expire, it checks to see if it and some other unmatched bid can be added to the matching (not necessarily being matched together).

Algorithm 2.2 Let W be the bids in the current matching. When a bid v is about to expire, consider all live unmatched bids sorted by expiration time. Starting with the bid v' that starts the soonest, see if there exists a perfect matching on $W \bigcup \{v, v'\}$ (making sure each match achieves a surplus $\geq G$). Add the pair $\{v, v'\}$ to W where v' is the first such bid that works. If no such v' exists, allow v to expire unmatched.

Algorithm 2.2 is $(ln(p_{max} - p_{min}) + 1)$-competitive [BSZ02]. Note that matching v and v' does not need to achieve a surplus of at least G (and in the general case does not even need to achieve a positive surplus at all), but it must be the case that a perfect matching exists on $W \bigcup \{v, v'\}$ where every match achieves a surplus of at least G. So algorithm 2.2 achieves a surplus of at least $G|M^*(G)|$.

Blum *et al.* also develop algorithms that attempt to maximize liquidity of the market in terms of the number of transactions, the total buy volume, and the total sell volume. This, while not as immediately beneficial to the common good as surplus maximization, can help the long-term fortunes of the market. The success of a market is often assessed in terms of liquidity, as buyers want to be able to buy and sellers want to be able to sell. So liquidity maximization is quite valid. The reader is referred to [BSZ02] for a further discussion.

2.3 Utility Theory

2.3.1 Introduction

The concept of utility is believed to have originated in work by Daniel Bernoulli [Ber38, Ber54], when he solved the famous St. Petersburg Paradox posed in 1713 by his cousin Nicholas Bernoulli. The paradox posed the following situation: a fair coin is tossed until the first "heads" appears. If this happens on the n^{th} toss, the payoff is 2^n ducats. How much should one pay to play this game? The idea was that since the expected payoff is infinite, a gambler should be willing to pay an infinite sum to play. Bernoulli demonstrated that the expected payoff of a gamble is not necessarily the same as one's *expected utility* for the gamble. Typically, one's marginal utility for money decreases as the amount increases. This is known as *diminishing marginal utility.* Expected utility, rather than expected payout, was shown to be the true measure of how much a gambler would be willing to pay. After 200 years, the subject matured with the classic work on maximizing expected utility by von Neumann and Morgenstern [vNM47]. This theory had seminal influence on the area of decision under risk, and encouraged work in the more general area of decision making under uncertainty. Two of the great works in this discipline were that of Wald [Wal50] and Savage [Sav54]. Some of the more recent treatments on utility theory are those of Fishburn [Fis70], Keeney and Raiffa [KR76], and Kreps [Kre88], upon which the presentation given here is largely based. Much of the historical background comes from [BEY98]

Utility theory is used as means for taking a person's judgments of preference, worth, usefulness, etc. and enabling them to be represented in numerically useful ways [Fis68]. This numeric representation allows decision-makers, whether human or automated, to make indisputable and objective choices from a set of given alternatives. There are two main schools of thought when it comes to utility theory: descriptive theory and normative theory. Descriptive utility theory is the study of human choice behaviour. It consists mostly of analysis of how people make choices, and what causes them to behave in certain ways. This type of utility theory plays a large role in economics.

Normative utility theory is the study of how people *should* behave and make decisions. It focuses on helping the decision-maker to act in a rational and logically consistent manner. A third category is discussed by Bell *et al.* [BRT88], referred to as *prescriptive* utility theory. According to them, this theory focuses on *how to* help people make rational decisions. However, this is not widely accepted. Most agree upon the usual dichotomy, and either place these prescriptive theories inside the category of normative theory, or consider the terms normative and prescriptive to be interchangeable and thus to represent the same theory.

2.3.2 Preference Orders and Utility Functions

The concept of *utility* can be considered by a decision-maker hoping to act rationally when faced with the problem of choosing from a set of *alternatives.* Each alternative has one or more *attributes* that need to be considered when assessing one's preferences for alternatives. More formally, a utility theory on a set X of comparable alternatives is based on a binary preference relation $\succeq$ (read as "is at least as preferable to") on X such that for all $x, y \in X$, exactly one of $x \succeq y$ or *not* $x \succeq y$ is true. Typically the relations of strict preference $\succ$ and indifference $\sim$, defined from $\succeq$, are used:

$$x \succ y \text{ means that } x \succeq y \text{ and } \textit{not } y \succeq x$$
$$x \sim y \text{ means that } x \succeq y \text{ and } y \succeq x$$

A utility theory is a set of consistent assumptions and theorems on X and $\succeq$. One such assumption that appears in most utility theories (including that used in this thesis) is that of transitivity:

$$\text{if } x \succeq y \text{ and } y \succeq z \text{ then } x \succeq z$$

One such theorem that is often used when incorporating utility theory into decision making assigns a numeric utility to each element of X. Specifically, there exists a function $u : X \rightarrow \Re$ such that:

$$\forall x, y \in X, x \succeq y \Leftrightarrow u(x) \geq u(y) \tag{2.1}$$

Each alternative in X may be comprised of several *attributes.* In this case, $u(x)$ can be computed as a function of the utilities of the attribute values that describe x. If the decision-maker can accurately assess his/her utility function over all alternatives, then it follows that the preference relation for all pairs of alternatives will be known. Before such an assessment can be performed, the notion of expected utility must be introduced.

2.3.3 The von Neumann-Morgenstern Theory of Expected Utility

The discussion of preference theory presented in the previous section introduced the theory of *decision under certainty.* That is, whichever alternative is chosen, the decision-maker is certain of the consequence of that decision (i.e. the associated attribute(s) of the alternative). *Expected utility* is used in the theory of *decision under risk.* As before, this decision problem consists of a set of alternatives X from which the decision-maker must choose. The problem differs in that, given the set of possible consequences Z, for each alternative $x \in X$, each $z \in Z$ has some likelihood of occurring if x is chosen. Expected utility theory takes into account the decision-maker's preferences for all consequences of alternatives, as well as the probabilities of these consequences occurring, and computes the expected utility for each alternative. The von Neumann-Morgenstern (NM) model views this uncertainty as objective, in the sense that there is a quantification of how likely the various outcomes are. This is given in the form of a probability distribution, which in the utility theory literature is most commonly referred to as a *probability measure.*

Probability Measures

Let Z be a finite set of consequences for a given alternative. A *probability measure* p on Z is a function $p : Z \to [0, 1]$ such that $\sum_{z \in Z} p(z) = 1$. Consider the structure presented in Figure 2.2. This is an example of a probability measure (often referred to as a gamble or lottery in the NM setup) on the consequence set $Z = \{\$0, \$400, \$1000\}$, where $p(0) = .1$, $p(400) = .7$, and $p(1000) = .2$. This means that, if the decision-maker chooses this lottery, he/she has a .1 chance of receiving \$0, a .7 chance of receiving \$400, and a .2 chance of receiving \$1000.

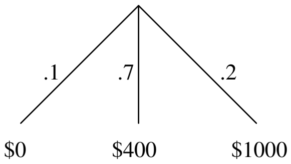


Figure 2.2: An example probability measure.

The NM Axioms

Let Z be a finite set of consequences and P be the set of probability measures on Z. P is the set of alternatives to be considered, referred to as the *choice set*. The following three axioms are necessary and sufficient for the NM theory of expected utility:

Axiom 2.1 $\succ$ is a preference relation.

Axiom 2.2 (substitution axiom) For all $p, q, r \in P$, and all $a \in (0, 1]$, $p \succ q$ implies $ap + (1 - a)r \succ aq + (1 - a)r$.

Axiom 2.3 (Archimedean axiom) For all $p, q, r \in P$, if $p \succ q \succ r$ then there exist $a, b \in (0, 1)$ such that $ap + (1 - a)r \succ q \succ bp + (1 - b)r$.

These axioms yield the following result:

Theorem 2.1 A binary relation $\succ$ on P satisfies the above axioms iff there exists a function $u : Z \rightarrow R$ such that

$$p \succ q \Leftrightarrow \sum_{z \in Z} u(z)p(z) > \sum_{z \in Z} u(z)q(z)$$

Refer to [Kre88] for a proof. This theorem leads to the notion of expected utility. Let L be a lottery whose outcome is selected by a probability measure p on a set of consequences Z. Let $\tilde{z}$ represent the uncertain outcome of L. The *expected utility* of L is

$$E[u(\tilde{z})] = \sum_{z \in Z} p(z)u(z) \tag{2.2}$$

which, by Theorem 2.1, is an appropriate index to maximize in choosing among lotteries.

Example 2.1 Consider a choice between the two lotteries given in Figure 2.3. Also, assume the decision-maker has utilities as follows: $u(\$50) = .1$, $u(\$400) = .5$, $u(\$600) = .7$, $u(\$1000) = 1$. The expected utility for lottery A is $.6(.5) + .4(.7) = .58$, and for lottery B is $.5(.1) + .5(1) = .55$. Therefore, this particular decision-maker should choose A since it has the higher expected utility.

Example 2.2 Consider again the choice between the two lotteries given in Figure 2.3. Assume this time that the decision-maker has utilities $u(\$50) = .1$, $u(\$400) = .2$, $u(\$600) = .7$, $u(\$1000) = 1$. In this case, the expected utility for lottery A is $.6(.2) + .4(.7) = .4$ and for lottery B is $.5(.1) + .5(1) = .55$, so this decision-maker should choose B.

A
.6
.4
$400
$600
B
.5
.5
$50
$1000

Figure 2.3: If lottery A is chosen, the decision-maker has a .6 chance of winning $400 and a .4 chance of winning $600, while lottery B gives a .5 chance of winning $50 and a .5 chance of winning $1000.

2.3.4 Assessing Utility Functions

Let Z be a set of consequences and P be a set of alternatives, which are probability measures on Z. In order to determine the expected utility for each $p \in P$, the utility of each consequence $z \in Z$ must first be assessed. There are two basic approaches to this problem: one for when Z is small (i.e. perhaps less than 50 [KR76]) and another for when Z is large and there is a natural ordering of the z's (e.g. a range of integers like amounts of money). These two methods are now presented.

Direct Assessment of Utilities for Consequences

With direct assessment, each $z \in Z$ is analyzed and assigned a utility on a case-by-case basis. First, the scale must be set. Since utility is relative and not absolute, utility can arbitrarily be assigned to two of the alternatives and the utilities of all others can be assessed relative to those two. This is typically done in the following way. Let z^0 and z^* be least and most preferred consequences, respectively, as determined by the decision-maker. Assign the utilities

$$u(z^0) = 0 \qquad\qquad u(z^*) = 1$$

The utility for all other consequences can now be set in relation to these two (or any other two consequences with established utilities) with use of certainty equivalence.

Definition 2.4 (Certainty Equivalent) A *certainty equivalent* of a lottery L with uncertain consequence $\tilde{z}$ is a consequence $\hat{z}$ such that the decision-maker is indifferent between L and the consequence $\hat{z}$ for certain. That is, $u(\hat{z}) = E[u(\tilde{z})]$.

The utility of any consequence z_i can be computed in relation to any two consequences z_j and z_k with known utilities $u(z_j)$ and $u(z_k)$ as follows: A probability π must be determined such that z_i is a certainty equivalent to the lottery that gives a π chance of getting z_j and a $1-\pi$ chance of getting z_k. That is, π is the probability such that the decision-maker is indifferent between z_i for certain and the lottery in Figure 2.4. By the definition of certainty equivalent and by equation 2.2,

$$u(z_i) = \pi u(z_j) + (1-\pi)u(z_k) \tag{2.3}$$

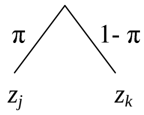


Figure 2.4: Lottery used to determine π.

Accurately determining these π values can be a difficult task for a typical decision-maker. And so, because of likely errors, several consistency checks will have to be made by comparing a consequence with many different lotteries. For large numbers of consequences, where there is some underlying ordering of the consequences, there is a much more efficient method. This method allows the decision-maker to determine all utilities without considering every consequence individually.

Example 2.3 Let $z_1, z_2, z_3 \in Z$ be consequences such that $u(z_1) = .4$ and $u(z_2) = .5$. If the decision-maker is indifferent between a lottery that yields .7 and .3 probabilities of getting z_1 and z_2 respectively, and getting z_3 for certain, then $u(z_3) = .7(.4) + .3(.5) = .43$.

Utility Functions for Real-Valued Consequences

When the set of consequences are associated with a numeric scale, an efficient method for determining the utilities over Z is to assess the utilities for a few values, plot these utilities on the plane, and fit a curve. In this way, the *utility function* can be determined. This function can subsequently be used to compute the utility for any $z \in Z$. This method is commonly used when the consequences are losses or gains of monetary values. Such consequences are common in many applications of utility theory.

As before, let z^0 and z^* be the least and most preferred consequences, respectively, with $u(z^0) = 0$ and $u(z^*) = 1$. If the consequences were, for example, different amounts of money to be won, z^0 would then represent the smallest amount that could be won and z^* the largest. On the plane, the x-axis is labeled by the consequences and the y-axis by the utilities. All other points are found by determining the decision-maker's certainty equivalents for 50-50 lotteries, as follows:

- Let $\hat{z}_1$ be the certainty equivalent for the 50-50 lottery between receiving z^0 and z^*. Since $u(z^0) = 0$ and $u(z^*) = 1$, then by equation 2.3, $u(\hat{z}_1) = .5(0) + .5(1) = .5$
- Let $\hat{z}_2$ be the certainty equivalent for the 50-50 lottery between receiving z^0 and $\hat{z}_1$. Since $u(z^0) = 0$ and $u(\hat{z}_1) = .5$, $u(\hat{z}_2) = .5(0) + .5(.5) = .25$
- Let $\hat{z}_3$ be the certainty equivalent for the 50-50 lottery between receiving $\hat{z}_1$ and z^*. Since $u(\hat{z}_1) = .5$ and $u(z^*) = 1$, $u(\hat{z}_3) = .5(.5) + .5(1) = .75$

The process of determining certainty equivalents for 50-50 lotteries between values for which the utilities are known continues until a suitable number of points are found. This is known as the *quantitative* assessment procedure. To make the process of accurately determining a utility function much easier, the *qualitative* characteristics must also be assessed. These include whether the function monotonically increases or decreases, as well as a characterization of the decision-maker's attitude towards risk.

Monotonicity

A monotonic function is one that either never increases or never decreases. Monotonicity is a reasonable assumption when dealing with utility functions for money. When the consequences consist of monetary gains, most (if not all) decision-maker always prefer more to less, and thus their utility function monotonically increases. Conversely, if consequences are losses of money (e.g. spending), then the utility function is typically monotonically decreasing. Note that a decreasing function can easily be transformed to an increasing function by changing the interpretation of the consequences. For example, if the consequences are "amounts of money spent" z, and the highest amount that one could possibly spend is \$1000, then the consequences could be changed to represent "amounts of money saved" $1000 - z$.

Nonmonotonic utility functions are not completely uncommon. Consider, for example, a human's utility function for internal body temperature. One would expect this utility to be maximized at about $u(37)$ (degrees Celsius). Depending on the goals of the decision-maker, it is possible that even a utility function for money could be nonmonotonic (e.g. earning less money keeps one in a lower tax bracket). However, in this thesis the assumption of monotonicity will be maintained for all monetary utility functions. That is, winning more money is always better and spending more money is always worse.

Risk Aversion

Recall Example 2.1 where the decision-maker is faced with the choice of either a lottery that gives a .6 chance of winning \$400 and a .4 chance of winning \$600 (A), or a 50-50 lottery between winning \$50 and \$1000 (B). Taking into account the decision-maker's utility for the relevant amounts of money involved, it was determined that lottery A should be chosen. This may seem like the wrong choice at first, since the expected outcome of A is only \$480 while the expected outcome for B is \$525. However, this is just an example of *risk aversion*. This is the result of a phenomenon known as *diminishing marginal utility*, which is quite common in decision-making. The idea of diminishing marginal utility, as used in economics, represents the fact that typically when one has a small number of units of some good, receiving another unit increases the consumer's satisfaction (commonly measured in *utiles*) more than if he/she had a large number of units. So, small numbers of units carry more weight per unit than large numbers when determining the consumer's satisfaction. So in this case, the negative prospect of receiving only \$50 (instead of either \$400 or \$600) carries more weight than the positive prospect of winning \$1000. In other words, the decision-maker is willing to take a loss in expected winnings of \$45 in exchange for eliminating the possibility of only winning \$50. Note that while risk is a concept independent of utility, it plays a role when determining the expected utility of an alternative for which there are uncertain consequences.

Characterizations of a decision-maker's attitude towards risk are now formally given.

Definition 2.5 (risk averse) A decision-maker is *risk averse* if, for any lottery L with expected outcome $E(\tilde{z})$, he/she prefers $E(\tilde{z})$ for certain to L. That is,

$$u[E(\tilde{z})] > E[u(\tilde{z})]$$

Definition 2.6 (risk neutral) A decision-maker is *risk neutral* if, for any lottery L with expected outcome $E(\tilde{z})$, he/she indifferent between $E(\tilde{z})$ for certain and L. That is,

$$u[E(\tilde{z})] = E[u(\tilde{z})]$$

Definition 2.7 (risk seeking) A decision-maker is *risk seeking* if, for any lottery L with expected outcome $E(\tilde{z})$, he/she prefers L to $E(\tilde{z})$ for certain. That is,

$$u[E(\tilde{z})] < E[u(\tilde{z})]$$

A decision-maker's attitude towards risk can give an indication of the shape of his/her utility function, as given in the following theorem (proven in [KR76]).

Theorem 2.2 A decision-maker is risk averse (neutral, seeking) iff his/her utility function is concave (linear, convex).

Note that Theorem 2.2 holds for both monotonically increasing and decreasing functions. For example, Figure 2.5 depicts an example of a (a) monotonically increasing and a (b) monotonically decreasing concave utility function for a risk averse decision-maker. Also note that knowledge of whether one's risk aversion or lack thereof increases or decreases can also help in determining the shape of the utility function.

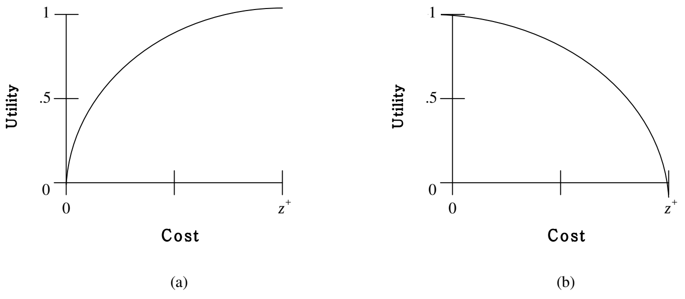


Figure 2.5: (a) Increasing and (b) decreasing concave utility functions for a risk averse decision-maker.

Once these qualitative characteristics have been assessed, finding a suitable utility function for a decision-maker can be reduced to the problem of finding a parametric family of functions possessing the relevant qualitative characteristics. Then, using the quantitative assessments (i.e. points found by using certainty equivalents of 50-50 lotteries), the specific member of the family appropriate for the decision-maker can be selected. See Meyer and Pratt [MP68] for work on effectively determining functions that satisfy the qualitative and quantitative restrictions simultaneously.

2.3.5 Multi-attribute Utility

This section gives some background for assessing the utility of consequences for which there is more than one attribute. In particular, methods for combining the utilities of the individual attributes to give an overall utility are discussed. The convenient situation for this combining occurs when the attributes in question are *utility independent* of each other. This allows the decision-maker to consider the utility of each attribute individually. Most of the discussion here will focus on utility independent attributes, but a short illustration of techniques to consider when no independence properties hold is also provided.

Presentation and Notation

In general, any number of attributes can be associated with any consequence. Here, only the two-attribute case will be specifically treated. This is for three reasons: 1) two-attribute utility theory is easily generalized for a larger number of attributes, 2) the two-attribute case is simpler for explanation and comprehension, and 3) only two attributes are used in the application of utility theory in this thesis. Consider the following notation to be used. Let X be the set of alternatives and $Y \times Z$ the set of consequences, where Y and Z are attributes and each $y \in Y$ and $z \in Z$ are specific instances of those attributes. The decision-maker has single-attribute utility functions $u_y : Y \rightarrow \Re$ and $u_z : Z \rightarrow \Re$. The goal is to determine the two-attribute utility function $u : Y \times Z \rightarrow \Re$, which, if some independence property holds and some additivity assumption is made, is a function of $u_y(y)$ and $u_z(z)$. These properties and assumptions are discussed here.

Utility Independence

One of the most fundamental concepts of multi-attribute utility theory is that of utility independence.

Definition 2.8 (utility independence) An attribute Y is *utility independent* of an attribute Z iff the conditional preferences for lotteries on Y given some $z \in Z$ do not depend on the level of z.

For a more intuitive feel for utility independence, consider the lottery and its certainty equivalent given in Figure 2.6(a). Note that z is constant throughout. If it is determined that the decision-maker's certainty equivalent $\hat{y}$ would not shift if the attribute Z was held constant at a different value, say z' (as in Figure 2.6(b)), then Y is utility independent of Z. Note that this does not imply that Z is utility independent of Y. A similar test would have to be carried out in order to make this determination. If Y and Z are each found to be utility independent of each other, then they are said to be *mutually utility independent.*

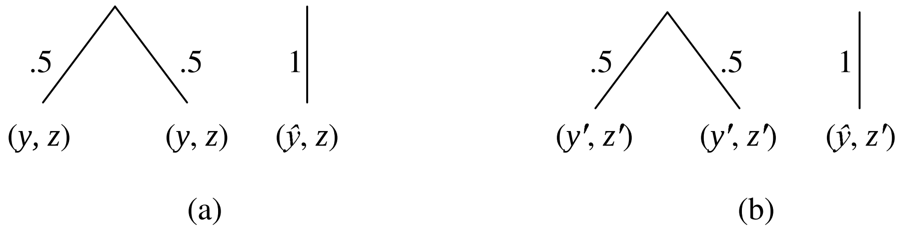


Figure 2.6: Demonstration of Y's utility independence with respect to Z.

When mutual utility independence is established, and an additivity assumption is made, the two-attribute utility function $u(y, z)$ can be completely assessed in terms of the single-attribute utility functions $u_y(y)$ and $u_z(z)$.

The Additive Utility Function

The simplest of the two-attribute utility functions is the additive utility function, which is based on the assumption of *additive independence.*

Definition 2.9 (Additive Independence (1)) Attributes Y and Z are *additive independent* if the paired preference comparison of any two lotteries, defined by two joint probability distributions on $Y \times Z$, depends only on their marginal probability distributions.

The above definition is written in a form that can easily be generalized for more than two attributes. A more intuitive feel for additive independence of two attributes is given in the next definition.

Definition 2.10 (Additive Independence (2)) Two attributes Y and Z are *additive independent* if, for any values y, y' for Y and z, z' for Z, the two lotteries given in Figure 2.7 are equally preferable.

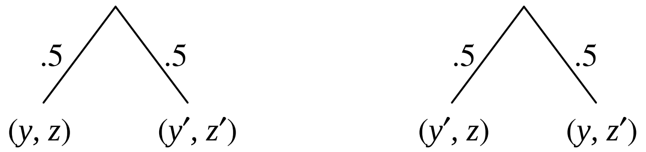


Figure 2.7: Indifference implies additive independence between Y and Z.

This means that, when faced with a 50-50 lottery for receiving one of two Y values together with a 50-50 lottery for receiving one of two Z values, the decision-maker is indifferent as to how the Y and Z values might be paired together. The following theorem is due to Fishburn [Fis65]. The reader is referred there for the proof. An excellent presentation of additive utility including this theorem and proof can also be found in [KR76].

Theorem 2.3 Attributes Y and Z are *additive independent* iff the two-attribute utility function is additive. The additive form can be written as

$$u(y, z) = k_y u_y(y) + k_z u_z(z) \tag{2.4}$$

where k_y and k_z are positive scaling constants.

The scaling constants are necessary since u_y and u_z are individually scaled from 0 to 1, but not in relation to each other. One attribute might be more important than the other, for example, and would therefore need more weight. There are various methods for determining the scaling constants for a utility function. Consider the following utilities that are initially set:

$$\begin{array}{lll} u(y^0, z^0) = 0, & u_y(y^0) = 0, & u_z(z^0) = 0 \\ u(y^*, z^*) = 1, & u_y(y^*) = 1, & u_z(z^*) = 1 \end{array}$$

Note that, for consistency, it must be the case that $k_y + k_z = 1$. By equation 2.4,

$$u(y^*, z^0) = k_y u(y^*) + k_z u(z^0) = k_y(1) + k_z(0) = k_y \tag{2.5}$$

The decision-maker can determine $u(y^*, z^0)$ in relation to $u(y^0, z^0)$ and $u(y^*, z^*)$ using the method of direct assessment as described in section 2.3.4, yielding the value for k_y. Similarly, k_z can be determined by finding $u(y^0, z^*)$. For consistency, $k_y + k_z = 1$ can be checked, perhaps causing the decision-maker to make a minor adjustment.

An easier method can be used if there exist values y, y', z, z' such that the decision-maker is indifferent between getting y with z and getting y' with z'. This implies that $u(y, z) = u(y', z')$, and gives a system of two equations with two unknowns (k_y and k_z):

$$\begin{aligned} k_y u_y(y) + k_z u_z(z) &= k_y u_y(y') + k_z u_z(z') \\ k_y + k_z &= 1 \end{aligned} \tag{2.6}$$

which allows k_y and k_z to be solved.

The Bilinear Utility Function

While additive independence implies mutual utility independence, the converse is not true (see [KR76, page 230] for an example). If two mutually utility independent attributes cannot be shown to be additive independent, the bilinear representation may have to be used, as given in the following theorem (proven in [KR76]).

Theorem 2.4 If two attributes Y and Z are mutually utility independent, then the two-attribute utility function is bilinear. The bilinear form can be written as

$$u(y, z) = k_y u_y(y) + k_z u_z(z) + k_{yz} u_y(y) u_z(z) \tag{2.7}$$

where k_y, k_z and k_{yz} are scaling constants and k_y and k_z are positive.

These scaling constants can be determined in a manner similar to that used for the additive utility function, except that

$$k_y + k_z + k_{yz} = 1 \tag{2.8}$$

As before, k_y and k_z can be determined by assessing $u(y, z^0)$ and $u(y^0, z)$ respectively, since

$$\begin{aligned} u(y^*, z^0) &= k_y u_y(y^*) + k_z u_z(z^0) + k_{yz} u_y(y^*) u_z(z^0) = k_y(1) + k_z(0) + k_{yz}(1)(0) = k_y \\ u(y^0, z^*) &= k_y u_y(y^0) + k_z u_z(z^*) + k_{yz} u_y(y^0) u_z(z^*) = k_y(0) + k_z(1) + k_{yz}(0)(1) = k_z \end{aligned} \tag{2.9}$$

Finally, by equation 2.8, k_{yz} is calculated by

$$k_{yz} = 1 - (k_y + k_z) \tag{2.10}$$

Dependent Attributes

A decision-maker's utility for some attribute may depend on how much he/she receives of some other attribute. When this is the case, computing the additive utility of the attributes is not so straightforward. For example, consider a farmer who has utilities for various amounts of rain and various amounts of sun. His/her utility for an amount of one depends on the amount of the other, since a lot of sunshine would typically increase the utility for large amounts of rain. Keeney and Raiffa [KR76] suggest some ideas for overcoming these problems. A few of these are now briefly discussed.

Transformation of attributes. It may be possible to select an alternative set of attributes, or to slightly change the meaning behind the attributes, to describe the consequences.

Direct assessment. This is done in the manner described in section 2.3.4. Each pair (y, z) is assessed individually. This requires many comparisons and many more consistency checks, and is therefore only feasible if the number of pairs is small (less than about 50).

Dividing $Y \times Z$ into subsets. The idea here is to break the consequence space into parts such that, in each part, the attributes act independently. Consider again the farmer example. For low levels of sun and rain, since the farmer requires higher levels of each no matter what, his/her utilities for sun and rain likely increase as the levels increase, regardless what is happening to the other.

There is no formal method for combining dependent attributes into an overall utility. Each case must be analyzed and solved on an individual basis. As for the use of utility theory in this thesis, mutual utility independence is a reasonable assumption.

2.3.6 Savage's Expected Utility Theory

To this point, all discussion on expected utility theory has assumed the existence of objective probabilities for consequences. This is, however, not always sufficient. What if the decision-maker must make a choice between two alternatives for which the outcomes are uncertain? In this problem, there are no convenient probability measures over the consequences to help with the decision-making, yet a choice must still be made. The decision-maker, in this case, must assess his/her *beliefs* of what might happen, and *subjectively* decide upon the probabilities. While these subjective probabilities do not arise in the application of utility theory in this thesis, a brief presentation of a model for this theory is included here strictly for the sake of completeness.

Savage's theory of expected utility [Sav54] is referred to by many as the crowning achievement in utility theory. It is used as a means for determining a decision-maker's utility for choices for which the probabilities of consequences are purely subjective.

Consider, for example, a family outing. You are uncertain of the weather for the day (it may be sunny, rainy, hot, cool, etc.), but you must make decisions on how to prepare for the trip (how to dress, whether to bring umbrellas, picnic blanket, etc.). Obviously, the likelihood of the various weather activities, along with utilities for the different family activities, must be assessed in order to make the best decision. Savage's model works for this.

Let Z be a set of consequences, S be a set of *states*, and F a set of *acts*. Each state $s \in S$ is a compilation of all characteristics and factors that are relevant to the consequences that will ensue from some choice. In the family outing example, a possible state could be *rain*. The set F is the choice set (i.e. alternatives). Each act $f \in F$ is a function from $S \rightarrow Z$. In this model, the problem faced by the decision-maker is to choose an act f without knowing which state s will prevail, resulting in some consequence $f(s)$ with utility $u(f(s))$. For example, if f_1 is the act of wearing shorts and taking a picnic blanket, then $u(f_1(\mathit{rain}))$ would probably be quite low, and $u(f_1(\mathit{sun}))$ would probably be quite high.

To make a decision, the decision-maker's expected utility for each act in F must be determined. Clearly, this is dependent on his/her beliefs about the weather. Formally, an decision-maker prefers an act f to an act f' iff

$$\sum_{s \in S} p(s)u(f(s)) > \sum_{s \in S} p(s)u(f'(s)) \tag{2.11}$$

where the probability function $p : S \rightarrow [0, 1]$ represents the decision-maker's beliefs as to the likelihood of each state occurring. The probability function is determined by assessing the decision-maker's *preferences*. For example, consider the acts (shown in a manner similar to that used earlier for lotteries) given in Figure 2.8 between which the decision-maker must choose. Let $a \subseteq S$ and $b \subseteq S$ be *events*, and let $a^c = S \setminus a$ and $b^c = S \setminus b$ be the *complements* of a and b. Note that, in this example, the possible consequences are the same for each act. If the decision-maker chooses act f, then that must mean that he/she *judges a to be more likely than b*, since taking a chance on a occurring (and thus yielding the better consequence) is preferred to taking a chance on b occurring (since winning \$1000 is better than \$0).

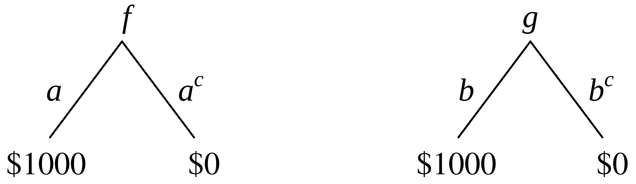


Figure 2.8: Preference for acts determines subjective probability.

Savage provides a set of axioms and theorems that show how the probability function p is uniquely defined by the decision-maker's preferences. The reader is referred to Savage for a more thorough treatment. Also see [Fis70] and [Kre88] for work on Savage's theory as well as work by Anscombe and Aumann [AA63], to whom the seminal work on using both subjective and objective probabilities together is attributed.

Chapter 3

Online Combinatorial Purchasing

3.1 Introduction

3.1.1 Motivation

Consider the problem faced by a single potential buyer who has a number of needs that must be fulfilled, perhaps to complete some project, plan a vacation, etc. All purchasing needs must be satisfied (partial fulfillment is unacceptable). The problem is that there may be more than one way to fulfill these needs. That is, there may be several different (not necessarily mutually exclusive) *bundles* of items such that each would individually satisfy the needs of the buyer. Some bundles may consist of more preferable items than others, and most likely would incur differing total costs if purchased. The task of the buyer is to consider his/her utility for each bundle of items, as well as his/her utility for spending the various amounts of money, and make rational decisions in hope of achieving a bundle purchase yielding maximum overall utility. Making this problem more difficult is the fact that cost and availability of items fluctuate over time. This problem, referred to as the *online combinatorial purchasing problem* (OCPP), resembles the real-life situation of a typical buyer in need of purchasing a set of items. Often, a buyer does not need to procure all items immediately, but might take days or even weeks to explore the market, comparison shop, wait for sales, etc. During this time, some products are added to the market, some are removed, and prices change. In this model, the buyer needs to not only consider the possible bundle purchases available at the current time, but also possible future bundle purchases. That is, at any given time, the buyer must answer the following question: Should I take the best bundle purchase available now, or do I have sufficient reason to believe that there will be a better bundle purchase available later? Note that the buyer must consider the possibility that the good bundle purchase(s) available now may become unavailable later.

To solve this problem, the buyer needs to have some information on not only what is available at the current time and at what price, but also some, albeit incomplete or probabilistic, information about the future. Specifically, it is assumed that at any given time, if the buyer knows that an item i will become available in the future, then he/she also 1) knows when i will become available, and 2) has an idea of what the cost of i will be, in the form of a probability distribution. Calculating the buyer's utility for bundle purchases that include these "future" items requires the use of expected utility theory. That is, in order to make a decision at some time point, the buyer must compare the utilities of currently available bundle purchases with the expected utilities of future bundle purchases.

This chapter gives a description of the tools needed to make decisions in the OCPP. After giving a formal definition of the OCPP, the Price-Quote-Rescind protocol is described as a set of message-passing rules for information exchange between a buyer and a seller, defining when information about items will become known to the buyer. The application of utility theory to the problem is then demonstrated by providing a technique for determining the utility of a bundle purchase. Finally, a discussion on expected utility provides some insight into how preferences for future possibilities are assessed, and also lays the groundwork for when and how purchasing decisions should be made.

3.1.2 Problem Formalization

Let I be a set of items and $\mathcal{B} \subseteq 2^I$ be a set of *bundles* of items in I. At any given time, let I contain only those items that are known to be available either currently or during some definite future time period. I and $\mathcal{B}$ may therefore change over time as new availabilities arise and others pass by. Each $i \in I$ has a quoted cost $c(i)$ and each $b \in \mathcal{B}$ has cost $c(b)$ equal to the sum of its item costs. Note that when two instances of the same item are offered by two different suppliers, or by the same supplier but as part of two different offers, they are treated as two different items in the present framework (although they may be the same from the buyer's point of view). If an item i is currently available, then assume the buyer knows $c(i)$. Otherwise, the buyer has a probability measure $p : Z \rightarrow \Re$ on the outcome of the cost of i, where Z is the set of monetary units. This could be any discrete or continuous distribution obtained from market history, from the supplier directly, a third party, or even subjectively decided upon by the buyer. The goal in the OCPP is to make decisions that maximize expected utility, ultimately giving the buyer the greatest chance of purchasing the $b \in \mathcal{B}$ that is most preferable in terms of b and $c(b)$.

3.2 The PQR Protocol

The Prequote-Quote-Rescind (PQR) protocol is a message-passing protocol for information exchange between a supplier and a purchaser for probabilistic and temporal information. It defines when information will become known by the purchaser about items such as cost, the distribution of possible outcomes on cost, the time a quote will be offered, and the time a quote will be terminated. This information can then be used when planning purchases.

Let $[t_0, t_n] \subseteq \Re$ be the period of time during which a buyer needs to purchase some bundle b of items I, and let $t_p : I \rightarrow \Re$, $t_q : I \rightarrow \Re$ and $t_r : I \rightarrow \Re$ assign time points to items $i \in I$ such that $t_0 \leq t_p(i) \leq t_q(i) < t_r(i) \leq t_n$.

Definition 3.1 (quote time) The *quote time* for an item i, denoted by $t_q(i)$, is the time at which i becomes available at a fixed known cost $c(i)$.

Definition 3.2 (rescind time) The *rescind time* for an item i, denoted by $t_r(i)$, is the time at which i becomes unavailable.

Note that by the definition of I given in section 3.1.2, $i \in I$ at time t only if $t < t_r(i)$.

Definition 3.3 (prequote time) The *prequote time* for an item i, denoted by $t_p(i)$, is the time at which the buyer learns about the upcoming availability of i. In particular, the quote and rescind times $t_q(i)$ and $t_r(i)$ are learned, and a probability measure $p : Z \rightarrow \Re$ on the outcome of the cost of i is learned or otherwise determined.

Note that by the definition of I given in section 3.1.2, $i \in I$ at time t only if $t_p(i) \leq t$. This, coupled with Definition 3.2, implies that $\forall i \in I$ at time t, $t_p(i) \leq t < t_r(i)$.

Definition 3.4 (prequote interval, quote interval) Let $[t_p(i), t_q(i)]$ and $[t_q(i), t_r(i)]$ be the *prequote interval* and *quote interval*, respectively.

Consider the simplest comparison shopping situation, where a supplier provides the purchaser with the asking price for items and the purchaser can purchase them at that price, for a possibly limited period. In this situation the supplier needs to estimate the market demand and set prices accordingly.

This simplest comparison shopping protocol consists of the following messages between the two principals: the purchaser P and the supplier S. From a purchaser's point of view this is a comparison shopping protocol because there may be several of these conversations occurring at once between the purchaser and various suppliers.

1. $P \rightarrow S$: $<$ request for item $i >$

   The purchaser tells the supplier what he wants to buy.

2. $S \rightarrow P$: $< c(i) >$ or $< c(i), t_r(i) >$

   The supplier gives the purchaser the cost of the item. Optionally a time limit (given by the rescind time $t_r(i)$) is provided. But if not, the purchaser may assume that $t_r(i) = t_n$, or may choose $t_r(i)$ based on the supplier's history of how long such offers last. Note that the purchaser would choose such a $t_r(i)$ only if he/she is certain that the offer will remain unchanged until at least that time.

3. $P \rightarrow S$: $<$ purchase order for $i >$

   Either there are no more messages or the purchaser accepts the offer and then provides payment and delivery instructions.

PQR is an enriched protocol in which the supplier has an opportunity to gauge market demand and prepare for a specific purchaser a customized pre-quote estimate, and a time interval in which the later-given quote is valid. The PQR protocol between one supplier and one purchaser consists of the same messages as the simple protocol above, replacing message 2 with the following:

2-a $S \rightarrow P$: $< p(\tilde{c}(i)), [t_q(i), t_r(i)] >$

   This is a prequote message. The supplier gives the purchaser a probability measure p on the outcomes for the uncertain cost $\tilde{c}(i)$ of the item, and the endpoints of the quote interval. The supplier may give several messages for the same product, using different quote periods and price estimates. Typically, one would expect the estimates to become more accurate as $t_q(i)$ approaches. The moment this message arrives is known as $t_p(i)$. The probability measure p and the rescind time $t_r(i)$ can be taken from sources other than the supplier, such as historical data.

2-b $S \rightarrow P$: $< c(i) >$ or $< c(i), [t_q(i), t_r(i)] >$

> The supplier is obligated to return a real numbered price for the item and this message should be sent and received no later than $t_q(i)$. There may be several of these messages. If the message contains just the cost, the purchaser may assume the item is available at that cost until at least the $t_r(i)$ time given in message 2-a. We assume that each prequote and quote period is associated with at least one cost valid in that period.

Table 3.1 summarizes the time periods during which the buyer will have information on the cost, potential cost, and availability of an item.

Interval	Information
$[t_0, t_p(i)]$	nothing is known about i
$[t_p(i), t_n]$	$t_q(i)$ is known; $t_r(i)$ is known; a probability measure on $c(i)$ (and perhaps $c(i)$ itself) is known
$[t_q(i), t_r(i)]$	i is available for purchase
$[t_q(i), t_n]$	the actual price of i is known
$[t_r(i), t_n]$	i is subject to unavailability or price change

Table 3.1: Summary of time periods during which the buyer will have certain information about an item i.

3.3 Assessing Utility

This section discusses the methods involved in determining a buyer's overall utility for purchasing a bundle of items. Although much of the background on utility assessment is given in chapter 2, specifics on determining a buyer's 1) utility function for money, 2) utility for each bundle, and 3) two-attribute utility function, which is used to calculate the buyer's overall utility for each bundle purchase, are given here.

3.3.1 Utility for Money

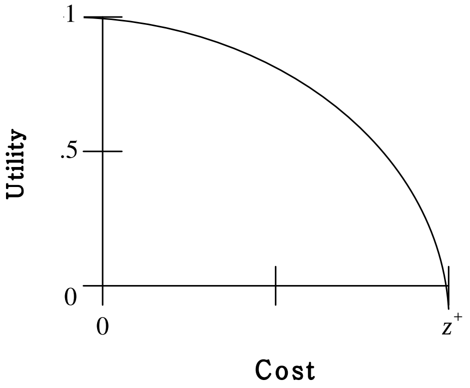
Section 2.3.4 gives a discussion on determining a utility function for a large set of consequences where the elements have some sort of natural ordering. This is the case when the consequences are gains or losses of monetary values. When analyzing a buyer's utility for money for the purpose of purchase utility assessment, all consequences are assumed to be losses, since monetary losses are equivalent to expenditures. That is to say that no positive monetary gains are considered valid consequences (i.e. we assume that a purchaser will not be paid to buy something). Note that this assumption is made strictly for simplicity, and does not affect the generality of problem.

Let Z be used to refer to the set of monetary units, with whatever desired degree of granularity (e.g. dollars, cents, thousands, etc.), let z^+ denote the highest value in Z, and let z^- denote the lowest value in Z, which unless otherwise specified, is assumed to be 0. Also, let $u_z : Z \to \Re$ be the monotonically decreasing utility function for money. This section outlines the qualitative and quantitative assessment for the determination of u_z for a given buyer.

Qualitative Characteristics

As stated before, all utility functions for spending money are assumed to be monotonically decreasing (spending more money is always worse than spending less, all other factors remaining equal). With this assumption in place, the remaining major qualitative assessment of a buyer's spending utility function is the level of risk aversion. This is determined by observing the buyer's preference between certain lotteries and their expected outcomes. As outlined in section 2.3.4, a decision-maker is risk averse iff he/she always prefers the expected outcome of a lottery for certain over the lottery itself. When the consequences are expenditures, then a risk averse buyer would, for example, prefer to spend \$100 on some item rather than take a 50-50 gamble on having to spend either \$80 or \$120. Such a buyer's utility function would be concave.

Figure 3.1 shows an example utility function for a risk averse spender, where a cost of \$0 (free) is the best consequence, and a cost of z^+ (the highest possible cost) is the worst consequence.



(b)

Figure 3.1: Utility function for a risk averse spender.

One can see from this function that, while the expected utility of a 50-50 lottery between spending 0 and z^+ would be $.5u_z(0) + .5u_z(z^+) = .5(1) + .5(0) = .5$, the buyer's utility for the expected outcome of the lottery (the midway point between 0 and z^+ on the x-axis) is about .75. Thus, the expected outcome is preferred. If the opposite is true, that the buyer always prefers the gamble, then he or she is risk prone and would have a convex utility function. If the decision maker is always indifferent between a lottery and its expected outcome, then he/she is said to be risk neutral and would have a linear utility function.

Quantitative Assessment

The quantitative assessment procedure involves the actual determination of a suitable number of points on the plane so that a curve can be fit. This is done with the use of certainty equivalents. Specifically, a point can be determined considering a 50-50 lottery between the expenditure of two values z_1 and z_2 for which the utilities $u_z(z_1)$

and $u_z(z_2)$ are already known. The certainty equivalent of this lottery is the value $\hat{z}$ such that the buyer is indifferent between spending the outcome of the lottery and $\hat{z}$ for certain. Thus the utility of $\hat{z}$ is equal to the expected utility of the lottery, and therefore $u_z(\hat{z}) = .5u_z(z_1) + .5u_z(z_2)$. This process is continued for different pairs of values until a sufficient number of points for some curve-fitting technique are determined. See [MP68] for a discussion on consistent simultaneous assessment of qualitative and quantitative properties.

3.3.2 Utility for Bundles

In order to determine the utility for each bundle of items, the method of direct assessment, described in section 2.3.4, must be used. Let $\mathcal{B}$ be a set of bundles, and let $u_b : \mathcal{B} \rightarrow \Re$ be the buyer's utility function for the bundles in $\mathcal{B}$. Let b^0 and b^* be the least and most preferred bundles in $\mathcal{B}$, respectively, and set $u_b(b^0) = 0$ and $u_b(b^*) = 1$. For any bundle b_i, some probability p, and two bundles b_j and b_k with known utilities $u_b(b_j)$ and $u_b(b_k)$, if b_i is a certainty equivalent of the lottery given in Figure 3.2, then $u_b(b_i) = pu_b(b_j) + (1 - p)u_b(b_k)$.

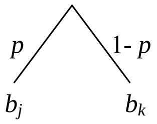


Figure 3.2: Direct assessment of bundle utility.

This method is only practical if there are few bundles (say ≤ 50 [KR76]), since accurate values can only reasonably be obtained with the help of multiple consistency checks, and the number of consistency checks needed grows exponentially with the number of bundles. For larger numbers of bundles, the problem becomes much more difficult. As a solution, one might suggest that since the number of items under consideration in $\mathcal{B}$ can be as low as $\log |\mathcal{B}|$, it would be better to directly assess the utilities of the items, and let the utility of a bundle be determined by an additive function of the utilities of its items. However, this would not work in the general case since these utilities are typically not additive. Each bundle must be assessed as a whole, not by the sum of its parts. A better idea might be to have the decision-maker put the bundles into groups, where all bundles in a group are (close to) equally preferable. The decision-maker would then have the task of assigning a utility to each group.

A full discussion on efficiently and accurately determining one's utility for goods is beyond the scope of this thesis, but is discussed extensively in the economics literature.

3.3.3 Utility of Bundle Purchases

Once the decision-maker's utilities for money and bundles are assessed, the overall utility for the purchase of a bundle b at total cost $c(b)$ can be determined. This utility, given by $u(b, c(b))$, is computed as a function of $u(b)$ and $u(c(b))$. This function is determined as follows.

The first step is to ascertain that the two attributes of a bundle purchase are mutually utility independent. That is, it must be shown that a buyer's 1) utility for bundles will be independent of cost and 2) utility for money will independent of the bundles that are procured. While any decision analysis will ultimately solely depend on the attitude of the decision-maker, it can easily be assumed that in most (if not all) reasonable circumstances, utility independence will hold between these two attributes in this situation. Consider the following demonstration where, for any given bundle purchase, attribute A_B is the bundle and attribute A_Z is the cost.

1) *A_B is utility independent of A_Z*: Let b_1, b_2, and b be bundles such that obtaining b at a cost of \$0 (i.e. for free) is the buyer's certainty equivalent for a 50-50 lottery between getting b_1 for free or b_2 for free. That is, the buyer is indifferent between the two lotteries in Figure 3.3(a). If, as the cost of these bundles is increased to some

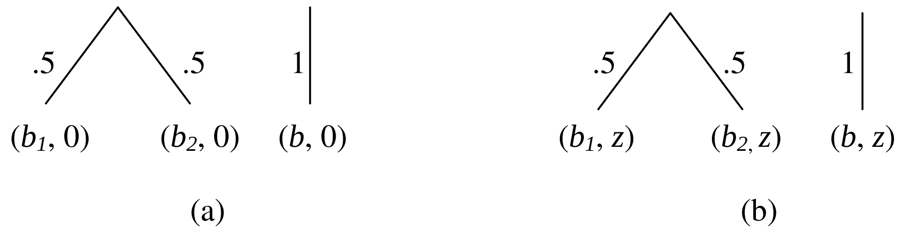


Figure 3.3: Demonstration of A_B's utility independence with respect to A_Z.

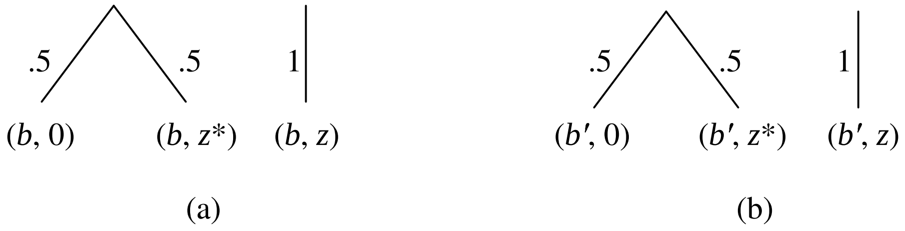


Figure 3.4: Demonstration of A_Z's utility independence with respect to A_B.

value, say z, the buyer sees no reason to adjust b (and thus is indifferent between the two lotteries in Figure 3.3(b)), then A_B is utility independent of A_Z.

2) *A_Z is utility independent of A_B*: Let z^+ be the highest possible cost for any bundle, let b be some arbitrary bundle, and let z be the amount such that the buyer is indifferent between a 50-50 lottery on obtaining b at a cost of z^+ or 0, and obtaining b at a cost of z for certain. That is, the buyer is indifferent between the two lotteries in Figure 3.4(a). If, for any different bundle b', the buyer sees no reason to adjust z (and thus is indifferent between the two lotteries in Figure 3.4(b)), then A_Z is utility independent of A_B.

In this thesis, we assume that this mutual utility independence holds. This is reasonable since, in almost any conceivable purchasing situation, a buyer will always have a desire to get a low price for any given bundle, regardless of which one, and will always have a desire to get the best bundle for any given cost, regardless of what that cost might be. Refer to section 2.3.5 for example situations where mutual utility independence definitely does not hold.

Once mutual utility independence has been established, an assumption must be made as to the additivity of the attributes in order to determine the two-attribute utility function. Section 2.3.5 defined two such two-attribute functions: the additive function and the bilinear function. In this thesis, we choose the bilinear

$$u(y,z) = k_y u_y(y) + k_z u_z(z) + k_{yz} u_y(y) u_z(z) \tag{3.1}$$

This choice is made for two reasons:

1) The additive function (see equation 2.4) can only be used if additive independence holds between the two attributes, which is difficult to show.

2) The additive function is just a special case of the bilinear function (with $k_{yz} = 0$).

Using this bilinear form, we obtain the two-attribute utility function for purchasing bundles as follows: Let $u_b : \mathcal{B} \rightarrow \Re$ and $u_z : Z \rightarrow \Re$ be the buyer's utility function for bundles and money, respectively. The buyer's two-attribute utility function $u : \mathcal{B} \times Z \rightarrow \Re$ is defined as

$$u(b,z) = k_b u_b(b) + k_z u_z(z) + k_{bz} u_b(b) u_z(z) \tag{3.2}$$

where k_b, k_z, and k_{bz} are scaling constants which sum to 1. To compute these scaling constants, as before let b^0 and b^* be the least and most preferred bundles respectively, and let 0 and z^+ be the minimum and maximum possible costs, respectively. Initially set the following utilities:

$$u(b^0, z^+) = 0 \qquad\qquad u(b^*, 0) = 1$$

Next, the utilities $u(b^0, 0)$ and $u(b^*, z^+)$ are individually determined in relation to $u(b^0, z^+) = 0$ and $u(b^*, 0) = 1$ using the method of direct assessment. By equations 2.9 and 2.10, the scaling constants k_b, k_z, and k_{bz} are calculated by

$$\begin{aligned} u(b^*, z^+) &= k_b(1) + k_z(0) + k_{bz}(1)(0) = k_b \\ u(b^0, 0) &= k_b(0) + k_z(1) + k_{bz}(0)(1) = k_z \end{aligned} \tag{3.3}$$

Since $k_b + k_z + k_{bz} = 1$, then

$$\begin{aligned} k_b &= u(b^*, z^+) \\ k_z &= u(b^0, 0) \\ k_{bz} &= 1 - (k_b + k_z) \end{aligned} \tag{3.4}$$

Once the two-attribute utility function $u(b, c(b))$ is defined for all $b \in \mathcal{B}$ for which a cost $c(b)$ exists (i.e. all items in b are available for sale), the best bundle to purchase is determined by selecting the b such that $u(b, c(b))$ is maximized.

3.3.4 Expected Utility of Bundle Purchases

For each item i in a bundle b there exists a probability measure $p_i : Z \to \Re$ on the price of i such that

$$\sum_{z \in Z} p_i(z) = 1 \tag{3.5}$$

Note that only *simple* probability measures have been considered to this point. This means that, while the number of possible outcomes may be infinite, p_i actually only assigns non-zero probability to a finite number of them. Also note that if the current time t is during the quote interval for an item i and its price is therefore known, say equal to z, then $p_i(z) = 1$. So there exists a probability measure for the price of all items in b, regardless of whether t is during their prequote interval or quote interval.

This allows a simple probability measure p_b on the price of the entire bundle to be specified, where the probability $p_b(z)$ of b costing z is the probability that the costs of all items in b sum to z. A buyer's expected utility for a bundle can now be calculated. Let $u_b : \mathcal{B} \rightarrow \Re$ be the buyer's utility for bundles, $u_z : Z \rightarrow \Re$ be utility for money, $u : \mathcal{B} \times Z \rightarrow \Re$ be the two-attribute utility function of u_b and u_z for bundles and money, and $p_b : Z \rightarrow \Re$ be the probability measure on the price for a bundle b. Also, let $\tilde{u}(b)$ denote the uncertain outcome of the two-attribute utility of purchasing b. The expected utility $E[\tilde{u}(b)]$ of purchasing b is

$$E[\tilde{u}(b)] = \sum_{z \in Z} p_b(z) u(b, z) \tag{3.6}$$

While it is beneficial for the sake of simplicity to discuss the basic concepts of expected utility theory using simple probability measures that give only a few discrete non-zero probability outcomes, in practice this might not be realistic. Typically, costs of future items can more easily be considered as outcomes of some (often normally distributed) random variable. For this reason, the cost $c(i)$ of such a future item i is henceforth described in this thesis by means of a continuous random variable X_i with a probability density function p_i, expected value $E(X_i)$, and variance $V(X_i)$. The cost $c(b)$ of a bundle of items is also a random variable with

$$E[c(b)] = \sum_{i \in b} E[X_i] \tag{3.7}$$

$$V[c(b)] = \sum_{i \in b} V[X_i] \tag{3.8}$$

Using these values to determine the probability density function p_b for the cost of b, the expected utility $E[\tilde{u}(b)]$ of purchasing b is then

$$E[\tilde{u}(b)] = \int_{-\infty}^{\infty} p_b(z) u(b, z) \; dz \tag{3.9}$$

Unfortunately, closed-form expressions for such integrals are often difficult to determine. They often must be estimated instead by means of an approximation formula or algorithm, a randomized algorithm that generates suitable sample points so that the integral can be estimated, or by splitting the function into pieces and treating it as a discrete distribution. The issues involved in calculating expected values for continuous random variables are addressed as needed in the next chapter.

Chapter 4

Simple Decision Making

4.1 Introduction

This chapter presents a strategy for making decisions in the online combinatorial purchasing problem. Here, the restriction is imposed that all items in a bundle must be purchased at the same time. This is a reasonable restriction to use in practice, since a buyer may be hesitant to make partial bundle purchases for fear that prices of subsequent required items may be too high. This would force the buyer to either pay too much for a bundle, or abandon some or all of the items purchased altogether. These costly possibilities are eliminated by this restriction. Another advantage of such an imposition is that the computational burden is somewhat lessened, since a number of possibilities are eliminated. All decisions are made at the bundle level, as opposed to the individual item level. This is more natural, since utility functions are developed for bundles and not items. While making decisions on whether or not to buy individual items based on the buyers' utility for bundles is much more difficult, it is not impossible. The development of techniques for decision-making in that domain is the topic of chapter 5.

In this chapter, the restriction of only purchasing complete bundles is formally added to the problem definition. A naïve solution to the decision problem is then presented that chooses to purchase a bundle if and only if its utility is greater than the expected utility of all other bundles. This is followed by a description of a more strategic technique that takes advantage of time periods during which many options are available by utilizing *comparison sets.* These sets help show, by grouping together future bundle purchases that will be available during the same time period, when is most likely to be the best time to buy [BS03a].

4.2 Restricting the Problem Definition

Definition 4.1 For a bundle b, the interval $pi(b) = [t_q(b),\ t_r(b)] = \bigcap_{i \in b}[t_q(i), t_r(i)]$ is known as the *purchase interval of* b. All items in a bundle can be purchased at any time during its purchase interval.

Definition 4.2 A bundle b is a *valid bundle* iff $pi(b) \neq \phi$.

Let t be the current time, let I be a set of items and $\mathcal{B}$ a set of bundles as defined in section 3.1.2, and let $t_p(i)$, $t_q(i)$ and $t_r(i)$ assign time points to each item $i \in I$ as defined in section 3.2. Also, let $\mathcal{B}_v \subseteq \mathcal{B}$ be the set of valid bundles. If $\exists b \in \mathcal{B}_v$ such that $t = t_r(b)$, then the purchaser must decide whether to buy b or allow it to expire and wait for another bundle in $\mathcal{B}_v \setminus b$.

This defines the set of decision points to be $\{t_r(b) \mid b \in \mathcal{B}_v\}$. In theory, any bundle is available for purchase at any time during its purchase interval, but it would be unwise to commit to purchasing it much before $t_r(b)$. First of all, since the cost of the bundle is fixed until $t_r(b)$ and $t_r(b)$ is known, there is no need to commit any earlier. Secondly, since new information on other bundles may arise, it would be best to wait until the last moment (perhaps leaving a minimal amount of time ϵ before $t_r(b)$ to perform the transactions or inform the suppliers of the buyer's intentions). Therefore, decisions only need to be made at (or just before) these $t_r(b)$ time points. At such a time, the utility of purchasing b is compared with the utilities and expected utilities of other available and future prospects, and a decision on whether or not to buy b is made. Note that if there are two bundles $b_j, b_k \in \mathcal{B}_v$ such that $t_r(b_j) = t_r(b_k)$ then the bundle with lower utility is eliminated from the decision process.

4.3 A Naïve Decision Procedure

This section formalizes a naïve decision procedure for the OCPP. The strategy presented is simple: Each time an item in a bundle b is about to expire, $u(b, c(b))$ is computed, as well as the expected utility of all other valid bundle purchases. If $u(b, c(b))$ is higher than all expected utilities, then buy b. Else, let it expire.

More formally, let b be the bundle currently available at cost $c(b)$ that is about to expire at time $t_r(b)$. Also let $\mathcal{B}_v$ be the set of valid bundles and let $\{E[\tilde{u}(b')] \mid b' \in \mathcal{B}_v\}$ be the set of expected utilities of all valid bundles. At (or just before) time $t_r(b)$,

> If $u(b, c(b)) \geq \max\{E[\tilde{u}(b')] \mid b' \in \mathcal{B}_v\}$, then purchase b.
>
> Else, allow b to expire.

This method is referred to as the naïve decision procedure since it merely pursues the bundle with the greatest expected utility, without using any strategy or taking any other factors into account. A more intelligent method that takes into account the impact of possible future options is now given.

4.4 An Improved Decision Procedure

4.4.1 Motivation

The presentation in the previous section offers a decision procedure that always chooses to pursue the bundle purchase with maximum expected utility. That is, each time a valid bundle purchase is about to expire, it must be determined whether or not there is a bundle purchase, either available now or in the future, that is likely to be better. As is shown in this section, this approach to decision-making is too naïve. Instead of determining whether or not *there is a future purchase that is likely to be better*, it should be determined whether or not *it is likely that a future purchase will be better*. These are two different questions, as explained with a simple example in the following section.

4.4.2 Expected Highest Value

Consider playing a game of chance with a typical six-sided fair die. You roll it once and get a 4. You are then given a decision to make: either you end the game and take $4, or you can choose to give up the $4 and roll the die twice more, winning the equivalent dollar amount of the higher of your two rolls. Making this decision by determining whether there is a future roll that is likely to give more than $4 would not be wise. If it was made in this way, the decision-maker would calculate that the expected value of each roll is 3.5 (and therefore the expected winnings $3.50), which is less than $4. So the decision-maker would choose to keep the $4. But the outcomes of these two rolls are not considered separately. The decision-maker has the opportunity to choose whichever is higher of the two, and should therefore compute the *expected higher* value. Let $\tilde{x}_h$ be the uncertain higher outcome of the rolls and let $P(\tilde{x}_h = k)$ be the probability that the higher value is k. The expected higher value is then

$$E(\tilde{x}_h) = \sum_{k=1}^{6} kP(\tilde{x}_h = k) \tag{4.1}$$

For example, if the events are two rolls of a die, the expected highest outcome is computed as follows:

Probability that each die value will be the highest of the two rolls:

6: 11 of the 36 possible outcomes will have at least one 6

5: 9 of the 36 outcomes will have at least one 5 and nothing higher

4: 7 of the 36 outcomes will have at least one 4 and nothing higher

3: 5 of the 36 outcomes will have at least one 3 and nothing higher

2: 3 of the 36 outcomes will have at least one 2 and nothing higher

1: 1 of the 36 outcomes will have at least one 1 and nothing higher

$$P(\tilde{x}_h = 6) = 11/36 \qquad P(\tilde{x}_h = 5) = 9/36 \qquad P(\tilde{x}_h = 4) = 7/36$$
$$P(\tilde{x}_h = 3) = 5/36 \qquad P(\tilde{x}_h = 2) = 3/36 \qquad P(\tilde{x}_h = 1) = 1/36$$

$$E(\tilde{x}_h) = 6(\frac{11}{36}) + 5(\frac{9}{36}) + 4(\frac{7}{36}) + 3(\frac{5}{36}) + 2(\frac{3}{36}) + 1(\frac{1}{36}) = 4.47$$

Since the expected highest value is 4.47, the decision-maker would expect on average to make \$4.47 if he/she chooses to continue.

The same idea comes up in making decisions about purchases. If one is trying to choose between making a purchase now and waiting until later, and it is known that there is a future time period where two or more bundles will be offered, the buyer needs to compare the utility of the current bundle with the *expected highest utility* of those future bundles, since the buyer will have the luxury of comparing them at that time and choosing the one with the highest utility. The notion of a *comparison set*, which is a set of bundles for which there is a period of time that the buyer will have complete information, is now introduced.

4.4.3 Comparison Sets

Recall that the purchase interval $pi(b)$ for a bundle b is the period of time during which the prices of all items in b are known, and all items are available for purchase.

Definition 4.3 Let $\mathcal{B}_v$ be a set of valid bundles and let $CS \subseteq \mathcal{B}_v$. CS is a *comparison set* of $\mathcal{B}_v$ iff it is maximal such that $ci(CS) = \bigcap_{b \in CS} pi(b)$ is non-empty. The interval $ci(CS)$ is called the *comparison interval* of CS. The *comparison set cover* $csc(\mathcal{B}_v)$ of $\mathcal{B}_v$ is the set of all comparison sets of $\mathcal{B}$.

Note that $ci(CS)$ is the period of time during which the prices of all items in all bundles in CS are known, and all items are available for purchase. So the buyer has complete information on all bundles in CS. Note that any bundle in $\mathcal{B}_v$ will appear in at least one comparison set even if by itself, and may also appear in more than one. Thus $csc(\mathcal{B}_v)$ is a covering of $\mathcal{B}_v$.

Algorithm 4.1 (Construction) The comparison set cover $csc(\mathcal{B}_v)$ for $\mathcal{B}_v$ is constructed by first finding the comparison intervals, and then determining the comparison sets from those. This is done as follows. Let T be a sorted list of the time points in $\{t_q(b) \mid b \in \mathcal{B}_v\} \cup \{t_r(b) \mid b \in \mathcal{B}_v\}$ from earliest to latest. Ties between a t_q and a t_r time are broken by placing the t_r time first, and all other ties are broken arbitrarily. For each pair of consecutive elements t_k and t_{k+1} in T, if t_k is a t_q time and t_{k+1} is a t_r time, then $[t_k, t_{k+1}]$ is a comparison interval, and $CS = \{b \in \mathcal{B} \mid [t_k, t_{k+1}] \subseteq pi(b)\}$ is therefore a comparison set. The comparison set cover $csc(\mathcal{B}_v)$ is then the set of all of these comparison sets [Hor02].

Example 4.1 Let $\mathcal{B}_v = \{b_1, b_2, b_3, b_4, b_5\}$ where each bundle has a purchase interval as depicted by horizontal lines in Figure 4.1 (e.g. the purchase interval for b_1 is $[0, 3]$). The comparison intervals are indicated by dotted vertical lines in Figure 4.2. The comparison set cover for $\mathcal{B}$ is then $csc(\mathcal{B}) = \{CS_1, CS_2, CS_3\}$, where $CS_1 = \{b_1, b_2\}$, $CS_2 = \{b_2, b_3, b_4\}$, and $CS_3 = \{b_5\}$.

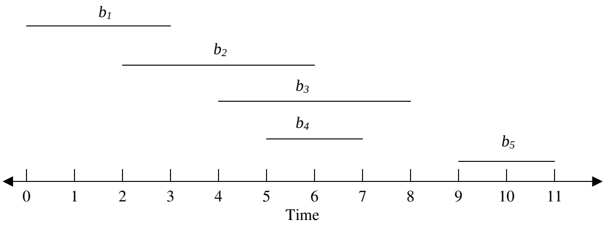


Figure 4.1: Purchase intervals for bundles in Example 4.1.

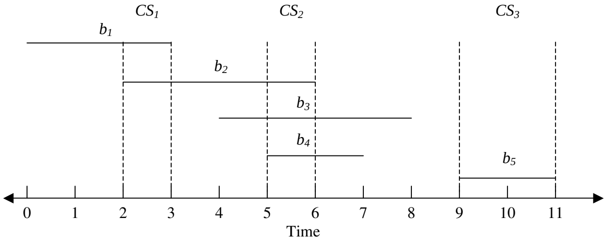


Figure 4.2: Comparison set cover of $\mathcal{B}_v$ in Example 4.1.

Since all items in all bundles in a given comparison set CS are available during a common interval and all prices are known, if the buyer chooses to buy during this period, he/she will choose the bundle in CS with the highest purchase utility. The utility one would expect to achieve during this period is therefore equal to the expected highest utility of the bundles in CS, referred to hereafter simply as the expected utility of CS and denoted by $E[\tilde{u}_{cs}(CS)]$.

4.4.4 The Proposed Decision Procedure

Let X be a set of alternatives upon which a decision must be made. This decision is said to be a *certain decision* if the outcomes of all alternatives in X are known for certain, and an *uncertain decision* if the outcomes for one or more alternatives in X are not known for certain. Note that utility is maximized when making a certain decision and expected utility is maximized when making an uncertain decision.

If a buyer chooses to buy during some comparison interval, he/she will make a certain decision on the bundles available during the interval and will thus choose the one that maximizes utility. So by determining the comparison intervals, we can reduce the large uncertain decision of choosing from several bundles into a potentially much smaller uncertain decision of choosing from a few certain decisions. The (uncertain) utility that one expects to have if faced with a future certain decision of choosing among bundles in a comparison set is the *expected highest utility* of those bundles.

Using this idea, whenever a bundle purchase with utility u is about to expire, the buyer determines whether or not there is a comparison set with expected utility greater than u. If so, this means that there is some future certain choice that is expected to be better, so the bundle should be allowed to expire. The entire decision procedure is now formally described.

Let b be the bundle currently available at cost $c(b)$ that is about to expire at time $t_r(b)$. Also let $\mathcal{B}_v$ be the set of valid bundles and let $csc(\mathcal{B}_v)$ be the comparison set cover for $\mathcal{B}_v$, each $CS \in csc(\mathcal{B}_v)$ with expected utility $E[\tilde{u}_{cs}(CS)]$.

> If $u(b, c(b)) \geq \max\{E[\tilde{u}_{cs}(CS)] \mid CS \in csc(\mathcal{B}_v)\}$, then purchase b.
>
> Else, allow b to expire.

Using this decision procedure is proven to provide the buyer with a higher expected utility than using the naïve decision procedure (see Theorem 6.1 in section 6.1.1). Unfortunately, computing expected utilities of comparison sets can be quite complex. This is the topic of the next section.

4.5 Calculating the Expected Utility of a Comparison Set

When using continuous random variables to represent item prices, in order to calculate the exact expected utility of a comparison set, one would have to solve the multiple integral

$$E[\tilde{u}_{cs}(CS)] = \int_0^1 \ldots \int_0^1 \max\{x_1, \ldots, x_n\} \prod_{i=1}^{n} p_i(x_i)\, dx_1 \ldots dx_n \tag{4.2}$$

where $x_1, \ldots, x_n$ are the utilities of the bundles in CS and $p_1, \ldots, p_n$ are their respective probability density functions. Since no closed-form expression exists for even the single integral of a normal probability density function [MS73], if some or all of the

p_i are normal (or some other complex form) then it is unlikely that the above can be expressed in closed form. Therefore it must be approximated.

One thing that needs to be considered is that, since bundles available in the same time period may share items, some interdependence may exist among bundle costs in a given comparison set. So if one bundle ultimately costs more than was originally predicted, it is likely that other bundles in the comparison set that have items in common will cost more as well. Moreover, interdependence may exist even if there are no common items, but still some interdependence in the costs of items exists (e.g. gas and oil). This makes the expected highest utility difficult to compute. Because of the potentially high quantity of bundles and items that could be under consideration, approximation methods for calculating probabilities become difficult and lead to high error. A simple and relatively accurate solution to the problem of approximating expected utilities for comparison sets is to use a *Monte Carlo* method.

4.5.1 Simple Calculation

If no interdependence exists between bundles in a comparison set, a simple method for approximating the expected highest utility is to divide the space of outcomes into discrete values (perhaps increments of 0.001 from 0 to 1), and determine the probability of each value being the highest. The expected highest value $E[\tilde{u}_{cs}(CS)]$ for a comparison set CS can be approximated by

$$E[\tilde{u}_{cs}(CS)] = \sum_{k \in K} k \left(P[\bigwedge_{b \in CS} \tilde{c}(b) \leq (k + d/2)] - P[\bigwedge_{b \in CS} \tilde{c}(b) \leq (k - d/2)] \right) \quad (4.3)$$

where K is the desired set of discrete units, d is the increment size of elements of K, and $\tilde{c}(b)$ is the uncertain outcome of the cost of b.[1]

[1] Here $P[\bigwedge_{b \in CS} q(b)]$ is the probability that property $q(b)$ holds for all $b \in CS$.

4.5.2 Monte Carlo Simulation

Monte Carlo [HH64, MU49] methods involve simulation to approximately solve a mathematical problem. Such a method can be used to estimate the expected highest utility of a set of bundles in a comparison set. This is done by first properly modeling the system of items residing in the bundles in question, which includes the probability distributions for costs of the items as well as possible interdependencies between the item costs, if they exist. The results of several independent simulations of the random elements involved in the system are then obtained. For each simulation, the outcomes of the item prices are used to determine the utility of purchasing each bundle, and the highest is noted. The average of these results is then taken as the unbiased estimator of the expectation θ, and the standard error is $\sigma/\sqrt{n}$, where σ is estimated by the sample standard deviation and n is the sample size.

This method is known as *crude* Monte Carlo, since it consists only of straight simulation. The simulation process can be improved upon, however, possibly producing a more accurate result with less work by using a *variance reduction technique.*

4.5.3 Variance Reduction

The efficiency of a Monte Carlo method strongly relies on the variability of the estimate. It is likely that a user of a Monte Carlo method will require some degree of confidence in the estimator, and therefore a sufficient number of values must be generated. This can become impractical, since in order to reduce the standard error $\sigma/\sqrt{n}$ by a factor of 10, for example, n must be increased by a factor of 100. An additional or alternative method for reducing the standard error is to use a variance reduction technique. Such techniques usually involve modifying the simulation in such a way so that the outcomes will have the same expected value but a smaller variance.

The variance reduction technique used in experiments for this thesis is referred to as the *antithetic variate* method [HM56]. Antithetic variates are estimators that mutually compensate for each other's variations. This idea is used to reduce the variance in the estimation of an expectation as follows (as given in Hammersley and Handscombe [HH64]): Two estimators t and t' are chosen, where t is the original estimator of θ, and t' is an estimator that has the same (unknown) expectation as t but has a strong negative correlation with t. Then $\frac{1}{2}(t+t')$ will be an unbiased estimator of θ, and its sampling variance is

$$var[\frac{1}{2}(t+t')] = \frac{1}{4}var(t) + \frac{1}{4}var(t') + \frac{1}{2}cov(t,t') \tag{4.4}$$

Suitably choosing t' so that the covariance $cov(t,t')$ is negative has the potential to significantly reduce the variance. Note that for the reduction to be significant, it must save more work than the extra work required by computing the second estimator.

This technique is used to reduce the variance when finding the expected utility of a comparison set as follows. Note that, for this particular technique to work properly as it is applied here, random variables must have symmetric probability distributions (e.g. normal). In each simulation, the cost $c(i)$ of each item i in the comparison set is selected at random from its corresponding probability distribution p. Let a second cost $c'(i) = 2E[c(i)] - c(i)$ be a value such that $p(c'(i)) = p(c(i))$ (see Figure 4.3 for an example). The first estimator t is then constructed by finding the highest utility of all bundles using the $c(i)$ values. The second estimator, t', is the highest utility of all bundles using the $c'(i)$ values.

Note that since $c(i)$ and $c'(i)$ are chosen from the same distribution for each i, then t and t' have the same expectation. So t and t' are both unbiased estimators of θ, and therefore $\frac{1}{2}(t+t')$ is an unbiased estimator of θ. Also note that, for each i, $c(i)$ and $c'(i)$ values should be negatively correlated (especially if the distributions are normal). This should cause t and t' to be negatively correlated, thus reducing the variance of the estimator. The effectiveness of the technique is tested in section 6.4.

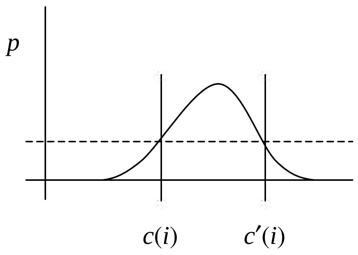


Figure 4.3: Obtaining the $c'(i)$ for the second estimator t'.

Chapter 5

Complex Decision Making

5.1 Introduction

The preceding chapter discusses a method for combinatorial purchasing in which all items in a bundle are purchased at once. That is, the exact bundle to be purchased is decided upon before any individual items are purchased, and any purchasing can only be done at a point in time when all items in the bundle are available. The advantage of this model is that the purchaser not only knows that all items are available, but also knows the total cost of the items before he/she commits to buying any of them. This eliminates the risk of purchasing a part of a bundle only to realize that the remaining items are more expensive than originally believed, or perhaps unavailable altogether. However, this restriction can significantly reduce the number of viable options that the buyer is allowed to pursue. It would be beneficial to allow such partial bundle purchases if the potential gain in utility is high and the risk is relatively low.

This chapter presents a method for bundle purchasing that allows for these partial purchases. This is done by means of building a *purchase procedure*. The procedure will guide the decision process through the individual purchases, helping to make rational decisions along the way, until an entire bundle is procured. This is done with the use of expected utility theory. Simply put, the buyer is instructed to purchase an item at a given time if the expected utility of buying the item is higher than the expected utility of not buying the item. Calculating these utilities, however, is a daunting task since several issues need to be considered. Consider an item i that is about to expire. The expected utility of buying i depends on the possible bundle purchases that include i and any items already purchased, and the expected utility of not buying i depends on the possible bundle purchases that include the items already purchased but do not include i. Determining the expected utility of a choice is more difficult here because one must consider not only the utilities of the bundles that could be completed as a result of the choice and their expected costs, but also what decisions will subsequently need to be made because of the path that has been chosen. One must also consider what information will be available to the purchaser at these later decisions, making this an extremely difficult mathematical problem.

The decision problem here is formally defined as follows: Let t be the current time, let I be a set of items and $\mathcal{B}$ a set of bundles as defined in section 3.1.2, and let $t_p(i)$, $t_q(i)$ and $t_r(i)$ assign time points to each item $i \in I$ as defined in section 3.2. If $\exists i \in I$ and $\exists b \in \mathcal{B}$ such that $i \in b$ and $t = t_r(i)$, then the purchaser must decide whether to buy i or allow it to expire.

After the structure of a purchase procedure tree is defined and its usage is demonstrated, a discussion on the general usage of decision trees to solve choice points in a purchase procedure is given. The rollback solution method of computing expected utilities for choices in a decision tree is then described. Next, a modified version of the decision tree that is more suited to solving decisions native to this particular problem, called the *Quote-Rescind tree* (QR-tree), is proposed. This decision tree is much smaller and requires less redundant work than would a general decision tree. Continuous random variables in the tree are handled by both discrete approximation and Monte Carlo simulation, and several solution techniques are presented. Finally, a few issues that had been ignored for the sake of simplicity are brought back to complete the discussion.

5.2 Purchase Procedure Trees

In order to structure the decision process, a purchase procedure tree is introduced. This tree graphically depicts the process of making decisions and purchases that, given the set of items for which there is a prequote or quote, when executed will result in a complete bundle purchase. Starting at the root node, the buyer proceeds toward the leaf nodes, buying items at purchase nodes and making decisions at decision nodes. Once the final purchase is made at some leaf in the tree, the buyer will have procured a complete bundle. This section presents the formal definition as well as the method for construction of such a tree.

5.2.1 Purchase Procedure Tree Definition

A purchase procedure tree is a tree $T = (V, E)$ where V is partitioned into two types of nodes: a set P of purchase nodes and a set D of decision nodes. The purchase nodes are labeled by the items they represent. There are also three functions on the nodes, $t_q : P \rightarrow \Re$, $t_r : P \rightarrow \Re$, and $t : D \rightarrow \Re$. At time t, let I be the set of items not yet procured for which the prequote or quote interval includes t, and $\mathcal{B} \subseteq 2^I$ be a set of bundles. T is a *purchase procedure tree at time t on $\mathcal{B}$* iff the following are true:

- Each purchase node in T has at most one child node.
- Each decision node in T has two child nodes.
- Each purchase node p in T represents the purchase of an item i and $t_q(p) = t_q(i)$ and $t_r(p) = t_r(i)$.
- For any two purchase nodes p_1 and p_2 in T, if p_1 is an ancestor of p_2 then $t_r(p_1) \leq t_r(p_2)$.
- For any two sibling nodes v_1 and v_2 in T, if v_1 is to the left of v_2 then $t_r(v_1) \leq t_r(v_2)$.

- For any decision node d with left child $\ell(d)$, $t(d) = t_r(\ell(d))$.
- For any root-to-leaf path in T, the set of items represented by the purchase nodes on the path is a bundle in $\mathcal{B}$, and all elements of $\mathcal{B}$ are represented by some path.

Execution of the purchase procedure begins at the root of the tree and continues toward the leaves. Whenever a purchase node p is encountered, the buyer is advised to purchase the item represented by p at (or just before) time $t_r(p)$. When a decision node d is encountered, the buyer must make the decision of which path to choose. There will always be two options: buy the item in the remaining bundles that will expire next (ties are broken arbitrarily), which is represented by the left child node, or allow it to pass. Execution continues until a leaf node is reached and the corresponding purchase is made, completing a bundle purchase.

Note that a purchase procedure tree built at time t will not depict purchases or decisions that occurred before t. The root always represents the first action at t.

Figure 5.1 depicts an example purchase procedure tree where $I = \{A, B, C, D, E, F\}$ and $\mathcal{B} = \{\{A, B\}, \{A, C\}, \{D, B\}, \{E, F\}\}$ (hereafter written simply as $\mathcal{B} = \{AB, AC, DB, EF\}$). The convention for drawing such trees is to label purchase nodes by the item each represents and decision nodes by a uniquely subscripted d.

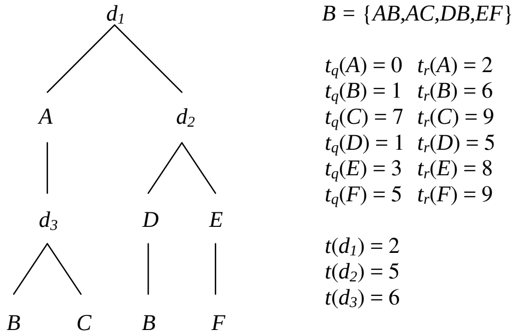

d_1
A
d_2
d_3
D
E
B
C
B
F
$B = \{AB,AC,DB,EF\}$
$t_q(A) = 0$ $t_r(A) = 2$
$t_q(B) = 1$ $t_r(B) = 6$
$t_q(C) = 7$ $t_r(C) = 9$
$t_q(D) = 1$ $t_r(D) = 5$
$t_q(E) = 3$ $t_r(E) = 8$
$t_q(F) = 5$ $t_r(F) = 9$
$t(d_1) = 2$
$t(d_2) = 5$
$t(d_3) = 6$

Figure 5.1: Example purchase procedure tree.

5.2.2 Construction

Algorithm 5.1 (Purchase Procedure Tree Construction) Let I be the set of items and $\mathcal{B} \subseteq 2^I$ the set of bundles. For any node n, let I_n be the set of items labeling ancestors of n and let L_n be the set of items labeling the left children of proper ancestor decision nodes of n. Then the set of bundles that can be procured below n is $\mathcal{B}_n = \{b \in \mathcal{B} \mid I_n \subseteq b\} - \{b \in \mathcal{B} \mid \exists \ell \in L_n \wedge \ell \in b\}$ (since no item labeling the left child of a decision node d can appear in d's right subtree), and therefore $I_{\mathcal{B}_n} = \{i \in b \mid b \in \mathcal{B}_n\} \setminus I_n$ is the set of items that can potentially label proper descendents of n.

1. Let r be the root;
2. *construct*(r);
3. While there is a non-terminal leaf node n, *constructChildren*(n);

construct(n): If $I_{\mathcal{B}_n} = \phi$, then let n be a terminal node. Else if there exists an i such that $t_r(i)$ is a minimum in $I_{\mathcal{B}_n}$ and $i \in b$ for all $b \in \mathcal{B}_n$, then let n be a purchase node labeled by i. Else, let n be a decision node.

constructChildren(n): If n is a purchase node, then let $c(n)$ be the child of n and *construct*$(c(n))$. Else n is a decision node. Let $\ell(n)$ and $r(n)$ be the left and right children of n respectively, and i be an item in $I_{\mathcal{B}_n}$ such that $t_r(i)$ is minimal. Let $\ell(n)$ be a purchase node labeled by i, let $t(n) = t_r(\ell(n))$, and *construct*$(r(n))$.

Note that, while the purchase procedure tree will never plan them, extraneous purchases are possible with this model. For example consider the purchase procedure depicted by the tree in Figure 5.1, in which the purchaser chooses to buy A. After A is purchased, the new set of bundles will be $\mathcal{B} = \{B, C, DB, EF\}$ (assuming no new offers have entered the situation). This allows for the possibility that EF could still be purchased if it turns out to have a highest expected utility of all options, even if it meant that A is wasted. This occurrence is unlikely, however, given the fact that the prospects associated with buying A were good enough to warrant A's purchase.

5.3 Conventional Decision Trees for Purchase Procedures

While executing a purchase procedure, the buyer must decide which course of action is likely to be most beneficial whenever a decision node is visited in the procedure tree. To facilitate the decision process, expected utility maximization is used as the criterion for ascertaining the best choice. Since the expected utility of a choice is dependent on choices that will be made at subsequent decision points, a *decision tree* is used.

5.3.1 Conventional Decision Trees

The general terminology and structure for the decision tree [Mag64] in this discussion is largely taken from Raiffa [Rai68] and Goodwin and Wright [GW99].

Representation

A decision tree, as it pertains to decision analysis, is a schematic presentation of a sequence of decisions and their possible consequences. Typically, the tree is drawn so that the root is on the left and the leaves are on the right. There are three types of nodes:

- decision nodes, typically represented by squares
- chance nodes, typically represented by circles
- leaf nodes (often referred to as the endpoints)

Edges are labeled as follows:

- Edges emanating from a decision node are labeled by the possible options.
- Edges emanating from a chance node, are labeled by the possible consequences, as well as the probability of each consequence occurring, if available.

Finally, the endpoints are labeled by the numeric measure (e.g. utility, money) that would result from the corresponding sequence of decisions and consequences. Figure 5.2 depicts an example decision tree for the decision problem described in Example 5.1. Execution moves left to right. At each square node, the decision maker needs to choose exactly one of the options labeling the following edges. Execution proceeds down that path. At circle nodes, one of the consequences on the following edges will be selected by chance, and execution will proceed down that path. This process continues until an endpoint is reached, and the decision-maker has achieved the corresponding value.

Example 5.1 John is 20 years away from retirement and has been offered a choice between two positions in his company. Position A will pay John \$40,000 per year, with a 10% chance for a \$10,000 raise after 10 years. Position B pays only \$30,000 per year, but there is a 50% chance of getting a \$30,000 raise after 10 years. Also, if he takes Position B and does not get this raise, he will have the option of switching to Position A at \$40,000 per year. John's only concern is making as much money as possible over the next 20 years. Figure 5.2 depicts the decision tree for this decision problem.

Solution

While the decision tree provides a good graphical understanding of a decision problem, the main object in modeling a decision problem in this way is to determine the best course of action. Specifically, it is used to determine the choice that will give the decision maker the best expected outcome at a given decision point. This is done using the *rollback solution method.*

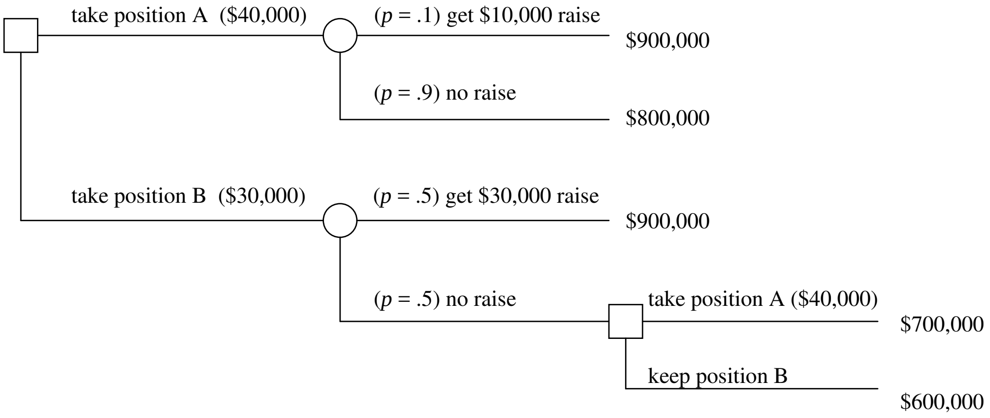

take position A ($40,000)
(p = .1) get $10,000 raise
$900,000
(p = .9) no raise
$800,000
take position B ($30,000)
(p = .5) get $30,000 raise
$900,000
(p = .5) no raise
take position A ($40,000)
$700,000
keep position B
$600,000

Figure 5.2: An example decision tree.

To apply this method, the tree is analyzed from right to left by considering the later decisions and chances first. Each node will be labeled by the outcome $E(n)$ expected by the decision maker if this node is traversed. Initially, all endpoints are labeled. For any chance node n such that all children nodes C of n are labeled, if $p(n, c_i)$ is the probability labeling the edge (n, c_i) of the corresponding consequence, then

$$E(n) = \sum_{c \in C} p(n, c)E(c) \tag{5.1}$$

For any decision node n such that all children nodes C of n are labeled, $E(n)$ is simply the maximum $E(c)$ for all $c \in C$, since the decision-maker would just choose the best of all choices. Consider Figure 5.3, which depicts the solution for the tree in Figure 5.2. This indicates that the decision-maker would expect to make $10,000 more if position A is taken.

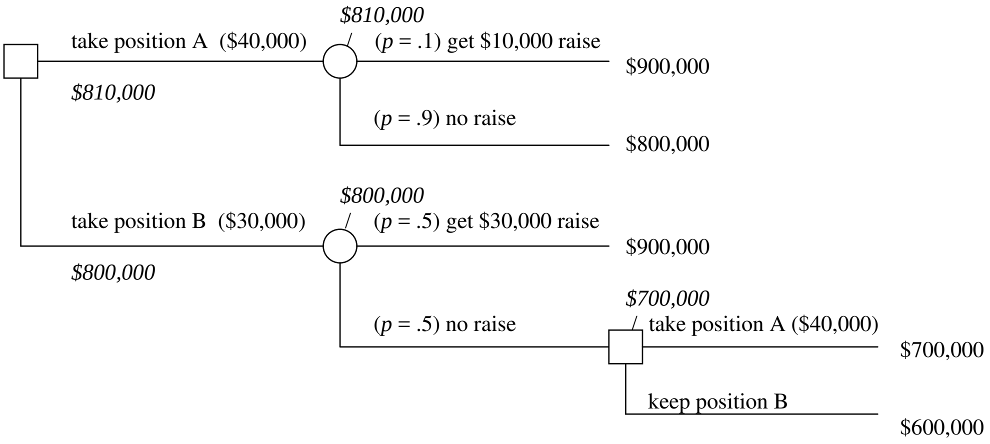


Figure 5.3: Solution to the example decision tree.

5.3.2 The Purchase Procedure Tree as a Decision Tree

Transformation without Chance Nodes

To determine the course of action to be taken at a decision in a purchase procedure, the purchase procedure tree can be transformed to a special case of a decision tree, where there are no chance nodes. For every pair of decision nodes d' and d'', if there are no intervening decision nodes between d' and d'', replace the path from d' to d'' with a single edge. Label this edge with all of the items that were on the original path. For every decision node d that has no descendent decision node, replace each of the two paths below it with an edge as just described and create a leaf node for each edge. These leaf nodes are the endpoints. Label the endpoints by the expected utility of purchasing the bundle formed by the items labeling edges on the path from the root. If the root is not a decision node, then remove the nodes and edges above the first decision node d and add the items that had labeled those nodes to the items labeling the edges emanating from d. Finally, draw the tree from left to right and the decision nodes as squares. The resulting tree is a decision tree as defined in section 5.3.1. Figure 5.4 illustrates an example transformation.

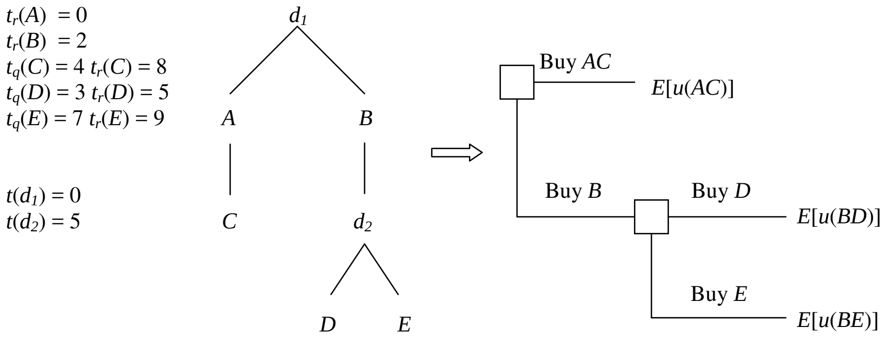


Figure 5.4: Example transformation of a purchase procedure tree to a decision tree.

This decision tree can be used to solve naïvely the problem of deciding whether or not to make a purchase. Using the rollback method on this tree would effectively advise the buyer to buy an item i iff there exists a bundle $b \in \mathcal{B}$ containing i such that expected utility $E[\tilde{u}(b)] = \max\{E[\tilde{u}(b')] \mid b' \in \mathcal{B}\}$. This would be the proper method if a decision had to be made up front on which bundle to buy before any information is known. However in real life, one would (almost) never buy an item without first knowing its price. The act of learning outcomes of item prices must be inserted into proper positions in the decision tree as chance nodes. Only then can the decision problem be properly modeled by the decision tree and can accurate expected values be computed.

Inserting Chance Nodes into the Decision Tree

Informally, a chance node is inserted between any pair of adjacent decision nodes in the decision tree if any new information is expected to become available during the interim between decision times. Formally, let d' and d'' be any two adjacent decision nodes such that $t(d') < t(d'')$, and let $I_{\mathcal{B}_{d'}}$ be the set of items that label edges below d'. If there exists an $i \in I_{\mathcal{B}_{d'}}$ such that $t(d') < t_q(i) \leq t(d'')$, then a chance node needs to be inserted between d' and d''.

Consider the following notation used to represent a set of outcomes for a set of items I. Let $K(I) = \{k_1, \ldots, k_n\}$ be the set of joint outcomes for I where each $k_j : V \rightarrow \Re$ is a function that assigns the cost to each vertex representing an item in I for that particular outcome. Note that it is possible for two $v, v' \in V$ to represent the same item $i \in I$. In this case, for any $k_j \in K(I)$, $k_j(v) = k_j(v')$. If the outcomes for item prices are discrete, then the procedure of adding a chance node is as follows. Let $I_{d'} = \{i \in I_{\mathcal{B}_{d'}} \mid t(d') < t_q(i) \leq t(d'')\}$ be the set of items below d' for which prices become known before the decision at d'', let c be the new chance node, and let $K(I_{d'})$ be the set of possible joint outcomes for prices of items in $I_{d'}$. Remove the edge (d', d'') and add edge (d', c). Create $|K(I_{d'})|$ copies of the subtree beginning at d'', joining an edge from c to the root of each subtree, and label each of these new edges with a unique $k \in K(I_{d'})$ as well as the probability of k occurring. Finally, re-label each endpoint in the tree by the utility of buying its corresponding bundle at the cost outcomes specified by the labels of the edges on its path from the root.

Example 5.2 Consider again the purchase procedure tree in Figure 5.4. At the root decision time (say time 0), the outcomes of A and B are already known, and all other items have two possible outcomes, either high or low, each with a 50% chance of occurring. Figure 5.5 represents the decision tree to be used for solving the root decision, with chance nodes included.

If the outcomes for item prices are continuous, then inserting a chance node is not as straightforward since there will be an infinite number of outcomes. One solution is to approximate the corresponding continuous probability density function (pdf) with a discrete distribution that has a finite number of outcomes. For example, the Pearson-Tukey (PT) three-point approximation [PT65, KB83] has been found to be surprisingly accurate with a number of such functions. The three discrete outcomes $\{x_1, x_2, x_3\}$, each with probability $p(x)$ of occurring, associated with a PT approximation of a continuous probability density function for a random variable X are as in Table 5.1. If X is a normally distributed random variable with mean μ

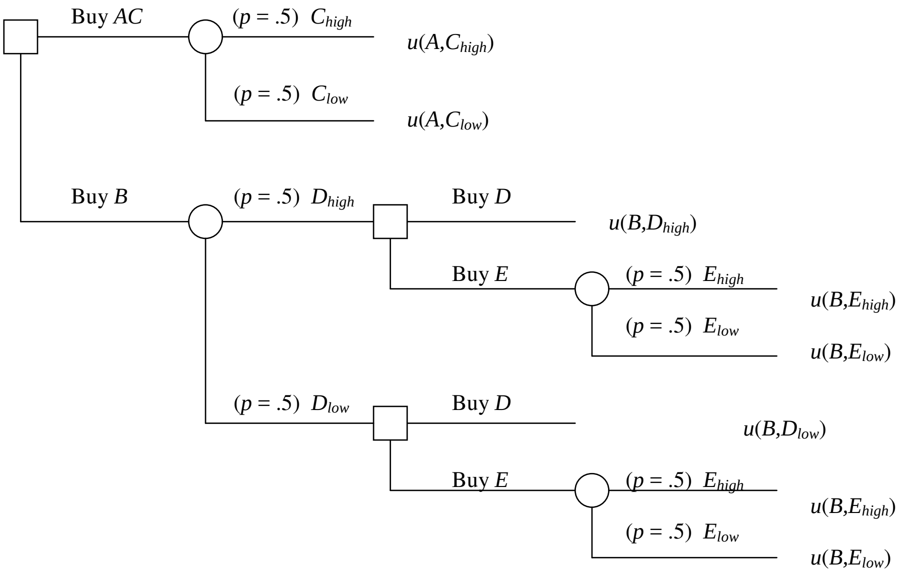


Figure 5.5: Transformation of the purchase procedure tree in Figure 5.4 to a decision tree (described in Example 5.2).

and standard deviation σ, then the three outcomes in a PT approximation are as in Table 5.2.

Outcome x	$p(x)$
x such that $P(X > x) = .95$	.185
x such that $P(X > x) = .5$	.63
x such that $P(X > x) = .05$	.185

Table 5.1: Outcomes and probabilities for the PT three-point approximation.

Chance nodes for items for which there is a continuous pdf on the price outcomes can now be inserted in the same manner as those with discrete distributions if the PT approximation is used. Another method for incorporating chance nodes with continuous pdf's into a decision tree is to use a *Monte Carlo* algorithm. This technique is investigated in section 5.6.

Outcome x	$p(x)$
$\mu - 1.645\sigma$	.185
μ	.63
$\mu + 1.645\sigma$	.185

Table 5.2: Outcomes and probabilities for the PT approximation of a normal random variable.

5.3.3 Inefficiencies of Conventional Decision Trees

While the previous section describes a transformation of the purchase procedure tree that yields a perfectly legal decision tree, because of the nature of the problem that the decision tree is being used to solve, there are two major inefficiencies with the design. Even if there are only three outcomes for each chance node, using this method for instances in which there are many items will cause trees to be too large. Fortunately, this unmanageable growth is the result of many unneeded subtree duplications in the construction that can be avoided. After giving a brief analysis of this growth, this section discusses the two problems that cause identical subtrees to be constructed.

Let T be a decision tree constructed as demonstrated above and let m be the number of discrete outcomes for each item. Let n be a chance node in T with child decision nodes (as opposed to leaf nodes) and let $I(n)$ be the items for which outcomes are learned at n. Then there are $m^{|I(n)|}$ joint outcomes for n, and therefore n has $m^{|I(n)|}$ child decision nodes. Since each decision node has two children, the rollback method has $2m^{|I(n)|}$ subproblems to solve before the decision nodes can be resolved and the expected value $E(n)$ of n can be computed. Fortunately, this number can be reduced since many of the subproblems are the same and therefore need to be solved just once. Each of the two causes for subtree duplication are now individually demonstrated.

Duplication Cause #1: Let d be a decision node child of n, let c and c' be the children of d and let $I(c)$ and $I(c')$ be the subsets of $I(n)$ that may be purchased if the decision is made to proceed to c or c' respectively. Then there are $m^{|I(n)|-|I(c)|}$ decision node children of n for the same outcome of the items in $I(c)$ (but different outcomes for $I(n) \setminus I(c)$). Since the subproblem of determining the expected utility of c only needs to be solved once (similarly for c'), there are only $m^{|I(c)|} + m^{|I(c')|}$ subproblems to solve. Refer to Figure 5.6 for example. The problems of solving each of subtrees T_1 and T_2 are identical, since both represent the problem of computing expected utility if the buyer buys B at \$5 and C at \$6.

Duplication Cause #2: Let c be a child of a decision node d and let I_d be the set of items purchased on the path from the root to d. If there is a subset $I \subseteq I_d$ such that there are two joint outcomes k and k' such that the total cost of the items in I for outcome k is equal to the total cost of the items in I for outcome k', then a duplicate of the subtree below c exists. Again, refer to Figure 5.6. The problems of solving each of subtrees T_2 and T_3 are identical, since, even though B and C have different outcomes, both represent the problem of computing expected utility if the buyer buys B and C at a total of \$11.

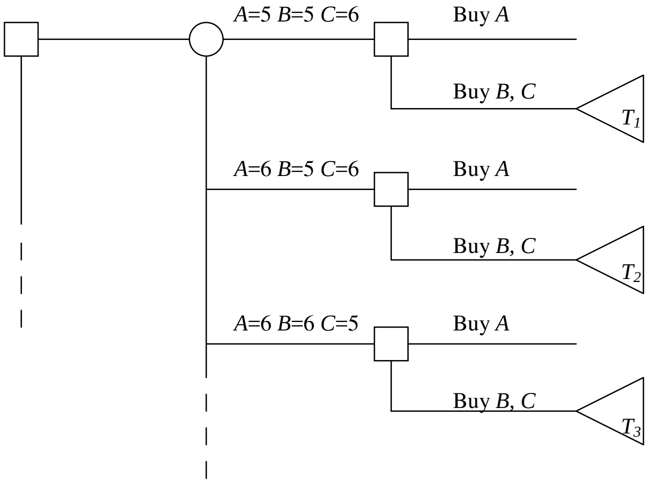


Figure 5.6: Decision tree where T_1, T_2 and T_3 represent the same subproblem of computing expected utility where the buyer has bought B and C at a total of \$11.

In order to eliminate the above inefficiencies caused in building general decision trees to solve the problem of decision making in a purchase procedure, a new type of tree referred to as the QR-tree is proposed.

5.4 QR-Trees for Purchase Procedures

The QR-tree [BS03b] is similar to and therefore about the same size as the purchase procedure tree, but has all the information necessary for rollback solution stored at the nodes. Like decision trees, each time a decision is to be made in the purchase procedure, a QR-tree is built to analyze the consequences and help determine the expected value of each choice. It is an organization of not only when decisions need to be made, but also of when quotes for items will open and close. Purchase nodes are sorted in the tree according to their quote times, while decisions are inserted appropriately according to relevant purchase rescind times. It is therefore simple to see what relevant information will be known at each decision point in a QR-tree.

5.4.1 Construction

The QR-tree is constructed by a direct transformation of a purchase procedure tree, as described in Algorithm 5.2. This process simply reorders the purchase nodes so that they are sorted by t_q time rather than t_r time. Decision nodes and their corresponding decision times are unchanged. Steps 1 and 2 move all purchase nodes to the appropriate segments of the tree (i.e. in between the appropriate pairs of decision nodes). Once all of these reside in the proper segments, step 3 sorts the purchase nodes within each segment by t_q time in ascending order.

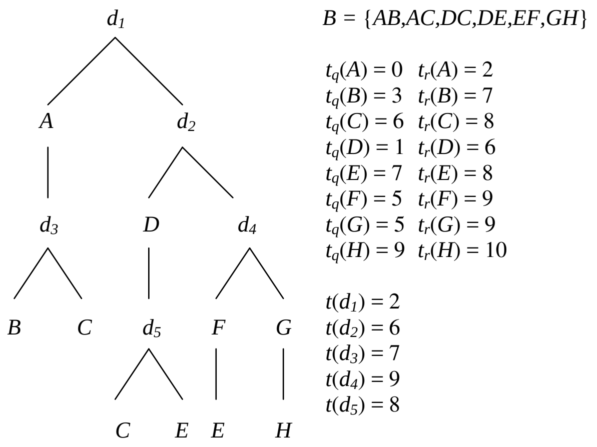
Algorithm 5.2 Let T be a purchase procedure tree with a decision node root. T is transformed to a QR-tree as follows:

1. For any decision node d in T, let $t_q(d) = \min\{t_q(v) \mid v \in des(d)\}$ where $des(d)$ is the set of descendent purchase nodes of d.

2. While there exists a purchase node p with first descendent decision node d such that $t_q(d) < t_q(p)$, remove p from its position in T and insert it below d as $\ell(d)$. Make a copy of $\ell(d)$ and insert the copy as $r(d)$.

3. For every pair of decision nodes d and d' in T such that d is an ancestor of d' and there are no intervening decision nodes on the path from d to d', sort the purchase nodes on this path by their t_q values in ascending order from d to d'.

4. Give each leaf in this tree a child node. These new nodes are called the *endpoints*.



d_1
A
d_2
d_3
D
d_4
B
C
d_5
F
G
C
E
E
H
$B = \{AB,AC,DC,DE,EF,GH\}$
$t_q(A) = 0$ $t_r(A) = 2$
$t_q(B) = 3$ $t_r(B) = 7$
$t_q(C) = 6$ $t_r(C) = 8$
$t_q(D) = 1$ $t_r(D) = 6$
$t_q(E) = 7$ $t_r(E) = 8$
$t_q(F) = 5$ $t_r(F) = 9$
$t_q(G) = 5$ $t_r(G) = 9$
$t_q(H) = 9$ $t_r(H) = 10$
$t(d_1) = 2$
$t(d_2) = 6$
$t(d_3) = 7$
$t(d_4) = 9$
$t(d_5) = 8$

Figure 5.7: Example of a QR-tree.

Figure 5.8 gives an example of such a transformation. For each purchase node in the example, the subscripts denote the t_q and t_r times respectively, while the lone subscript for each decision node denotes the decision time. Notice that in the QR-tree, decisions are made at the same time as in the purchase procedure tree, and the same bundles are pursued for each choice. The only difference is that some purchases are pushed closer to the end. Note that this reordering is done only for the sake of

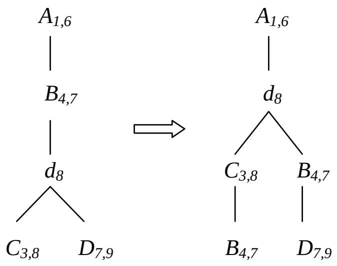


Figure 5.8: Example of a QR-tree transformation as described in Algorithm 5.2.

look-ahead to predict expected utility. In reality, all purchases will occur at (or just before) their t_r times. Since endpoints serve no purpose other than for convenience in computation (as is shown later), they typically are omitted in QR-tree drawings. A QR-tree has all of the same properties as a purchase procedure tree except:

- For any two nodes v_1 and v_2 in the tree, if v_1 is an ancestor of v_2 then $t_q(v_1) \leq t_q(v_2)$.
- A decision node d with decision time $t(d)$ may have a descendent purchase node p such that $t_r(p) < t(d)$. If this is the case, however, the purchase represented by p will be part of any bundle procured below d, and is thus not relevant to the decision.

Theorem 5.1 Let T^{PP} be a purchase procedure tree and T^{QR} be the QR-tree transformed from T^{PP} as in Algorithm 5.2. Making purchases when purchase nodes are encountered and using any decision procedure to make decisions at decision nodes, the expected utility of traversing T^{PP} is equal to that of traversing T^{QR}.

Proof. Let T_d^{PP} and T_d^{QR} be the tree structures containing only decision nodes and edges (no purchase nodes) of T^{PP} and T^{QR}, respectively. Since the transformation does not change the decision times nor the position of the decision nodes relative to each other, $T_d^{PP} = T_d^{QR}$. Let d be a decision node in T^{PP} and let $anc_i(d)$ be the set of items labeling d's ancestor purchase nodes and $\ell_i(d)$ be the item labeling the left child of d. Then d represents the decision to be made at time $t(d)$ between pursuing either bundles $\mathcal{B}_\ell^{PP} = \{b \mid anc_i(d) \cup \{\ell_i(d)\} \subseteq b\}$ or $\mathcal{B}_r^{PP} = \{b \mid anc_i(d) \subseteq b \ \wedge \ell_i(d) \notin b\}$ of bundles. Let d^{QR} be the decision node in T^{QR} that corresponds to d, and let $\mathcal{B}_\ell^{QR}$ and $\mathcal{B}_r^{QR}$ be the set of bundles that can be procured in the left and right subtrees of d_{QR}, respectively. Since $T_d^{PP} = T_d^{QR}$ and each path in either tree represents a unique bundle within the tree, $|\mathcal{B}_\ell^{PP}| = |\mathcal{B}_\ell^{QR}|$. And since $\mathcal{B}_\ell^{PP} = \{b \mid anc_i(d) \cup \{\ell_i(d)\} \subseteq b\}$ and $\forall b \in \mathcal{B}_\ell^{QR}$, $anc_i(d) \cup \{\ell_i(d)\} \subseteq b$ because each time a purchase node is moved below a decision node in the transformation a copy is inserted as both the left and right child, then $\mathcal{B}_\ell^{PP} = \mathcal{B}_\ell^{QR}$. Similarly, $\mathcal{B}_r^{PP} = \mathcal{B}_r^{QR}$. Therefore, any decision node in T_d^{QR} represents a choice between the same two sets of bundles, to be made at the same time as its corresponding decision node in T_d^{PP}. And since the hierarchical sequences of decisions are equal, the expected utility of traversing either tree is equal. □

The main result of Theorem 5.1 is that, in order to determine the expected utility of a choice in as purchase procedure, one can instead compute the expected utility of the corresponding choice in the QR-tree. This is preferable since nodes in the QR-tree are sorted by t_q time, a necessary property for the bottom-up computation methods described in this chapter.

Information necessary to rollback solution is stored in the QR-tree as follows:

- For each node n there is a set A_n of *above* values. These are the possible outcomes for the sum of the prices of items ("above" n) to be purchased before n is encountered. For the root r, $A_r = \{z\}$ where z is the amount spent so far in the purchase procedure before this particular decision.

- Let each endpoint contain the two-attribute utility function of the bundle that would be procured on the path from the root.

The goal in rollback solution for this type of tree is, beginning at the leaves and working to the root, to determine for a node n the expected utility of n for each value in A_n. This ensures that each subproblem is solved exactly once. If there is only one such value in A_n, as there will be for the two children of the root, then the expected utility for that value is the expected utility of n. The child of the root with higher expected utility is deemed to be the best choice at the corresponding decision in the purchase procedure.

5.4.2 Defining Certainty at Decision Points

The key to accurate utility expectation computation is to properly consider what information will be available to the decision-maker at each future decision point. The following terms are used to represent this future knowledge.

Definition 5.1 (q-horizon) The *q-horizon* of d, denoted by $qh(d) = \{n \in des(d) \mid t_q(n) < t(d)\}$, is the subset of $des(d)$ for which item prices will be known when d must be resolved.

Definition 5.2 (q-set) Let d be a decision node. The set $qs(d)$ of items that are represented by the nodes in $qh(d)$ is the *q-set* of d.

Definition 5.3 (q-subhorizon) Let d and d' be decision nodes such that d' is an ancestor of d and the path from d' to d has no decision nodes. The *q-subhorizon* of d, denoted by $qsh(d) = qh(d) \cap qh(d')$, is the subset of $qh(d)$ consisting of elements that are also in $qh(d')$.

Definition 5.4 (q-subset) Let d be a decision node. The set $qss(d)$ of items that are represented by the nodes in $qsh(d)$ is the *q-subset* of d.

Definition 5.5 (q-subset-complement) Let d be a decision node. The q-subset-complement $qssc(d) = qs(d) - qss(d)$ of d is the set of items in the q-set but not the q-subset of d.

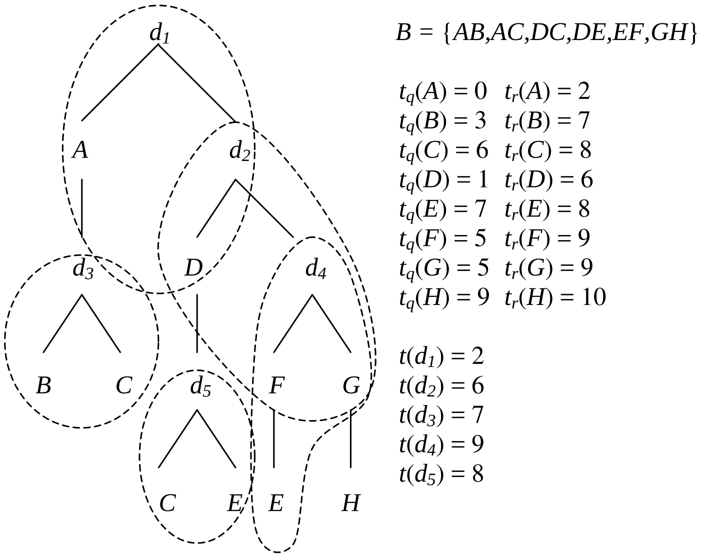

d_1
A
d_2
D
d_3
d_4
B
C
d_5
F
G
C
E
E
H
$B = \{AB,AC,DC,DE,EF,GH\}$
$t_q(A) = 0$ $t_r(A) = 2$
$t_q(B) = 3$ $t_r(B) = 7$
$t_q(C) = 6$ $t_r(C) = 8$
$t_q(D) = 1$ $t_r(D) = 6$
$t_q(E) = 7$ $t_r(E) = 8$
$t_q(F) = 5$ $t_r(F) = 9$
$t_q(G) = 5$ $t_r(G) = 9$
$t_q(H) = 9$ $t_r(H) = 10$
$t(d_1) = 2$
$t(d_2) = 6$
$t(d_3) = 7$
$t(d_4) = 9$
$t(d_5) = 8$

Figure 5.9: Example of a decision tree with q-horizons indicated by dotted lines. For example, for d_4 the q-horizon consists of the nodes representing purchases of F, G and E, and the q-subhorizon consists of the nodes representing purchases of F and G.

The example tree in Figure 5.9 shows the tree from Figure 5.7 with q-horizons indicated by dotted lines. Note that Algorithm 5.2 constructs QR-trees in such a way that, for any decision node d and purchase node n, if $n \in qh(d)$ then $n' \in qh(d)$ for all purchase nodes n' on the path between d and n. So all elements of a q-horizon are connected.

5.5 Solving the QR-tree Using Discrete Approximation

This section describes the technique to be used with a tree for which all outcomes are characterized by discrete probability measures. Note that since all possible outcomes are analyzed, even though all probability measures are discrete, it may be necessary to further approximate to a small number (perhaps no more than 3) of outcomes to ease the computational burden. To solve the tree bottom up, each node must be solved so that the expected utility can be ascertained (or estimated) for any occurrence above it. Such occurrences include not only outcomes for money spent before the node is encountered, but also outcomes of the items represented in the q-subhorizon of decision nodes, since they are part of ancestor q-horizons and therefore are simulated when these ancestor nodes are evaluated. Solution is done by computing 1) a *utility projection function* for each purchase node, and 2) a *q-subset-mapping* for each decision node. Each of these two types of functions takes a state at a node, consisting of the amount already spent at that point and, for decision nodes, the costs of items in the q-subset. Since the actual state is known for each child node at the root of the QR-tree, the function for each child is a constant representing the expected utility of that choice.

5.5.1 Utility Projection Functions

Let n be a purchase node or an endpoint in a QR-tree. The set A_n of *above* values for n is the set of all possible outcomes for the sum of 1) the cost of items represented by proper ancestor purchase nodes of n and 2) the amount already spent on items procured before the QR-tree was built. A utility projection function $up_n : A_n \rightarrow \Re$ is a function that maps a value $a \in A_n$ to the expected utility of buying n, given that a is the amount spent before n is encountered.

If n is an endpoint, then the utility projection function is simply the two-attribute utility function for the bundle procured above n. That is, if b is the particular bundle procured, then for each $a \in A_n$,

$$up_n(a) = u(b, a) \tag{5.2}$$

Otherwise, let n be a purchase node. Note that if a purchase node resides in a q-horizon, it is not considered separately but rather as part of the information available for a particular decision, and an individual utility projection function does not need to be computed. Thus only purchase nodes not residing in any q-horizon are considered here. Consider the following notation used to represent the set of joint price outcomes for items in a QR-tree. Let $I' \subseteq I$ be a subset of the items represented in the QR-tree, let $N_{I'}$ be the set of purchase nodes that represent an $i \in I'$, and let $K(I')$ be the set of joint price outcomes for items in I' where each element $k : N_{I'} \rightarrow \Re$ of $K(I')$ is a function that assigns a price to each purchase node in $N_{I'}$. Let n be a purchase node that does not reside in a q-horizon. Let $K(\{i\})$ be the set of possible outcomes for the cost of the item i represented by n, let $p : K(\{i\}) \rightarrow \Re$ be the probability measure on the outcomes, and let n' be the child of n. For each $a \in A_n$,

$$up_n(a) = \sum_{k \in K} up_{n'}(a + k(n))p(k) \tag{5.3}$$

Note that $up_{n'}(a + k)$ exists since $(a + k) \in A_{n'}$.

5.5.2 q-subset-mappings

Computing expected utilities for decision nodes is much more complicated, since future information must be considered. At the time a decision node d must be resolved, the prices of all items in the q-set of d will be known. Therefore, in order to compute the expected utility of d for some above value a, all possible joint outcomes of the prices of items in the q-set should be considered. For each outcome, the expected utility of d is taken as the choice with the higher expected utility, since we assume that the buyer will always make choices that maximize expected utility, given the available information. Making the computation even more complicated is the fact that the q-horizon of d may contain other decision nodes. So, at decision time, the purchaser will know part of the information that will be known at these future decisions. This problem can be overcome, however, by carefully considering what information known at a decision node will also be known at ancestor decisions, and properly passing that information up during solution. This is the purpose of the q-subhorizon.

Since we need to know the expected utility of each choice at a decision node d given the amount spent so far and the costs of items represented in the q-subhorizon, if there is a descendent decision node d' where $qh(d) \cap qh(d') \neq \phi$, then we need to be able to determine the expected utility of d' given item prices represented by nodes above d' and in the q-subhorizon of d'. For this reason, a q-subset-mapping is computed. The q-subset-mapping $qssm_d$ for a decision node d is a function that maps a joint outcome for prices of items in the q-subset of d to a utility projection function. The utility projection function in turn maps the above amount to the expected utility. Consider determining the q-subset-mapping for d_1 in the partial QR-tree given in Figure 5.10. For a joint outcome k_{qs} for the items in $qss(d_1) = \{A, D\}$, the utility projection function for d_1 is computed as follows: Given an above value $a \in A_{d_1}$, for each joint outcome k_{qssc} for the items in $qssc(d) = qs(d_1) - qss(d_1) = \{B, C\}$, the expected utility for each choice at d_1 is computed: For a path below the decision node, if the final node in the q-horizon is a purchase node (as is the case with node B in the left path below d_1 in the example) with child n', then the expected utility of that choice is $up_{n'}(x)$, where x is the sum of the prices of all items above n' given a, k_{qs} and k_{qssc} (if n' is a decision node, since its q-horizon must be empty then $up_{n'} = qssm_{n'}$). Otherwise the path reaches a decision node before the q-horizon is exited, as is the case with d_2. In this case, given a, k_{qs} and k_{qssc}, the outcome for prices of items in $qss(d_2)$ is entered into $qssm_{d_2}$ to determine the appropriate utility projection function to use for d_2, and the sum of item prices above d_2 are used with the utility projection function to determine the expected utility.

Formally, let $K(qss(d))$ be the set of joint outcomes of items in the q-subset of d, let $K(qssc(d))$ be the set of joint outcomes of items in the q-subset-complement of d and let $p : K(qss(d)) \rightarrow \Re$ be the probability measure. Let the q-subset-mapping $qssm_d$ for d be a function mapping each outcome $k \in K(qss(d))$ to a utility projection function up_{dk}. With this q-subset-mapping, the expected utility of any decision node d can thus be computed for any outcome in $K(qss(d))$ and above value in A_d. Note

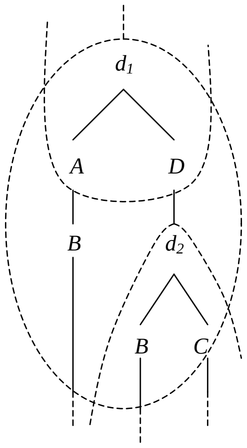


Figure 5.10: Partial QR-tree.

that if $qss(d) = \phi$ then there is only one utility projection function, so $qssm_d = up_d$.

The q-subset-mapping is computed as follows. For each outcome $k \in K(qss(d))$, a utility projection function up_{dk} is constructed such that for each $a \in A_d$

$$up_{dk}(a) = \sum_{k' \in K(qssc(d))} \max\{\omega(a, k + k', \ell(d)), \omega(a, k + k', r(d))\} p(k') \tag{5.4}$$

where $k + k'$ is the concatenation of the outcomes given by k and k' for the set of items $qss(d) \cup qssc(d) = qs(d)$, $p(k')$ is the probability of k' occurring, $\ell(d)$ and $r(d)$ are the left and right children of d, and ω is a well-defined function where $\omega(a, k, n)$ for a node n is computed as follows: If n is a purchase node and $k(n)$ exists (i.e. k assigns an outcome to n's item), then $\omega(a, k, n) = \omega(a + k(n), k, n')$ where n' is the child of n. If $k(n)$ does not exist (as is always the case with decision nodes and endpoints), $\omega(a, k, n) = up_n(a)$ where, if n is a decision node, $up_n = qssm_n(k')$ where k' is the outcome for items in $qss(n)$ consistent with the item outcomes given by k. Informally, $\omega(a, k, n)$ is the expected utility of n for a given $a \in A_n$ and a given outcome k for some of the descendents of n.

5.6 Solving the QR-tree using Monte Carlo Simulation

In the method described in section 5.5, the problem of solving trees that include continuous random variables is handled by using some discrete approximation, such as Pearson-Tukey. While the literature shows that the Pearson-Tukey method gives very favourable results when used to approximate a variety of distributions [PBF91], approximating an entire density function with only three points can only achieve a limited level of accuracy. With the current power of computing, using simulation to solve difficult mathematical problems such as those seen in investment decision-making, physics and statistics, is becoming more and more common (see for example [BAD03, SDP03, Lub01]). In this section, the use of Monte Carlo methods to solve decision trees during rollback is investigated. Here, the discussion is based on solving QR-trees, but a similar theory can be developed to solve general decision trees.

With Monte Carlo simulation, several independent simulations of the variable(s) in question are run, and the average result is taken as the mean. Typically, for a given sample variance s^2, a number n of trials will be required so as to reduce the standard error $\sqrt{\frac{s^2}{n}}$ below a specified threshold. This variance can be reduced, thus reducing n, by using the antithetic variate sampling technique as described in section 4.5.3.

With this sort of decision tree, Monte Carlo simulation can be used to more accurately determine the expected utility of a node for a given above value. As in the discrete case, computation must be done in a bottom-up manner. A top-down solution of the entire tree is infeasible since too much simulation is required. Consider attempting a top-down solution of the example in Figure 5.9. To determine the expected utility of proceeding to decision node d_2, for example, the information to be known at that decision time (namely the prices of D, F, G) must be simulated. For each simulation, the expected utility of each of the left choice (D) and right choice

(d_4) must be computed and the higher noted, since this is the choice the decision-maker would make if this simulation represented actual outcomes. To determine the expected utility of the right choice d_4, several simulations of the extra information known at d_4 (namely E) need to be run for the given values of F and G. If d_4 had any descendent decisions, then for each of these simulations, several further simulations would be needed, and so on. For any decision node d, let x be the required number of simulations of item costs that will be known at decision time $t(d)$ in order to compute the expected utility of d. Then if h is the decision node height of d (the maximum number of decision nodes on a path from d to a leaf), then the number of simulations required to compute the expected utility of d is $O(x^h)$. Since x can be quite large (say 10,000 - 100,000 for reasonable accuracy), then x^h can get unmanageably large for even small h.

It is desirable to utilize a bottom-up approach for computing these utilities that ensures that the number of simulations required grows linearly with the number of nodes in the tree. The goal is, for any node n in the tree, to be able to estimate the expected utility of n for any outcome for the nodes above n in the tree without doing any further simulation on n's subtree. Unfortunately, two problems arise when trying to use simulation bottom-up: 1) for a node n, the set A_n is neither discrete nor finite, and 2) for a decision node d the set $K(qss(d))$ of outcomes for the q-subset items is neither discrete nor finite. The next subsection gives a discussion on how the utility projection functions for purchase nodes and endpoints can be approximated. Following that, three different techniques for handling this problem associated with the q-subset-mappings for decision nodes is presented.

To denote results obtained through the use of Monte Carlo simulation, consider the following notation: Let $MC(I, \alpha(k), \varepsilon)$ be a function that takes a set I of items and a function $\alpha : K(I) \rightarrow \Re$, and returns the average result of $\alpha(k)$ for several independently generated random outcomes $k \in K(I)$, within a standard error of ε.

5.6.1 Utility Projection Functions for Purchase Nodes and Endpoints

Let n be a purchase node or an endpoint in a QR-tree, with above values A_n. As in the discrete case, if n is the endpoint for the path on which bundle b is procured, then the utility projection function for n is $up_n(a) = u(b, a)$. If n is a purchase node and A_n is infinite, then up_n likely cannot be computed exactly over the entire domain. However, since up_n is a monotone strictly decreasing function (higher cost leads to lower expected utility), if we have actual values for some points then we know that the values at other points are tightly constrained and therefore can be accurately estimated. So a rather simple solution to the problem of computing a utility projection function for n is to choose a few above values $A'_n \subset A_n$, compute the expected utility up'_n for each chosen value, and then fit a curve, thus specifying an estimated up_n for all of A_n. For this thesis, the set of points chosen is $A'_n = \{x \mid P(X < x)$ is a multiple of .05 and $.05 \leq P(X < x) \leq .95\}$.

For a purchase node n with child n', for each $a \in A'_n$, the expected utility $up'_n(a)$ is computed by

$$up'_n(a) = MC(\{i_n\}, up_{n'}(a + k(n)), \varepsilon) \tag{5.5}$$

where i_n is the item represented by n. Testing shows that using regression to find a degree-three polynomial representing up_n works quite well.

Now the expected utility of n can be predicted for any above value on the continuous scale. Note that if n is a purchase node that resides in a q-horizon for a decision d, then it is not considered separately but rather as part of the information available at d. Thus its utility projection function is irrelevant and not computed.

5.6.2 Functions for Decision Nodes: the Restricted q-horizon Method, the q-subset Discretization Method and the Classification Tree Method

Computing q-subset-mappings for decision nodes is difficult when the space of outcomes for the q-subset items is continuous. If there are only one or two items in the q-subset, then techniques such as regression can be used to estimate a function, given the utilities for a few chosen values. However, for more items this technique can be time consuming and produce very inaccurate results. In this section, three different Monte Carlo techniques for handling decision nodes during rollback solution of the QR-tree are presented.

The Restricted q-horizon Method

One way to solve the problem associated with computing q-subset-mappings is to simply ignore the q-subhorizons in the QR-tree. This can be done be putting a restriction on the q-horizon that it cannot extend past any descendent decision node, resulting in empty q-subsets for all decision nodes. If a decision node's q-subset is empty, then the q-subset-mapping reduces to a single utility projection function. See Figure 5.11 for an example of such a transformation. The utility projection function up_d, approximated by up'_d (as described in section 5.6.1), for a decision node d is then computed as follows: For each $a \in A'_n$,,

$$up'_d(a) = MC(qs'(d), \max\{\omega(a, k, \ell(d)), \omega(a, k, r(d))\}, \varepsilon) \tag{5.6}$$

where $qs'(d)$ is the new q-set of d after the q-horizons are restricted, $\ell(d)$ and $r(d)$ are the left and right children of d, respectively, and ω is as defined in section 5.5.2.

While this method will most definitely result in some degree of inaccuracy since not all information is being considered at the various decision points, this inaccuracy may be insignificant compared to the accuracy gained by choosing to use simulation

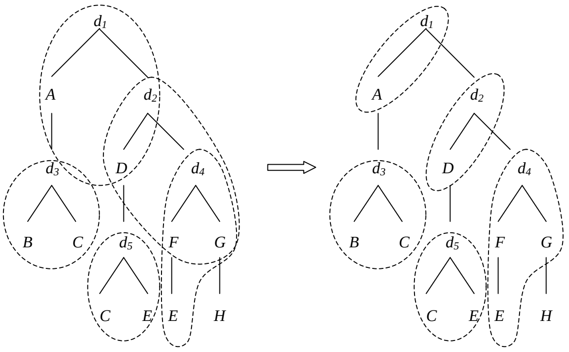


Figure 5.11: Example of restricting the q-horizons.

over some discrete approximation, regression technique, etc. One potential problem with this method, as is, is that it may achieve even worse results than the greedy method of decision making that always chooses to buy an item if it is in the bundle with greatest expected utility. Consider the example decision tree in Figure 5.12. There are five bundles AB, AC, D, E and F and a decision has to be made on whether to buy A. The q-sets are $qs(d_1) = \{A\}$, $qs(d_2) = \{B, C\}$, $qs(d_3) = \{D, E\}$ and $qs(d_4) = \{E, F\}$, but for the purpose of computing the expected utility of not buying A, the restricted q-horizon method considers $qs(d_3)$ to be only $\{D\}$. Let it be the case that, given the information known at the time d_1 is to be decided, the bundle with maximum expected utility does not include A, but the restricted q-horizon method determines that the expected utility of buying A is higher than that of not buying A. So, a purchaser using this more complex method would buy A, while a purchaser using the greedy method would not. However, it is possible that the greedy purchaser has a higher expected utility, since the complex method may have underestimated the expected utility of not buying A, since it ignored the fact that the cost of E will be known at the time decision d_3 is to be made.

This possibility can be eliminated with a modification to the restricted q-horizon method that incorporates the utility that a greedy purchaser would expect to achieve in certain regions of the tree. To do so, the method for determining the expected

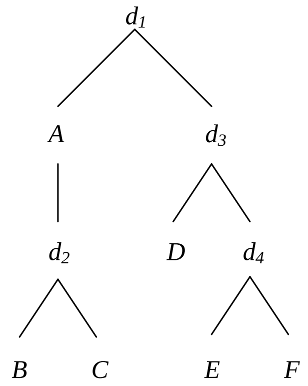


Figure 5.12: Tree for the example situation where using the greedy procedure would achieve higher expected utility than using the restricted q-horizon method.

utility of the greedy procedure must be demonstrated. This method uses only top-down simulation, since no expected utilities of future decisions and purchases are ever needed. The simulation process works as follows. For each random outcome of all items in the tree, the course of action from the root to a leaf for that particular outcome is determined. At each decision point, given the information that will be known at the time, the expected utility of every potential bundle purchase is calculated, and the choice that leads to the bundle whose expected utility is maximized is taken. Execution is continued until a leaf node is reached, resulting in the purchase of some bundle at some total cost, and the utility of this bundle purchase is noted. The average of these utilities is taken as the expected utility.

To incorporate this into the restricted q-horizon method, a function is developed that performs the task described above.

Definition 5.6 (q-region) A subgraph R of a QR-tree T is a q-region iff the root r of R is a decision node with $qsh(r) = \phi$, $qsh(d)$ is non-empty for all decision nodes $d \in R \setminus r$, and for all nodes $n \in T$, $n \in R$ iff there exists a decision node $d \in R$ such that $n \in qh(d)$.

For the sake of intuition, one can view a q-region as the union of intersecting q-horizons. That is, every q-horizon is a subgraph of exactly one q-region, and the union of two q-horizons Q and Q' is a subgraph of a q-region iff Q intersects a subgraph of a q-region that includes Q'. For example, consider the QR-tree given in Figure 5.9. The q-regions in this tree are indicated by dotted lines in Figure 5.13.

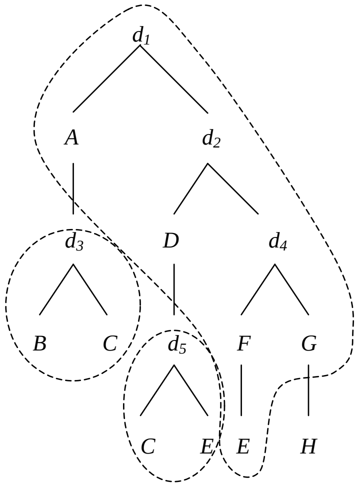


Figure 5.13: The QR-tree given in Figure 5.9 with q-regions indicated by dashed lines.

Let $greedy(R, a, k)$ be a function that takes q-region R in a QR-tree, an above value a, and an outcome k for the items in R. The function returns the expected utility of the root of R for above cost a if the greedy decision procedure were to be used through R, given utility projection functions for the nodes in the QR-tree below R. This function is incorporated into utility projection function computation for the restricted q-horizon method as follows. Let d be a decision node, and let $qs'(d)$ be the q-set of d as used by the restricted q-horizon method. For each $a \in A'_n$,

$$\begin{aligned} up'_d(a) = \;& \max\{MC(qs'(d), \max\{\omega(a,k,\ell(d)), \omega(a,k,r(d))\}, \varepsilon), \\ & \qquad MC(I_R, greedy(R,a,k), \varepsilon)\} \end{aligned}$$

(5.7)

where R is the q-region rooted at d and I_R is the set of items represented by purchase nodes in R. If d is not the root of a q-region, then $MC(I_r, greedy(R, a, k), \varepsilon) = 0$. Since the true expected utility of d for the above value a must be at least as high as $MC(qs'(d), \max\{\omega(a, k, \ell(d)), \omega(a, k, r(d))\}, \varepsilon)$ and $MC(I_R, greedy(R, a, k), \varepsilon)$ (since both measures partially ignore future available information and are therefore underestimates of true expected utility), then $up'_d(a)$ is taken as the maximum. Using this method to make decisions is guaranteed to yield an expected utility at least as high as using greedy purchasing for any instance (see section 6.2.1 for the proof).

The q-subset Discretization Method

While the restricted method provides a nice guarantee, it may not give a very accurate estimate. A better idea might be to compute the q-subset-mapping in a manner similar to that used in the discrete case. With this approach, the q-subset-mapping applies only to a few outcomes for the items in the q-subset. This results in a mixture of simulation and discrete approximation in the solution process. Here the Pearson-Tukey three-point approximation (see section 5.3.2) is incorporated. When computing q-subset-mappings, q-subset items are assumed to have only three possible outcomes. An above mapping for a decision node with m q-subset items will thus be computed for each of the 3^m joint outcomes (referred to as PT-outcomes). This is a reasonable number for fairly small m. For each PT-outcome for the items in a decision node's q-subhorizon, the utility projection function is computed by using simulation for the items in the q-subset-complement that do not reside in another decision node's q-subset, and PT-outcomes for the remaining q-subset-complement items.

The formal technique for determining the q-subset-mapping is now given. Let d be a decision node. The q-subset-mapping $qssm_d$ for d maps each PT-outcome for $qss(d)$ to an utility projection function, which is computed as follows: Let d be a decision node with d_ℓ and d_r the first left and right descendent decision nodes of

d respectively (if they exist), let $qss'(d) = qss(d_\ell) \cup qss(d_r)$ be the union of the q-subsets for d_ℓ and d_r (if either of d_ℓ or d_r does not exist then treat their q-subset to be empty), and let $sim(d) = qss^c(d) - qss'(d)$ be the subset of the q-subset-complement of d to be simulated. The q-subset-mapping $qssm_d$ maps a PT-outcome k' for the q-subset of d to a utility projection function up_d, approximated by up'_d (as described in section 5.6.1), where, for an $a \in A'_d$,

$$up'_{dk'}(a) = \sum_{k'' \in K(qss'(d))} p(k'') \cdot MC(sim(d), \max\{\omega(a, k + k' + k'', \ell(d)), \omega(a, k + k' + k'', r(d))\}, \varepsilon) \tag{5.8}$$

where $k + k' + k''$ is the concatenation of the outcomes given by k, k' and k'' for the set of items $sim(d) \cup qss(d) \cup qss'(d) = qs(d)$, $p(k'')$ is the probability of k'' occurring according to the PT approximation, $\ell(d)$ and $r(d)$ are the left and right children of d, and ω is as defined in section 5.5.2.

The Classification Tree Method

One problem with the q-subset discretization method is that, because of the usage of an approximation method, it is difficult to ascertain anything theoretically about its results. While all methods described in this thesis give only estimates of expected utility, it is useful to know theoretical bounds on their performance. In particular, a method might be deemed more favourable if it was guaranteed to give an underestimate of the true expected utility. This is true of the restricted q-set method. However, the reason for this guaranteed underestimate is that certain information is ignored. This is a serious drawback as it can cause significant inaccuracy.

The classification tree method provides a guaranteed underestimate of the true expected utility, and makes use of all information available in order to make a reasonably accurate estimation. The initial stage of the method performs the same calculations as the q-subset discretization method. That is, at each decision node d, the utility projection function is calculated for each joint PT-outcome of the items in $qss(d)$ to produce a q-subset-mapping. The second stage involves the creation of a *decision tree* [Qui86] for each decision node. This is a decision tree in the *machine learning* sense. Since the term "decision tree" is already used in this thesis (as in the decision analysis literature), this tree is hereafter referred to as a *classification tree.* The purpose of a classification tree is to provide a structure for classifying new data based what has been learned from the training data. When computing a q-subset-mapping, the machine is essentially learning what the utility projection functions are for a few joint outcomes for the q-subset prices. The classification tree shows how to determine under which of these utility projection functions any other joint outcome should be classified.

For a decision node d, a classification tree CT_d is built with height $|qss(d)| + 1$ where each interior node has three children. Each level of nodes represent an arbitrary but unique element of $qss(d)$, except for the leaves which represent utility projection functions. Each edge represents a PT outcome of the cost of the item represented by the node from which the edge emanates. Each leaf contains the utility projection function that would be used if the items represented by nodes on the path from the root have the cost outcomes represented by the edges on the path from the root. The utility projection functions are computed for each PT outcome of the items in $qss(d)$ in the manner used by the q-subset discretization method.

Example 5.3 Let d be a decision node with $qss(d) = \{A, B\}$, where the prices for A and B are normally distributed, each with mean 10 and standard deviation 1. For simplicity, let the set of above values $|A_d| = 1$, so each utility projection function in the q-subset-mapping is a constant. Consider the q-subset-mapping given in Table 5.3. The classification tree for this data is given in Figure 5.14.

The classification tree is used as a mechanism for classifying new data. Given any

A	B	up_d
8.355	8.355	u_1
8.355	10	u_2
8.355	11.645	u_3
10	8.355	u_4
10	10	u_5
10	11.645	u_6
11.645	8.355	u_7
11.645	10	u_8
11.645	11.645	u_9

Table 5.3: The q-subset-mapping for Example 5.3

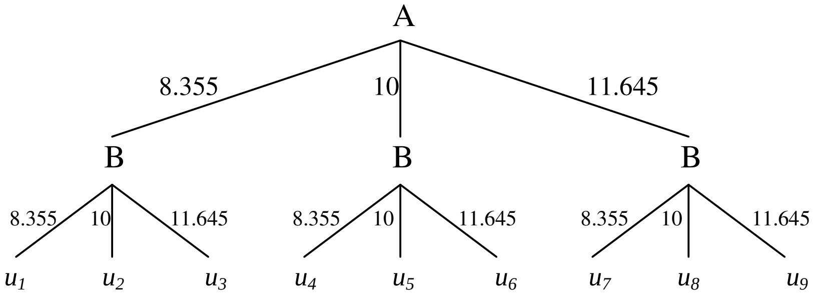


Figure 5.14: Classification tree for Example 5.3

independent joint outcome for the q-subset item costs, the path in the classification tree that represents this closest joint outcome is followed, and the outcome is "classified" under the utility projection function reached. Thus, this utility projection function is used to compute the expected utility for this outcome. To do this, the continuous space of outcomes for each variable in the tree has to be divided into three discrete regions. This is necessary in order to clearly define to which child to move during tree traversal. Staying consistent with the theory behind the Pearson-Tukey approximation, the following values are chosen: Let x be a random outcome for the cost of item in the classification tree with mean μ and standard deviation σ,

- if $x < \mu - .9\sigma$, proceed to the leftmost child.
- if $\mu - .9\sigma < x < \mu + .9\sigma$, proceed to the center child.
- if $\mu - .9\sigma < x$, proceed to the rightmost child.

As a result, the value of μ will be chosen with .63 probability (since the probability that the outcome of a normally distributed random variable will lie between $\mu - .9\sigma$ and $\mu + .9\sigma$ is .63), and the values $\mu - 1.645\sigma$ and $\mu + 1.645\sigma$ will each be chosen with .185 probability, which is consistent with Pearson-Tukey. The discretized tree for the above example is given in Figure 5.15. The resulting estimate $ct_d(k)$ for an outcome k on the variables in the classification tree for a decision node d is the utility projection function labeling the leaf node reached by traversing the tree in the manner described above. For a particular above value a for d, the estimated expected utility of d is $ct_d(k)(a)$.

With a classification tree for every decision node in the QR-tree, the expected utility of a decision node can be estimated for any above value and q-subset outcome. This means that the simulation can be performed on the tree *top-down*. This is the final stage in the method. For each simulation, an outcome is chosen for all items in the QR-tree. Execution through the tree beginning at the root is then simulated.

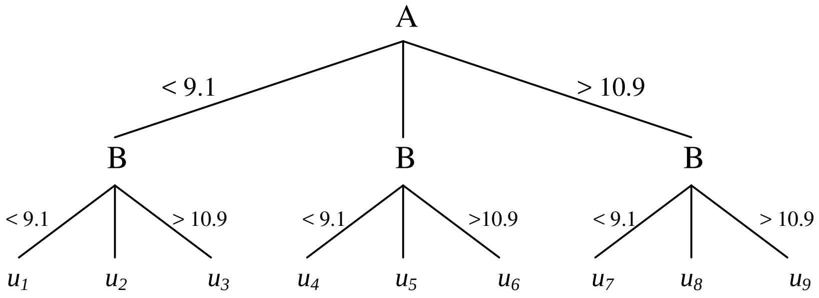


Figure 5.15: Discretized classification tree for Example 5.3

When a purchase node is encountered, the item is assumed to be purchased at the price for the given outcome and execution moves to the child. When a decision node is encountered, the expected utility of proceeding to each child is estimated, and the choice associated with the higher expected utility is chosen. The estimated expected utility $E_d(n, a)$ of a child node n of a decision d with above value a for a joint outcome k on the items is computed by:

$$\begin{aligned} E_d(n, a) &= E_d(n', a + k(n)) && \text{if } n \text{ is a purchase node in } qs(d) \\ &= up_n(a) && \text{if } n \text{ is a purchase node not in } qs(d) \text{ or an endpoint} \\ &= ct_n(k)(a) && \text{if } n \text{ is a decision node} \end{aligned}$$

where n' is the child of n. Thus when a decision node d is encountered during simulation where a has been spent, proceed to $\ell(d)$ if $E_d(\ell(d), a) > E_d(r(d), a)$ and proceed to $r(d)$ otherwise. When a leaf is reached, the utility achieved by the bundle purchased is noted. The average of these achieved utilities after an appropriate number of simulations is taken to be the expected utility.

5.7 Other Issues

This section points out a few issues that are left out of the above discussion for the sake of simplicity in either the approach or the explanation.

5.7.1 "Garden Paths"

It is possible that the best choice at a decision point is a course of action that will result in a previous purchase being wasted. This can occur if the buyer was "fooled" at a previous decision point, and made a choice where the outcomes of subsequent item prices turned out to be unexpectedly high. The price outcomes of future items would be so high, in fact, that the purchaser is better off abandoning some of the items already purchased so that different bundles can be pursued. Another cause of this could simply be that the buyer's situation changed after one or more purchases had already been made. A purchaser that makes an ill-fated choice such as this can be referred to as going "down the garden path" in the purchase procedure.

While the model described in this chapter allows the purchaser to backtrack in such situations so that more attractive bundles can be pursued, expected utility estimation does not, however, factor in this possibility. Consider the purchase procedure tree in Figure 5.16 (a). If the buyer chooses A, the purchase procedure tree at time 3 will be the as in Figure 5.16 (b) (given that no new offers have been made in the interim). The second tree shows that, even though A has been purchased, buying C and D is still an option.

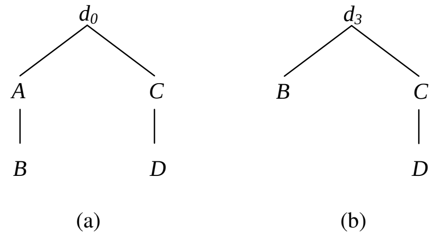


Figure 5.16: Purchase procedure tree before and after A is purchased

Given the fact that buying A and B was initially expected to be better than buying C and D, it would be typically be a low-probability occurrence if, at time 3, buying B by itself is worse than buying C and D. For this reason, this possibility is not considered when the expected utility of buying A is estimated. Moreover, considering every one of these possibilities would cause QR-trees to be too large for larger purchase procedure trees. Accuracy would need to be sacrificed to maintain reasonable computing times, thus defeating the purpose.

However, if one can predict a few specific cases beforehand where such an occurrence is a fairly reasonable possibility, some measures can be taken to have it considered. For example, the situation described above would have been considered more likely if A's mean price was very small and the variance of B's price very high. If the expected utility of each choice was very close, since the ultimate utility of AB was dependent almost exclusively on the outcome of B's price, this situation could have been more predictable and the option of pursuing CD might thus be factored into the QR-tree at time 0 (see Figure 5.17).

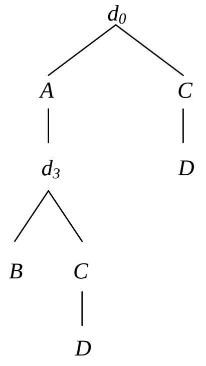


Figure 5.17: Purchase procedure tree before A is purchased that considers "garden path" possibility

5.7.2 Minimum Allowable Utility

It is possible that the buyer might have a lower bound on the acceptable level of utility to be achieved. That is, he/she would prefer to end up with no purchases than to achieve a utility less than some minimum. This is very difficult to enforce in the setting where partial bundle purchases are allowed, since some items may be purchased before it is realized that the minimum utility is impossible to achieve. It could still be encoded in the decision process that, for example, no purchase should be made if the expected utility of that purchase is less than some minimum, unless the buyer has already spent x dollars.

At first glance, it seems as though this type of decision-making does not fit well in the context of the problem. In this thesis, the goal of the buyer is simply to maximize expected utility. If the main goal was to achieve a certain minimum level of utility, then at each decision point, one would want to choose the path such that the probability of achieving this utility is greater. However, while this seems to be a different decision-making criterion, if this is the desire of the buyer then such a desire will naturally be reflected in the buyer's two-attribute utility function. If the utility function is properly assessed, then the expected utility maximization methods presented in this thesis will naturally have the desired effect. Consider the extreme case where the buyer only cares about achieving a certain level of utility, and given that this level is met, is indifferent to all outcomes. Then the utility function will reflect this desire since it must be that every bundle purchase that does not meet this minimum level has utility 0 and every purchase that does meet this level has utility 1. Maximizing expected utility in this case now has the effect of maximizing the likelihood of achieving this minimum utility.

5.7.3 Item Cost Dependencies

If two nodes inside a q-horizon represent items for which cost outcomes are not entirely independent, or even represent the same item for that matter, an accurate result can be obtained as long as the randomness of the outcomes is properly modeled to be consistent with the dependence. However, this cannot be done if the nodes are not in the same q-horizon, for the reason that they are not simulated together.

This becomes a problem if the two nodes are on the same path in the decision tree. Let n_1 and n_2 be two such nodes representing the purchase of items i_1 and i_2, respectively, where n_1 is an ancestor of n_2. If the cost of i_2 depends on the earlier outcome of the cost of i_1, then this dependency will be difficult to model in the tree. The rollback solution method described simply calculates the expected utility of a node for outcomes of the sum of item costs above it, without any consideration for the costs of particular items above it. To include the effect that the cost of i_1 will have on the cost of i_2, the utility projection function for n_2 would need to consider both the cost of i_1 as well as the sum total of all item costs above n_2. That is, the utility projection function would be the function $au_{n_2} : A \times K(\{i_1\}) \rightarrow \Re$ where A is the set of above values and $K(\{i_1\})$ is the set of outcomes for the cost of i_1. While the methods described in this thesis can be extended to handle this, since it requires much more computation, especially if there are many such influential items, it is recommended that this be done only if the dependence is strong enough to warrant such extra work.

Chapter 6

Results and Analysis

This chapter provides both theoretical and experimental results on the performance of the methods described in the previous chapters. In the complete bundle purchasing setting, the improved decision process is proven to yield higher expected utility than the naïve decision process. Results on the magnitude of the expected increase and how this increase can be approximated is also given. For the setting in which partial bundle purchases are permitted, four methods are examined: the greedy method, the restricted q-horizon method, the classification tree method, and the q-subset discretization method. Three major results are given. First, the restricted q-horizon method is shown theoretically to yield a higher expected utility than the greedy method. Next, the estimate given by the classification tree method is proven to be always less than or equal to the true utility expected when using this method throughout the purchase procedure, and is demonstrated to be a tight lower bound. Finally, the q-subset discretization method is shown experimentally to provide the most accurate estimates of all methods. The performance of the antithetic variate technique used in Monte Carlo simulation during testing is also assessed.

Note that, while all estimates produced by methods in this thesis are partially generated using simulation, the results given in proofs are based on the assumption that an appropriate number of simulations are performed to achieve the necessary degree of accuracy.

6.1 Complete Bundle Purchases

6.1.1 Proofs

Recall the naïve and improved decision processes described in chapter 4. This section provides a few proofs that show, in the setting where only complete bundle purchases are made, 1) that the buyer always has a higher expected utility if the improved process is used over the naïve process, 2) the expected increase in utility if the buyer chooses to use the improved process, in the situation where there is one future comparison set, and 3) that the increase derived in 2) is a lower bound when there is more than one future comparison set.

For a comparison set cover $csc(\mathcal{B})$ of bundles $\mathcal{B}$, let $j = \max\{E[\tilde{u}(b)] \mid b \in \mathcal{B}\}$ and $k = \max\{E[\tilde{u}(CS)] \mid CS \in csc(\mathcal{B})\}$.

Lemma 6.1 *$j \leq k$.*

Proof. Let CS_j be a comparison set containing a bundle b such that $E[\tilde{u}(b)] = j$. Since the expected utility of choosing from a number of bundles must be at least as high as the expected utility of any one of the bundles, $E[\tilde{u}(CS_j)] \geq j$. So $k = \max\{E[\tilde{u}(CS)] \mid CS \in csc(\mathcal{B})\} \geq E[\tilde{u}(CS_j)] \geq j$. □

Lemma 6.2 The expected utility E_{im} when using the improved decision process is greater than or equal to k.

Proof. By induction on the size of $csc(\mathcal{B})$.

Base Case. Let $|csc(\mathcal{B})| = 1$. Then $E_{im} = k$.

Induction. Let $csc(\mathcal{B})$ be of arbitrary size, let CS_i be the current comparison set, and let b be the bundle in CS_i available at cost z such that $u(b, z)$ is maximized in

CS_i. Assume that the expected utility when using the informed decision process for $csc(\mathcal{B}) \setminus CS_i$ is at least $\max\{E[\tilde{u}(CS)] \mid CS \in \ csc(\mathcal{B}) \setminus \ CS_i\}$. Consider the two outcomes:

1. $u(b, z) \geq k$. Buy b and achieve a utility $\geq k$. So $E_{im} \geq k$.

2. $u(b, z) < k$. Then $k = \max\{E[\tilde{u}(CS)] \mid CS \in csc(\mathcal{B}) \setminus CS_i\}$. Since b would not be purchased in this case (and thus nothing from CS_i is purchased, so it expires) and by induction the expected utility for $csc(\mathcal{B}) \setminus CS_i$ is at least $\max\{E[\tilde{u}(CS)] \mid CS \in csc(\mathcal{B}) \setminus CS_i\}$, then $E_{im} \geq k$. □

Theorem 6.1 Let E_{im} and E_{na} denote the utilities expected when using the improved and naïve decision processes, respectively. Then $E_{im} \geq E_{na}$.

Proof. By induction on the size of $csc(\mathcal{B})$.

Base Case. Let $|csc(\mathcal{B})| = 1$. Both processes will choose the bundle purchase with the highest utility, so $E_{im} = E_{na}$.

Induction. Let $csc(\mathcal{B})$ be of arbitrary size, let CS_i be the current comparison set, and let b be the bundle in CS_i available at cost z such that $u(b, z)$ is maximized in CS_i. Assume that the expected utility when using the improved decision process is greater than or equal to that when using the naïve decision process for $csc(\mathcal{B}) \setminus CS_i$. By Lemma 6.1 there are three cases:

1. $k \leq u(b, z)$. Both processes would choose b so $E_{im} = E_{na}$.

2. $j \leq u(b, z) < k$. The naïve process would choose b and the improved would not. So $E_{na} = u(b, z) < k$. Since, by Lemma 6.2 $E_{im} \geq k$, $E_{im} > E_{na}$.

3. $u(b, z) < j$. Neither process would choose b, and would thus allow CS_i to pass. By induction, $E_{im} \geq E_{na}$. □

While it is necessarily the case that expected utility improves when the improved decision process is used, one may wonder if the improvement justifies the extra work required for computation. For this reason, the increase in utility that one would expect to achieve when using the improved process over the naïve process for a simple case is derived. Here we consider the situation where there is a bundle b that is about to expire, one comparison set CS, and there is no interdependence between the utilities of b and CS. We also assume that $ci(CS)$ begins after $t_r(b)$ so that nothing is known for certain about CS. The analysis is then extended to the more general case where there are many future comparison sets, and a lower bound on the expected increase is proven.

For the case where there is only one future comparison set, let $j = \max\{E[\tilde{u}(b')] \mid b' \in CS\}$ and $k = E[\tilde{u}(CS)]$. Also, let $p_b : \Re \rightarrow \Re$ be a probability density function that maps utilities for the purchase of b to probabilities, and let

$$A_1 = \int_0^j p_b(x)\,dx \qquad A_2 = \int_j^k p_b(x)\,dx \qquad A_3 = \int_k^1 p_b(x)\,dx$$

For the sake of intuition, let an example probability density function for the outcome of the utility of purchasing b be as depicted in Figure 6.1. Example points for j and k are also displayed. Using these points, the area under the curve is divided into three regions. A_1, A_2, and A_3 represent the areas of these regions.

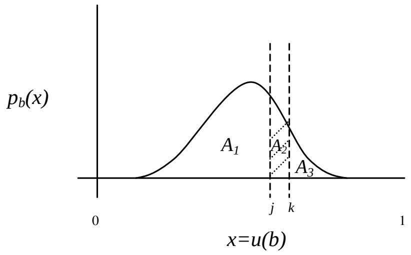


Figure 6.1: Probability density function for the utility of purchasing b

Theorem 6.2 The expected increase in utility to be achieved by using the improved decision process is

$$\int_j^k (k-x)p_b(x)\,dx \tag{6.1}$$

where b is the bundle that will expire first, CS is a comparison set such that $ci(CS)$ begins after $t_r(b)$, $j = \max\{E[\tilde{u}(b')] \mid b' \in CS\}$, $k = E[\tilde{u}(CS)]$, and $p_b(x)$ is the probability density function on the utility outcomes for b.

Proof. Let E_{na} be the expected utility of using the naïve decision process and E_{im} be the expected utility of using the improved process. Since the naïve decision process will choose b if its utility is higher than j, and let it expire otherwise, thus expecting utility k, then[1]

$$E_{na} \;=\; A_1k + A_2E[\tilde{u}(b) \mid j < \tilde{u}(b) < k] + A_3E[\tilde{u}(b) \mid \tilde{u}(b) > k] \tag{6.2}$$

Since the improved decision process will choose b if its utility is higher than k, and let it expire otherwise, thus expecting utility k, then

$$E_{im} \;=\; A_1k + A_2k + A_3E[\tilde{u}(b) \mid \tilde{u}(b) > k] \tag{6.3}$$

Subtracting gives

$$\begin{aligned} E_{im} - E_{na} &= A_2k - A_2E[\tilde{u}(b) \mid j < \tilde{u}(b) < k] \\ &= A_2k - A_2 \cdot \frac{\int_j^k xp_b(x)\,dx}{A_2} \\ &= k\int_j^k p_b(x)\,dx - \int_j^k xp_b(x)\,dx \\ &= \int_j^k (k-x)p_b(x)\,dx \quad \square \end{aligned} \tag{6.4}$$

Now consider the general case where there is a bundle b that is about to expire, and a comparison set cover $csc(\mathcal{B})$ on the remaining bundles $\mathcal{B}$.

[1] Here $E[X|C]$ is the expected value of X given that condition C holds.

Theorem 6.3 The expected increase in utility to be achieved by using the improved decision process where $csc(\mathcal{B})$ is any size is

$$\int_0^j (E'_{im} - E'_{na}) p_b(x)\, dx + \int_j^k (E'_{im} - x) p_b(x)\, dx \tag{6.5}$$

where b is the bundle that will expire first, $csc(\mathcal{B})$ is the comparison set cover, $ci(CS)$ begins after $t_r(b)$ for all $CS \in csc(\mathcal{B})$, $j = \max\{E[\tilde{u}(b')] \mid b' \in \mathcal{B}\}$, $k = \max\{E[\tilde{u}(CS)] \mid CS \in csc(\mathcal{B})$, E'_{im} and E'_{na} are the expected utilities of the improved and naïve method if b goes unpurchased, respectively, and $p_b(x)$ is the probability density function on the utility outcomes for b. Moreover, this value is at least $\int_j^k (k - x) p_b(x)\, dx$.

Proof. Let E_{na} be the expected utility of using the naïve decision process and E_{im} be the expected utility of using the improved process. Since the naïve decision process will choose b if its utility is higher than j, and let it expire otherwise, thus expecting utility E'_{na}, then

$$E_{na} \;=\; A_1 E'_{na} + A_2 E[\tilde{u}(b) | j < \tilde{u}(b) < k] + A_3 E[\tilde{u}(b) | \tilde{u}(b) > k] \tag{6.6}$$

Since the improved decision process will choose b if its utility is higher than k, and let it expire otherwise, thus expecting utility E'_{im}, then

$$\begin{aligned} E_{im} &= A_1 E'_{im} + A_2 E'_{im} + A_3 E[\tilde{u}(b) | \tilde{u}(b) > k] \\ &= A_1 (E'_{na} + E'_{im} - E'_{na}) + A_2 (k + E'_{im} - k) + A_3 E[\tilde{u}(b) | \tilde{u}(b) > k] \\ &= A_1 E'_{na} + A_1 (E'_{im} - E'_{na}) + A_2 k + A_2 (E'_{im} - k) + A_3 E[\tilde{u}(b) | \tilde{u}(b) > k] \end{aligned} \tag{6.7}$$

Subtracting gives

$$\begin{aligned} E_{im} - E_{na} &= A_2 k - A_2 E[\tilde{u}(b) | j < \tilde{u}(b) < k] + A_1 (E'_{im} - E'_{na}) + A_2 (E'_{im} - k) \\ &= k \textstyle\int_j^k p_b(x)\, dx - \int_j^k x p_b(x)\, dx + A_1 (E'_{im} - E'_{na}) + A_2 (E'_{im} - k) \\ &= \textstyle\int_j^k (k - x) p_b(x)\, dx + A_1 (E'_{im} - E'_{na}) + A_2 (E'_{im} - k) \end{aligned} \tag{6.8}$$

Since $E'_{im} \geq E'_{na}$ by Theorem 6.1 and $E'_{im} \geq k$ by Lemma 6.2, this value is greater than or equal to $\int_j^k (k-x)p_b(x)\,dx$. Simplifying (6.8) gives,

$$\begin{aligned} E_{im} - E_{na} &= \textstyle\int_j^k kp_b(x)\,dx - \int_j^k xp_b(x)\,dx + \int_0^j E'_{im}p_b(x)\,dx - \\ &\quad \textstyle\int_0^j E'_{na}p_b(x)\,dx + \int_j^k E'_{im}p_b(x)\,dx - \int_j^k kp_b(x)\,dx \\ &= \textstyle\int_0^j (E'_{im} - E'_{na})p_b(x)\,dx + \int_j^k (E'_{im} - x)p_b(x)\,dx \quad \square \end{aligned} \tag{6.9}$$

6.1.2 An Example

This section shows how Theorem 6.2 is used to compute the utility increase yielded by the improved decision method for a simple example. The performance of the theorem is then tested for a slightly different example with a risk averse buyer. Let b_1, b_2 and b_3 be bundles, where b_1 expires during the prequotes for b_2 and b_3 and $CS = \{b_2, b_3\}$ is a comparison set. For simplicity, let the bundle costs be independent and normally distributed, and consider the buyer to be risk neutral. Then the bundle purchase utilities will be normally distributed. Consider the parameters given in Table 6.1.

b_i	μ_i	σ_i
b_1	.5	.06
b_2	.475	.13
b_3	.484	.1

Table 6.1: Means and variances of bundle purchase utilities in the example

At time $t_r(b_1)$, the buyer will know $c(b_1)$ (and therefore $u(b_1, c(b_1))$), μ_2, σ_2, μ_3, and σ_3. A decision must be made at that time between purchasing b_1, which will achieve utility $u(b_1, c(b_1))$, and allowing b_1 to expire, which will achieve the higher of the two utility outcomes for b_2 and b_3.

Recall that the naïve decision process chooses b_1 iff $u(b_1, c(b_1)) > j$, where $j = \max\{E[\tilde{u}(b)] \mid b \in CS\} = .484$, and the improved process chooses b_1 iff $u(b_1, c(b_1)) >$

k, where $k = E[\tilde{u}(CS)]$. Using MC simulation, we find that $k = .545$. Then by Theorem 6.2, the expected increase in utility is

$$\int_{.484}^{.545} (.545 - x) p_{b_1}(x)\, dx \approx .012 \tag{6.10}$$

where $p_{b_1}(x)$ is the normal probability density function with $\mu = .5$ and $\sigma = .06$.

This means that we expect to achieve .012 more utility by using the improved decision process. It is difficult to make any conclusions as to the significance of this increase without knowing the utility function, since it is the result of some combination of more highly preferred bundles and lower costs. But, for the sake of a simple example, assume that all bundles are preferred equally. If this is the case, then lower costs will be completely responsible for the increase in utility. Also, assume that the range of possible outcomes of bundle costs is $[\$100, \$200]$, and therefore $u_z(\$100) = 1$ and $u_z(\$200) = 0$. If the buyer is risk neutral then each .01 of bundle utility represents \$1, and therefore an increase in utility of .012 represents a savings of \$1.20. This is a significant result considering that this is such a small-scale example. Consider a more large-scale example, such as purchasing materials for a major construction project, where the range of values spans one million dollars instead of one hundred. In this case, the increase in utility represents \$12,000 in savings.

While this example assumes that the buyer is risk neutral, this is often not the case. Most buyers will typically show some degree of risk aversion. Consider the utility function

$$u_z(z) = \frac{1 - e^{\frac{z-200}{200}}}{1 - e^{-\frac{1}{2}}} \tag{6.11}$$

Note that this is a risk averse function where 200 is the risk tolerance, $u_z(100) = 1$, and $u_z(200) = 0$. See Keeney and Raiffa [KR76] for material on risk averse utility functions. Again for simplicity, assume that all bundles are preferred equally and therefore bundle purchase utility is dependent entirely on cost.

Testing was done using bundles with parameters as given in Table 6.2. These values were chosen because they give the same means and standard deviations on utility as the bundles described above. For example, using (6.11) for the utility function, the mean utility for purchasing b_1 would be .5 and the standard deviation .06. As before, b_1 expires before the costs of b_2 and b_3 are known, and $CS = \{b_2, b_3\}$ is a comparison set. Since the bundle costs are normally distributed and the utility function is risk averse, the resulting distribution of purchase utility outcomes for each bundle will be negatively skewed. In MC simulations for this example, the naïve decision process achieves an average utility of .540, while the improved process achieves an average of .552. This gives an increase of .012, as predicted by the theorem when the utility of purchasing b was assumed to be normally distributed. While the exact increase can be computed if the utility distribution function for b is known, this indicates that a moderate level of risk aversion does not excessively skew the distribution, and it is therefore reasonable in such cases to use the normal distribution function.

b_i	μ_i (\$)	σ_i (\$)
b_1	156.10	5.88
b_2	158.23	12.56
b_3	157.51	9.71

Table 6.2: Means and variances of bundle costs in the example for a risk averse buyer

6.1.3 Adding New Choices

This section gives an idea of how much the expected utility of a comparison set rises when a new bundle is added to the set. This can be used to obtain an estimate of the expected utility of a comparison set without using Monte Carlo or other methods. Note that this provides a good estimate only when bundle purchase utilities are highly independent and close to normally distributed. In general, MC methods should be used.

Let X_1 and X_2 be independent random variables representing the expected utility of bundle sets $\mathcal{B}_1$ and $\mathcal{B}_2$ (each possibly consisting of a single bundle), and let X_1 and X_2 have parameters (μ_1, σ_1) and (μ_2, σ_2), respectively. Note that, for X_1 and X_2 to be independent, the cost outcomes for bundles in $\mathcal{B}_1$ must be independent of cost outcomes for all bundles in $\mathcal{B}_2$. Also, let X_1 and X_2 be such that $\mu_1 \geq \mu_2$. The goal is to determine the mean μ of the utility of $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$.

Let the *relative mean* $\mu_r = (\mu_2 - \mu_1)/\sigma_1$ be the mean of X_2 in relation to X_1 in terms of the standard deviation of X_1 and the *relative standard deviation* $\sigma_r = \sigma_2/\sigma_1$ be the standard deviation of X_2 in terms of the standard deviation of X_1. The mean μ of the new set of choices $\mathcal{B}$ is calculated by

$$\mu = \mu_1 + \Delta\mu \cdot \sigma_1 \tag{6.12}$$

where $\Delta\mu$ is the increase in the mean of X_1 in terms of the standard deviation of X_1 when the choices from X_2 are added. This can be closely approximated by

$$\Delta\mu = a_1\mu_r^3 + a_2\mu_r^2 + a_3\mu_r + a_4 \tag{6.13}$$

$a_1 =$	$0.00122453\sigma_r^3 + 0.00198985\sigma_r^2 - 0.02717656\sigma_r + 0.04144293$
$a_2 =$	$0.00967392\sigma_r^3 - 0.02017891\sigma_r^2 - 0.07310484\sigma_r + 0.24396784$
$a_3 =$	$0.00166528\sigma_r^3 - 0.00878871\sigma_r^2 + 0.00856184\sigma_r + 0.50629631$
$a_4 =$	$-0.02798339\sigma_r^3 + 0.15946105\sigma_r^2 + 0.04476181\sigma_r + 0.38835235$

Table 6.3: Values for a_i constants in (6.13)

where the a_i constants are given in Table 6.3. These constants were generated by using simulation to determine $\Delta\mu$ for several (μ_r, σ_r) pairs. Linear regression was then used to estimate the coefficients.

For example, consider again b_2 and b_3 with parameters as in Table 6.1. This function can be used to approximate the expected utility of $CS = \{b_2, b_3\}$ by calculating the estimated increase in mean when the two bundles are added together. Since $\mu_3 > \mu_2$, $\mu_r = (\mu_2 - \mu_3)/\sigma_3 = (.475 - .484)/.1 = -.09$, and $\sigma_r = \sigma_2/\sigma_3 = .13/.1 = 1.3$. From (6.13), $\Delta\mu = .610$. Using (6.12), $\mu = .484 + .610 \cdot .1 = .545$, which is equal (to 3 decimal places) to $E[\tilde{u}(CS)]$ as found by MC simulation in section 6.1.2.

Testing shows that this approximation is accurate with a standard error of less than .001 when $-2 \leq \mu_r \leq 0$ and $0.4 \leq \sigma_r \leq 2$. These ranges are reasonable to consider since, because we are typically comparing similar interchangeable items, we expect that commonly μ_r stays close to 0 (and by definition is never positive), and σ_r stays close to 1. Therefore, the new utility calculated by (6.12) is accurate with a standard error of $.001\sigma_1$.

Table 6.4 provides an excerpt of the accuracy tests of the approximation function. This is simply to give the reader a rough idea of how much improvement is gained by adding a new set of choices to an existing set, by giving the results of a few example values. Recall that an improvement of 0.12, for example, would mean that μ is expected to rise by $0.12 * \sigma$ if the choices are added together, where μ and σ are those of the set of choices with the higher μ.

For example, consider two bundles b_1 and b_2, each with means and standard deviations (μ_1, σ_1) and (μ_2, σ_2), respectively, where $\mu_1 = .6$, $\sigma_1 = .1$, $\mu_2 = .52$, and $\sigma_2 = .15$, giving a relative mean $\mu_r = \frac{.52 - .6}{.1} = -0.8$ and a relative standard deviation $\sigma_r = .15/.1 = 1.5$. From Table 6.4, we see that the increase in mean is about .39. So the expected highest utility of the two choices put together is $\mu_1 + .39\sigma_1 = .6 + .039 = .639$.

μ_r	σ_r	MC μ	Approx μ	Difference
-1.2	0.5	0.0807	0.0807	0.0000
-1.2	1.0	0.1550	0.1560	0.0010
-1.2	1.5	0.2719	0.2729	0.0010
-0.8	0.5	0.1545	0.1559	0.0014
-0.8	1.0	0.2537	0.2521	0.0016
-0.8	1.5	0.3908	0.3889	0.0019
-0.4	0.5	0.2738	0.2744	0.0006
-0.4	1.0	0.3864	0.3860	0.0004
-0.4	1.5	0.5377	0.5367	0.0010

Table 6.4: Examples of utility increase when adding a new set of choices, where μ_r is the relative mean, σ_r is the relative standard deviation, MC μ is the mean found using Monte Carlo, Approx μ is the mean found using (6.12) and (6.13), and Difference is the difference between MC μ and Approx μ.

6.2 Partial Bundle Purchases

This section provides an analysis of the performance of the three methods developed for making decisions in the setting where partial bundle purchases are allowed. Note that all proofs are based on the assumption that a sufficient degree of accuracy can be achieved in results obtained through simulation. Also, while the implementation described in this thesis strictly generates degree-three polynomials to represent utility projection functions, results here are based on the added assumption that sufficiently accurate functions are computed.

6.2.1 The Restricted q-horizon Method

This section shows that, in the setting where partial bundle purchases are permitted, the buyer's expected utility when using the restricted q-horizon method is always at least as high as when using the greedy method. For a node n in a QR-tree, consider the notation given in Table 6.5.

While, in reality, the buyer actually traverses the purchase procedure tree during the

Notation	Meaning
$E_{rq}(n)$	the true (possibly unknown) expected utility if the restricted q-horizon method is used from n until the end
$E_{gr}(n)$	the true (possibly unknown) expected utility if the greedy method is used from n until the end
$\theta_{rq}(n)$	the expected utility at n estimated by the restricted q-horizon method
$\theta'_{rq}(n)$	the expected utility at n estimated by the restricted q-horizon method if all q-horizons in n's q-region are eliminated (i.e. for any decision node d in the q-region, $qsh(d')$ is subtracted from $qh(d)$ for every descendent d')
$\theta_{gr-rq}(n)$	the estimated expected utility at n if the greedy method is used throughout n's q-region, and the restricted q-horizon method is used afterward

Table 6.5: Notation used in proofs where n is a node in a QR-tree

process of making decisions and purchases towards the goal of procuring a bundle, for simplicity in this section the QR-tree is used to guide the purchase procedure. This is sufficient since, because all decision nodes are the same in each tree (decision times are the same, and the information known at decision time is the same for each) and each path represents the same purchases (albeit possibly in a different order), the expected utilities for choices at decisions are the same in each tree. To avoid the confusion of changing the context from purchase procedure trees to QR-trees and back again, only the QR-tree is used in this discussion.

Definition 6.1 (fringe) Let R be a q-region in a QR-tree T. The fringe F of R is a set of nodes from T such that a node n is in F iff n's parent is in R but n is not.

Lemma 6.3 For the root node n in the QR-tree, $\theta_{gr-rq}(n) \geq E_{gr}(n)$.

Proof. If n is a purchase node, then $\theta_{gr-rq}(n) = \theta_{rq}(n)$. Since $\theta_{rq}(n)$ is always greater than or equal to $E_{gr}(n)$ (since it is always at least as high the expected utility of using the greedy method through any region), then, in the case where n is a purchase node, $\theta_{gr-rq}(n) \geq E_{gr}(n)$. Let n then be a decision node with q-region R. For any node f in the fringe F of R, by definition $\theta_{rq}(f) \geq E_{gr}(f)$ for any above value in A_f. Since $\theta_{gr-rq}(n)$ is the utility expected by the greedy method given that the expected utility of each $f \in F$ is $\theta_{rq}(f)$ for above values in A_f and $E_{gr}(n)$ is the utility expected by the greedy method given that the expected utility of each $f \in F$ is $E_{gr}(f)$ for above values in A_f, then $\theta_{gr-rq}(n) \geq E_{gr}(n)$. □

Theorem 6.4 For the root node n in the QR-tree, the expected utility $E_{rq}(n)$ when using the restricted q-horizon method is greater than or equal to the expected utility $E_{gr}(n)$ when using the greedy method.

Proof. Since by Lemma 6.3 $\theta_{gr-rq}(n) \geq E_{gr}(n)$, showing that $E_{rq}(n) \geq \theta_{gr-rq}(n)$ is sufficient. This is done by induction on the height of the QR-tree.

Base Case. Let n be a leaf. Using any method will simply result in the purchase of the item represented by n, so $E_{rq}(n) = \theta_{gr-rq}(n)$.

Induction. Assume that, for any child n' of n, $E_{rq}(n') \geq \theta_{gr-rq}(n')$ for any above value in $A_{n'}$. If n is a purchase node, then using any method will result in the purchase of the item represented by n, and execution will move to the child n'. By induction, $E_{rq}(n') \geq \theta_{gr-rq}(n')$. Let n be a decision node. If the each method suggests the same choice then, by induction, $E_{rq}(n') \geq \theta_{gr-rq}(n')$. Assume then that the restricted q-horizon method suggests proceeding to n' while the greedy method suggests proceeding to n''. Then

$$\theta_{rq}(n') \geq \theta_{rq}(n'') \tag{6.14}$$

Since $\theta_{rq}(n'') = \max\{\theta'_{rq}(n''), \theta_{gr-rq}(n'')\}$,

$$\theta_{rq}(n'') \geq \theta_{gr-rq}(n'') \tag{6.15}$$

and therefore

$$\theta_{rq}(n') \geq \theta_{gr-rq}(n'') \tag{6.16}$$

Since $\theta'_{rq}(n')$ is the estimated expected utility of n' if the q-horizons are restricted throughout the q-region of n, then the true expected value of n' where all info is known at each decision must be higher. Thus

$$E_{rq}(n') \geq \theta'_{rq}(n') \tag{6.17}$$

Using the induction hypothesis with (6.17)

$$\begin{aligned} E_{rq}(n') &\geq \theta_{gr-rq}(n') \\ &\geq \max\{\theta'_{rq}(n'), \theta_{gr-rq}(n')\} \\ &\geq \theta_{rq}(n') \end{aligned} \tag{6.18}$$

Then by (6.16) and (6.18)

$$E_{rq}(n') \geq \theta_{gr-rq}(n'') \tag{6.19}$$

So, whenever the restricted q-horizon method suggests n' and the greedy method suggests n'',

$$E_{rq}(n) = E_{rq}(n') \geq \theta_{gr-rq}(n'') \geq E_{gr}(n'') = E_{gr}(n) \tag{6.20}$$

and therefore $E_{rq}(n) \geq E_{gr}(n)$. □

6.2.2 The Classification Tree Method

The goal of this section is to show that the expected utility estimate provided by the classification tree method for some choice is an underestimate of the true utility that one would expect if this method is used throughout the purchase procedure. It is clear from the nature of the solution method that, if choices are made at each decision node in the purchase procedure tree using classification trees in the initial QR-tree, then the estimate computed for a choice is equal to its expected utility. This is the case since this decision-making process is exactly what is simulated by the classification tree method. However, this is not likely to be the way decisions are ultimately made. It does not make sense to use previously constructed classification trees if new information has been obtained in the interim. In fact, any time any new information is obtained (e.g. prices become known, new prequotes are received, other quotes expire), purchase procedure trees and their corresponding QR-trees should be reconstructed. If this is done, however, the classification tree method is not guaranteed to provide an underestimate. It is possible that when attempting to solve some decision point d in the purchase procedure tree, even though more information may be available at d's decision time than when the classification trees were initially determined, the decision-maker may make a "mistake" that was not made during the initial simulation. Here, a mistake refers to the act of opting for the choice with the lower true expected utility. A possibility such as this could, in theory, result in the buyer's true expected utility actually being less than that originally estimated.

That said, it is still interesting to be able to declare that expected utility is at least some value θ_{ct}, *given the condition that the buyer only uses these classification trees to solve future decisions.* It should also be stated that previously constructed classification trees need only be used at a decision node d if there exists a descendent decision node d' such that $qss(d')$ in the QR-tree includes an item for which the price is unknown at time $t(d)$. At any other decision time, new trees can be constructed and expected utilities recomputed, and the underestimation property still holds. This method is formally described in Algorithm 6.1, and the underestimation property is subsequently proven.

Algorithm 6.1 (ctMethod(d)) Let T_{pp} and T_{qr} be the purchase procedure tree and QR-tree rooted at decision node d, respectively. For each decision node d' in T_{pp}, let $qr(d')$ be the corresponding decision node in T_{qr}. Also, let D_{CT} be the subset of the decision nodes in T_{pp} where $d_{ct} \in D_{CT}$ iff there exists a descendent decision d'' of d_{ct}

such that there is an item $i \in qss(qr(d''))$ for which the price outcome is unknown at time $t(d_{ct})$. Finally, let $\theta_{ct}(\ell(d))$ and $\theta_{ct}(r(d))$ be the expected utility estimated for the left and right choice at d, respectively. The following dictates how to traverse T_{pp}:

1. If $\theta_{ct}(\ell(d)) > \theta_{ct}(r(d))$, then let $curNode = \ell(d)$. Else, $curNode = r(d)$. Proceed to $curNode$.

2. Let $n = curNode$. If n is null, then end. Else if n is a purchase node, buy n, let $curNode$ be the child of n, and repeat 2. Else n is a purchase node. If $n \in D_{CT}$, let $curNode$ be the child of n with the higher expected utility estimated using T_{qr} with the current price outcomes (see section 5.6.2), and repeat 2. Else, do *ctMethod(n)*. □

Theorem 6.5 Let d be the root decision node in a purchase procedure tree T. The expected utility $\theta_{ct}(d) = \max\{\theta_{ct}(\ell(d)), \theta_{ct}(r(d))\}$ of a choice n at d estimated by the classification tree method is less than or equal to the utility one is expected to achieve if *ctMethod(d)* as given in algorithm 6.1 is used in the traversal of T.

Proof. Consider the abstract representation of the purchase procedure tree rooted at d depicted in Figure 6.2. The tree is divided into two main areas: the area where decision nodes are in D_{CT} and the area where they are not (referred to as $\overline{D}_{CT}$). A portion of $\overline{D}_{CT}$ referred to as the *fringe* of $\overline{D}_{CT}$ is also shown. A decision node d_f is in the fringe of D_{CT} iff d_f is the first decision node on the path from the root to be in $\overline{D}_{CT}$. Since there are no q-subsets for decision nodes below $qr(d_f)$ in the QR-tree T_{qr} containing items for which prices are unknown at time $t(d_f)$ and the therefore expected utilities can be computed exactly, the buyer will know exactly which choice at d_f will maximize expected utility. The buyer will therefore make decisions in the fringe at least as well as was done during simulations at $t(d)$ using T_{qr}. Thus the buyer's expected utility at d_f, for any above value $a \in A_{d_f}$, is at least as high as that estimated at $t(d)$. And since decisions occurring before the fringe is reached are done in exactly the same manner as they were in T_{qr} simulations, then the buyer's true expected utility if *ctMethod(d)* is used is greater than or equal to $\theta_{ct}(d)$. □

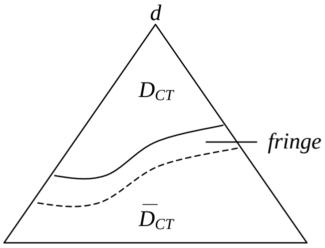


Figure 6.2: Abstract representation of a QR-tree for the proof of Theorem 6.5

6.2.3 The q-subset Discretization Method

Since the q-subset discretization method uses approximation to discretize the outcomes for q-subset items, it is difficult to prove anything theoretically about its performance. Before any solid conclusions can be made, therefore, the technique must endure thorough testing. This is done in section 6.3. However, to give the reader an understanding of how accurately this method estimates expected utility, this section presents an analysis of its performance on a simple case where the true expected utility can be computed. Consider the QR-tree given in Figure 6.3.

The purpose of this analysis is to compare how accurately the q-subset discretization method estimates the expected utility of the right subtree of d_1, compared to simply restricting the q-horizon of d_2 and ignoring the fact that B is included (thus eliminating the q-subhorizon of d_3. This can be done here since the problem is small enough that the true expected utility can be computed quite accurately. The test will attempt to answer the question of whether considering B to be part of $qh(d_2)$ is worth the tradeoff associated with accuracy loss due to the PT-approximation of B's outcomes.

For all tests, A, B and C are considered to be items in singleton bundles, each

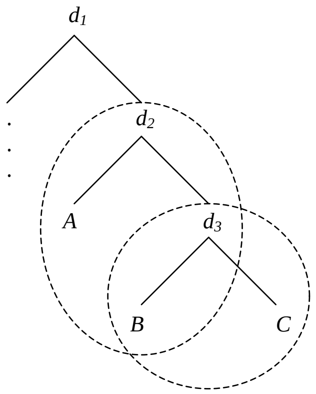


Figure 6.3: A simple QR-tree used to demonstrate the performance of the q-subset discretization method.

bundle with equal utility (so only the cost has any effect on the outcome of utility). Also, for simplicity, a risk-neutral utility function is used. For the first test, the utility outcome for each of B and C have equal mean (say 0, but the actual number chosen is irrelevant), and as well as equal standard deviation (0.05). A is also given a standard deviation of 0.05 and its mean is incremented by 0.01 from -0.2 to 0.4. For each of these values, the restricted q-horizon estimate, the q-subset discretization estimate and the true expected utility (with a standard error ≤ 0.001) were computed. Figure 6.4 shows the results of these tests.

This shows that, while the q-subset discretization estimate stays within .001 of the true value, the restricted method gives a higher error particularly when the mean of A is between 0 and 0.1. Next, the performance is tested when each of the μ_B, μ_c, σ_B, and σ_C values are increased. These effects are demonstrated in Figure 6.5

When the mean of B is increased, the error in the restricted estimate rises dramatically, since the outcome of B is much more influential and it is thus much worse to ignore the fact that this outcome will be known before the decision at d_2 is made. Increasing the mean of C on the other hand is not very interesting, because it makes the outcome of B less important. Increasing the standard deviation of C does not

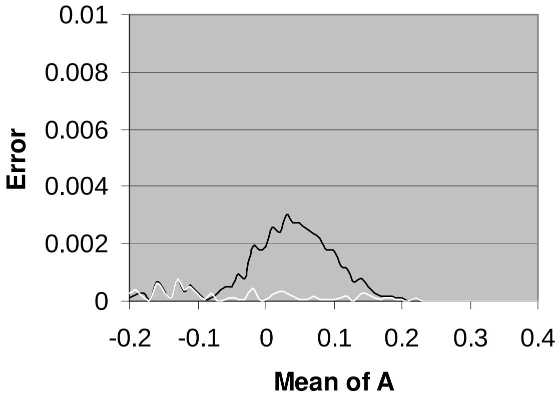
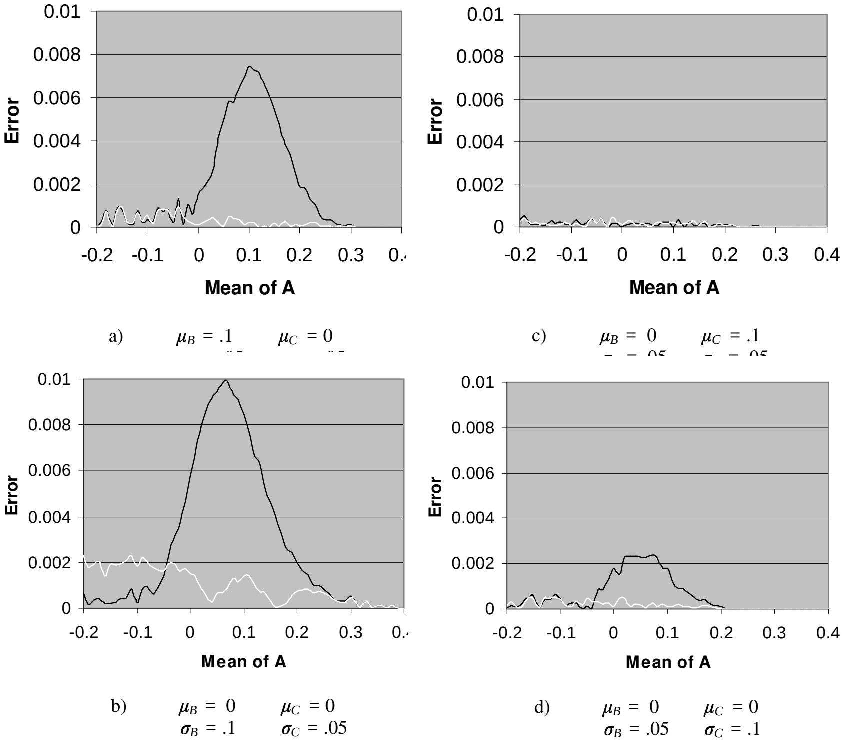


Figure 6.4: Estimation error for each method where $\sigma_A = \sigma_B = \sigma_C = 0.5$ and $\mu_B = \mu_c = 0$. The black line indicates the restricted q-horizon error while the white line indicates the q-subset discretization error.

seem to have much effect. However, increasing the standard deviation of B provides a very interesting case. The q-subset discretization estimate is poor, especially when $\mu_A < 0$. This is the area where A will almost never be purchased, so the expected utility of the right subtree is equal to the expected utility of the subtree starting at d_3. Naturally, the q-subset discretization estimate will not be as accurate here. However, as A rises, the restricted estimate becomes as much as 5 times worse. This is due to the fact that, since the variability of the outcome of B is high, it is important to consider that this outcome will be known before the decision at d_2 is made.

For completion, Figure 6.6 demonstrates the effect of increase combinations. Note that no results are given for increases in both μ_B and μ_C, since only the levels of these means in relation to each other are relevant. Note that, while the only case above where the PT estimate was poor is when σ_B was increased and μ_A was low, the error seems to disappear when σ_C is increased to the same level (as shown in Figure 6.6(e)).



a) μ_B = .1 μ_C = 0

c) μ_B = 0 μ_C = .1

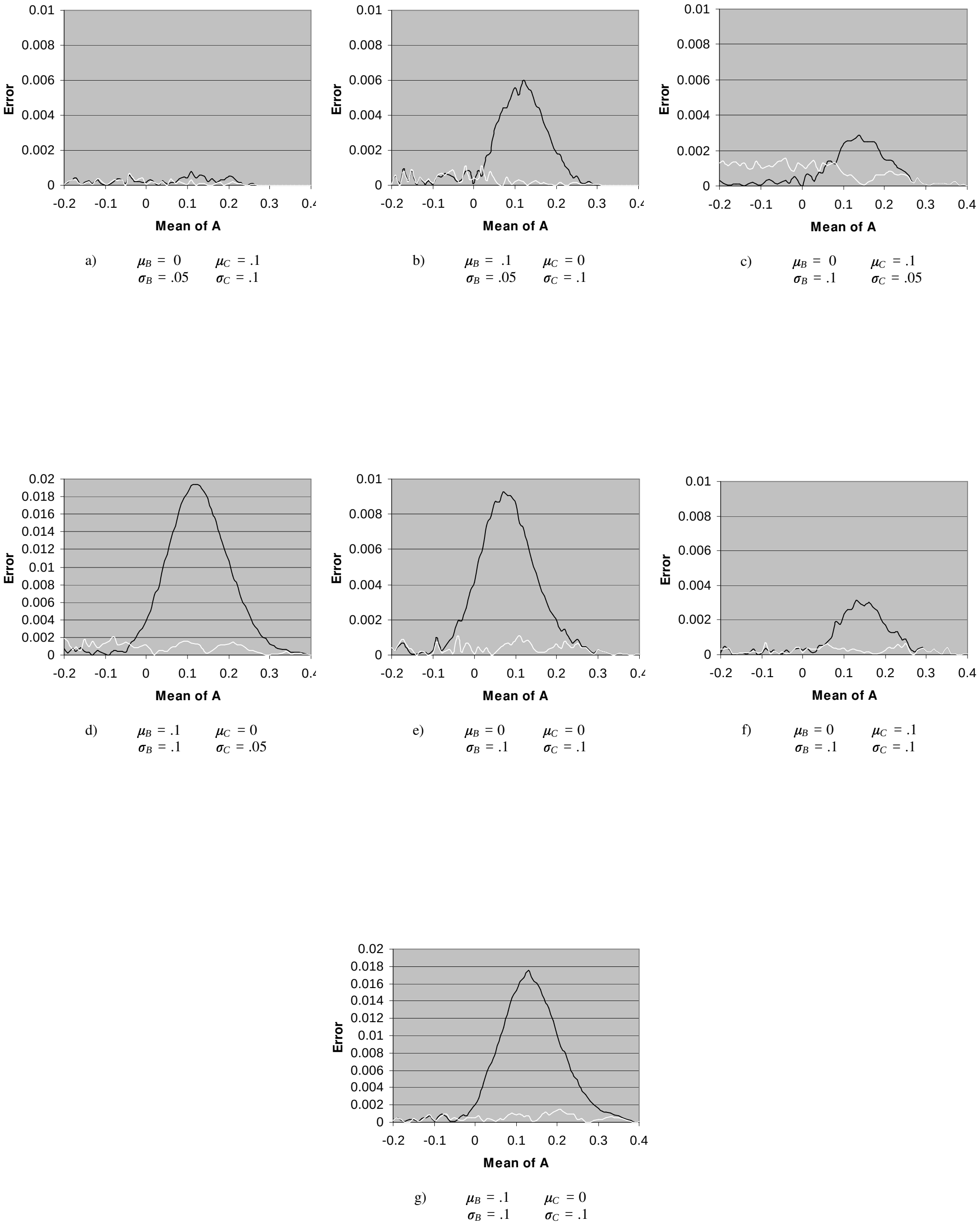
b) μ_B = 0 μ_C = 0
σ_B = .1 σ_C = .05

d) μ_B = 0 μ_C = 0
σ_B = .05 σ_C = .1

Figure 6.5: The effects of individually increasing (a) μ_B to .1 (b) μ_C to .1 (c) σ_B to .1 (d) σ_C to .1.

6.3 Testing of QR-Tree Solution Methods

This section discusses the implementation of the QR-tree solution techniques described in this thesis, as well as results of tests performed using these methods on an example QR-tree. The purpose of this section is merely to provide a demonstration of how well these methods perform in a semi-random simulated environment when compared to the greedy decision procedure. No serious claims regarding the statistical significance of the performance results are made here. Such a test would be pointless since it is possible that in some rare situations a buyer could actually do worse by using one of these techniques. This is due to the fact that approximation techniques are used. However, the results of the previous section (i.e. the proofs that the restricted q-horizon method is always better than the greedy method and that



a) μ_B = 0 μ_C = .1
σ_B = .05 σ_C = .1

b) μ_B = .1 μ_C = 0
σ_B = .05 σ_C = .1

c) μ_B = 0 μ_C = .1
σ_B = .1 σ_C = .05

d) μ_B = .1 μ_C = 0
σ_B = .1 σ_C = .05

e) μ_B = 0 μ_C = 0
σ_B = .1 σ_C = .1

f) μ_B = 0 μ_C = .1
σ_B = .1 σ_C = .1

g) μ_B = .1 μ_C = 0
σ_B = .1 σ_C = .1

Figure 6.6: The effects of increasing (a) μ_C,σ_C (b) σ_B,σ_C (c) σ_B,μ_C (d) σ_B,μ_C,σ_C (e) μ_B,σ_C (f) μ_B,σ_B (g) μ_B,σ_B,σ_C .

the classification tree method always gives a lower bound on the expected utility) combined with the evidence presented in this section make a solid case in favour of the techniques presented in this thesis.

Each of the three methods were implemented and tested, as well as the greedy method for the sake of comparison. Two results are examined for each method: the accuracy of the estimated expected utility produced and the overall utility achieved. Tests were carried out over 100 random instances of the means and standard deviations of items in the QR-tree. For each instance, the expected utility was estimated using each technique. The purchase procedure was then run 10000 times to determine an accurate estimate of the true utility expected for each instance.

6.3.1 Implementation

Implementation was done using Java 2 Standard Edition (J2SE) 1.4.0. All tests were run on a Dell LATITUDE C840 PC with a Pentium 4 CPU on the Microsoft Windows XP platform. Each of the restricted q-horizon, q-subset discretization and classification tree methods were implemented. A few details on each implementation are presented here.

Input

The file qrtree.java can use any of the three methods to estimate the expected utility of each choice at a decision point. The program requires:

- the choice of method to use
- the standard error to use in simulations
- the QR-tree
- the utility function for money and the two-attribute utility function for bundle purchases

The estimation method to use is set in the first line of `main` using `method_type`. The values used are: 0 for the q-subset discretization method (referred to as PT in the program), 1 for the classification tree method (referred to as CT) and 2 for the restricted q-horizon method (referred to as RQ).

```
int method_type = ‘‘PT’’; //PT, CT, RQ
```

In the second line in `main`, the standard error is set. This value is used for every part of the algorithm in which simulation is done, determining the number of simulations that need to be performed. Each time a series of simulations is needed to estimate the average result of some function, 1000 runs are carried out in order to estimate the variance s^2. Since the standard error of the mean of n simulations is $\sqrt{\frac{s^2}{n}}$, the number of simulations needed to determine the mean with standard error se is $\frac{s^2}{se^2}$.

```
double sterr = 0.0005;
```

The QR-tree is entered in the method `createTree`. An `Item` object is first created for each item. The values given to construct the object are the item name, the mean price, standard deviation, and quote and rescind times. A single dummy item is created for decisions.

```
//Item(ItemName,meanPrice,stDev,q,r)
Item A = new Item("A",345,0,0,1);
Item B = new Item("B",350,15,2,5);
Item C = new Item("C",340,11,6,8);
Item D = new Item("D",315,10,3,7);
Item E = new Item("E",370,13,4,8);
Item F = new Item("F",360,10,5,9);
Item G = new Item("G",350,15,4,9);
Item d = new Item("d",0,0,0,0);
```

`Node` objects are then created to build the tree. Purchase nodes require the item, the child node, and the bundle utility (if it is the final purchase node on a path, 0 otherwise). Decision nodes require the item (always d), the left child node, the right child node, and the decision time.

```
  //Node = new Node(item, child , bundleUtil);
  //Node = new Node(decItem, leftchild, rightchild,decTime);
  Node pB1 = new Node(B,null,0.45);
  Node pC  = new Node(C,null,0.4);
  Node d2  = new Node(d,pB1,pC,5);
  Node pA  = new Node(A,d2,0);
  Node pE1 = new Node(E,null,0.6);
  Node pF  = new Node(F,null,0.55);
  Node d4  = new Node(d,pE1,pF,8);
  Node pE2 = new Node(E,null,0.3);
  Node pG  = new Node(G,null,0.25);
  Node d5  = new Node(d,pE2,pG,8);
  Node pB2 = new Node(B,d4,0);
  Node pD  = new Node(D,d5,0);
  Node d3  = new Node(d,pB2,pD,5);
  Node d1  = new Node(d,pA,d3,1);
```

Finally, the utility functions must be entered. This is done with the `uz` and `u` methods.

```
public static double uz (double z){
  //range 600-800
  //risk averse; tolerance=500
  return (1-Math.exp((z-800)/500))/(1-Math.exp(-0.4));
}

public static double u (double ub, double z){
  double uz=uz(z);
  double kb=0.4;
  double kz=0.6;
  double kbz=0;
  return kb*ub + kz*uz + kbz*ub*uz;
}
```

When executed, the system begins by preprocessing the QR-tree. This involves the construction of the q-sets, q-subsets and q-subset-complements for each decision node, as well as the set A' of above values (the 19 chosen values as discussed in section 5.6.1) for each node. Each of the left and right subtrees of the root (which must be a decision node in this implementation) are then printed before the specified estimation method is employed.

The Restricted q-horizon Method

The restricted q-horizon method starts at the leaves and moves to the root. Each time a purchase node n is encountered that does not reside in a q-horizon, the expected utility of n is determined for each above value $a \in A'_n$ by simulating the outcome for n's item, as described in chapter 5. After an appropriate number of simulations, the average result is taken to be the expected utility of n given a. Each a and the corresponding expected utility are stored in a 2×19 array. Once the expected utility is determined for all 19 values in A'_n, the regression method is called to fit a degree-three polynomial function for the utility projection function of n. This function is stored in a 1×4 array where the i^{th} column contains the coefficient for x^i.

Any purchase node that does reside in a q-horizon is skipped.

When a decision node d is encountered, for each $a \in A'_n$ the outcomes for restricted q-set items (i.e. the items represented by nodes in the restricted q-horizon) are simulated and a temporary expected utility u' of d given a is determined. The greedy method is then simulated throughout the q-region of d, and the maximum of this result and u' is taken to be the expected utility of d given a. After all 19 values are determined, the regression method is called and the utility projection function is determined. Since there is only a single above value for each child of the root of the tree, the expected utility of each child given this above value is taken to be the expected utility of each choice. These values are then output to the user.

The q-subset Discretization Method

The q-subset discretization method computes utility projection functions for purchase nodes in exactly the same manner as the restricted q-horizon method, but decision nodes are handled quite differently. Let $K(qss(d))$ be the set of joint PT-outcomes for the items in the q-subset of a decision node d. For each $k \in K(qss(d))$, the expected utility of d for each $a \in A'_n$ is determined as described in section 5.6.2, and an utility projection function is constructed. These utility projection functions are used to make up the q-subset-mapping. The q-subset-mapping is stored in a $|K(qss(d))| \times 4$ array, where the i^{th} row contains the utility projection function that corresponds to the outcome k as follows: Let i_3 be the number i in base 3 and let `qss_array` be the array of items in $qss(d)$. If x is the j^{th} digit in i_3, then the outcome for the item `qss_array[j]` in k, where μ and σ are the mean and standard deviation of the price of `qss_array[j]` respectively, is

$$\begin{array}{ll} \mu - 1.645\sigma & \text{if } x = 0 \\ \mu & \text{if } x = 1 \\ \mu + 1.645\sigma & \text{if } x = 2 \end{array}$$

An array of length $|K(qss(d))|$ is also kept to hold the probability of each k occurring. The q-subset-mapping and the corresponding probabilities are then used to compute q-subset-mappings for ancestor decision nodes. Once the expected utility of each child of the root given the single is above value is computed, the values are output to the user.

The Classification Tree Method

The classification tree method traverses the QR-tree entirely in a top-down manner. Items are bought whenever purchase nodes are encountered, and the utility projection functions and q-subset-mappings constructed by the q-subset discretization method (which must be run first) are used to make decisions, as described in section 5.6.2. The classification tree for each decision node is represented by the same array used to store the q-subset-mappings. When determining which path in the classification tree to follow for a given q-subset outcome k, the appropriate PT-outcome is determined for each item in `qss_array` to form a string of digits. This string is then considered to be a base 3 number which is converted to base 10, yielding the index of the appropriate utility projection function (the utility projection function at the leaf of the corresponding path in the classification tree) in the q-subset-mapping.

Each of the left and right subtrees at the root is simulated in this manner an appropriate number of times, and the average of each is returned as their respective expected utilities.

Output

The program displays the subtree of the QR-tree for each choice at the root, as well as the expected utility estimated for each choice. For the example input given above, the program determines that the first decision to make is that of whether or not to buy item A, and gives information necessary for the decision as shown in Figure 6.7.

```
cd c:/MyFiles/JavaFiles/cs1073/qrtree/
c:/MyFiles/JavaFiles/java/jdk/bin/javaw.exe -classpath c:/MyFiles
/JavaFiles/cs1073/qrtree/ -Xdebug -Xrunjdwp:transport=dt_shmem,
address=javadebug,server=y,suspend=n qrtree


A subtree:
 - A(1) - d(5) - B(5)
               - C(8)

~A subtree:
 - d(5) - B(5) - d(8) - E(8)
                      - F(9)
        - D(7) - d(8) - E(8)
                      - G(9)

Expected utility of buying A:     0.5465410308495285
Expected utility of buying d:     0.5615281672860315

Process qrtree finished
```

Figure 6.7: Sample output for qrtree.java for given input

For each purchase node, the number in parentheses indicates the rescind time, while for each decision node (represented by "d") the decision time is indicated. This shows that, using the q-subset discretization method, the expected utility of buying A is estimated to be about .5465, while the expected utility of not buying A is about .5615.

6.3.2 Test Bed

Testing was done on the bundle set $\mathcal{B} = \{ACE, ACF, ADG, ADH, BCK, BJL, X\}$. The quote and rescind times for each item are given in Table 6.6. All prequote times occur at or before time 0. For simplicity, these quote periods are chosen so that the purchase procedure tree and the QR-tree are identical. This tree is given in Figure 6.8.

Item	t_q	t_r
A	1	5
B	2	6
C	3	6
D	6	8
E	4	8
F	7	9
G	7	9
H	8	9
J	4	7
K	5	7
L	7	9
X	0	1

Table 6.6: Quote and rescind times for items used in testing

While the bundles and quote periods for items are kept static, the means, standard deviations, and actual outcomes are chosen at random for each test. The task for each instance is to accurately estimate the expected utility of the right subtree at the decision time of d_1 (1), and compare the estimate to the certain utility of proceeding to the left subtree (simply purchasing X). The right subtree contains many

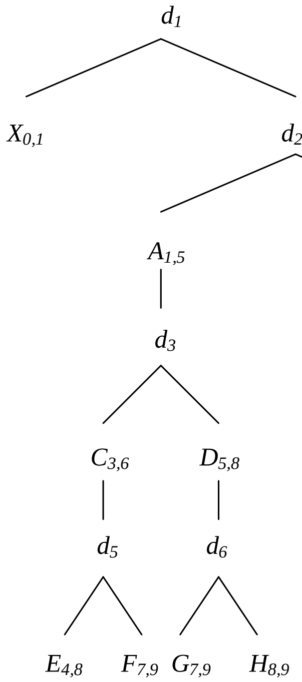


Figure 6.8: QR-tree for experiments. The subscripts for each purchase node denote the t_q and t_r times, respectively. The q-sets for each decision node are $d_1 : X$, $d_2 : ABCEJ$, $d_3 : CDE$, $d_4 : CJK$, $d_5 : EF$, $d_6 : GH$.

q-subhorizons, since if there were no q-subhorizons all three methods would produce the same result. In particular, the subtree includes a decision node with a singleton q-subhorizon (d_5), a decision node with a size 2 q-subhorizon that does not intersect another q-subhorizon (d_4), a decision node with a size 2 q-subhorizon that does intersect another q-subhorizon (d_3) (i.e. the q-horizon for d_2 includes two consecutive subsequent decisions), as well as two decision nodes with empty q-subhorizons (d_2 and d_6). In addition, the outcomes for all purchase nodes that reside in a q-subhorizon will be known at d_2. Therefore, given that d_2 is chosen, all three methods will give the same result. This will help when comparing performances.

One hundred random instances were chosen for testing. For each instance, the means were chosen from a uniform distribution with range [0.9, 1.1] and standard deviations from a uniform distribution with range $[0, \frac{1}{3}]$ for the costs of items A-L. All bundle utilities were considered to be equal and therefore could be ignored, and the risk neutral utility function

$$u_z(z) = 1 - \frac{z-2}{2} \tag{6.21}$$

for money was used. Thus, the two-attribute utility function was simply $u(b, z) = 0(u_b(b)) + 1(u_z(z)) = u_z(z)$. For each instance, the expected utility of the right subtree was calculated using each of the four methods. Since the primary concern is to test the performance of the q-subset discretization method versus the restricted q-horizon method (the greedy method is expected to be much worse than all others and the classification tree method is expected to be quite close to the q-subset discretization method), the mean of item X (the left child of d_1) was taken as the cost that would make the utility of buying X equal to the average of these two estimates for the right subtree. That is, for the q-subset discretization estimate θ_{PT} and restricted q-horizon estimate θ_{rq}, the mean μ_X of the cost of X was chosen so that $u(X, \mu_X) = (\theta_{PT} + \theta_{rq})/2$. This was done for the following reason: If the expected utility of the left subtree is either less than or greater than both expected utilities estimated for the right subtree, then a decision-maker using either method will make the same choice, making for an irrelevant test case. Setting the left mean utility to be exactly in between the two right estimates with a relatively small standard deviation ensures that most test runs will be relevant. The standard deviation σ_X of the cost of X was chosen so that $u(X, \mu_X + \sigma_X) = \max\{\theta_{PT}, \theta_{rq}\}$ (and therefore $u(X, \mu_X - \sigma_X) = \min\{\theta_{PT}, \theta_{rq}\}$). This means that the restricted q-horizon method and the q-subset discretization method will suggest a different choice at d_1 (and therefore produce a different result) 68% of the time. Monte Carlo simulations in calculating the expected utilities made use of antithetic variate sampling [HM56], and calculations used a standard error threshold of .001. The values used for each instance are given in Table A.1 in Appendix A.

6.3.3 Accuracy of Each Method

For each instance, the expected utility at time 1 of the subtree rooted at d_2 was estimated using each method. 10000 simulations of the purchase procedure beginning at d_2 were then executed out to closely determine the true average utility (with standard error ranging from .001 to .0015). Table A.2 in Appendix A gives the estimated expected utility for each method as well as the average utility achieved for each instance. Table 6.7 summarizes the results by giving the average utility estimated and achieved for each method over all 100 cases. Table 6.8 gives the standard error, which is calculated as $\sqrt{sse/99}$ where sse is the sum of squared errors over the 100 cases.

All three methods work quite well when compared to greedy estimation, which is simply just the highest expected utility over all bundles that can be purchased in the right subtree. The q-subset discretization method performed the best, producing only about 4/5 the error of the classification tree method and about 1/3 that of the restricted method.

Method	Estimate
Greedy	0.5464
Restricted q-horizon	0.6372
Classification Tree	0.6429
q-subset Discretization	0.6445
Utility Achieved	0.6468

Table 6.7: Average expected utility estimates for right subtree compared with the true utility achieved (average over 10000 runs per instance)

Method	Standard Error
Greedy	0.1046
Restricted q-horizon	0.0120
Classification Tree	0.0050
q-subset Discretization	0.0040

Table 6.8: Standard error for each estimation method over the 100 test cases

6.3.4 Utility Achieved by Each Method

10000 simulations of the entire purchase procedure were performed for each instance to determine the actual utility achieved by using each method through the entire purchase procedure starting at d_1. The results for each test are given in Table A.3 in Appendix A. Table 6.9 summarizes the results by giving the average utility achieved by each method over all 100 cases.

Method	Utility
Greedy	0.6408
Restricted q-horizon	0.6427
Classification Tree	0.6464
q-subset Discretization	0.6467

Table 6.9: Average utility achieved using each method with μ_X and σ_X such that $u(X, \mu_X) = (\theta_{PT} + \theta_{rq})/2$ and $u(X, \mu_X + \sigma_X) = \max\{\theta_{PT}, \theta_{rq}\}$.

To complete the analysis, a test was carried out to determine how much better the q-subset discretization is over the greedy method. To do this, the mean and standard deviation for the cost of X were set as above, except in relation to the *greedy* estimate, instead of the restricted q-horizon method. The results are given in Table 6.10.

Method	Utility
Greedy	0.6149
q-subset Discretization	0.6506

Table 6.10: Average utility achieved using each method with μ_X and σ_X such that $u(X, \mu_X) = (\theta_{PT} + \theta_{gr})/2$ and $u(X, \mu_X + \sigma_X) = \max\{\theta_{PT}, \theta_{gr}\}$.

6.4 The Effect of the Antithetic Variate Technique

A simple experiment was performed to test the effectiveness of the antithetic variate sampling technique used in MC simulation, where the expected higher utility of two purchases was estimated. In the case where only complete bundle purchases can be made (chapter 4), this test is similar to the problem of calculating the expected utility of a comparison set (where each purchase represents a bundle purchase). In the case where partial bundle purchases can be made (chapter 5), the test can be viewed as solving the problem of computing the expected utility of a future decision point (where each purchase represents a choice at that point).

Recall the concepts of relative mean and relative standard deviation as discussed in section 6.1.3. In the test, one item had varying relative means and standard deviations, while the other's parameters were fixed at (0,1). The test bed consisted of every pair (μ_r, σ_r) where μ_r was raised by increments of 0.05 from -2 to 0, and σ_r was raised by increments of 0.05 from 0.4 to 2. These ranges are consistent with those used in section 6.1.3. Simulations were run to compute the expected higher value for each case within a standard error of .001. Table 6.11 shows the average variance over all test cases for each of crude MC and MC with antithetic variate sampling. Note that to achieve a standard error of $\sqrt{\frac{s^2}{n}} = .001$, the sample size n must be 10^6 times the sample variance s^2. MC with antithetic variate sampling therefore required an average sample size of about 160,000, as opposed to 904,000 for crude MC, or roughly a little more than 1/6. Taking into consideration the fact that this method requires twice as much work since a second estimator is used, we see that the total amount of work required with antithetic sampling is only about 1/3 that of crude MC.

Average variance with crude MC:	0.904
Average variance with antithetic variate MC:	0.160

Table 6.11: Average variance during testing

Chapter 7

Conclusions

7.1 Thesis Results

This thesis describes a theory for decision-making in a dynamic purchasing environment where one of possibly many bundles of items must be purchased from possibly many suppliers. The *online combinatorial purchasing problem* is defined as the problem with which a purchase agent in such an environment is faced when deciding which items, from whom and at what time to buy in order to maximize overall satisfaction. Expected utility maximization is used as the criterion on which decision-making is based. To facilitate the exchange of probabilistic and temporal information between suppliers and purchasers, the PQR protocol is defined. This protocol dictates when information will be known by the purchaser about items such as cost, distribution of possible outcomes on cost, the time a quote will be obtained, and the time a quote will expire. The theory considers two distinct situations: 1) the situation in which only complete bundle purchases are made (i.e. a bundle is chosen in which all items have open quotes, and all items are purchased at once) and 2) the situation in which partial bundle purchases are allowed.

In the situation where only complete bundle purchases can be made, a technique is presented that provably yields higher expected utility than simply pursuing the best bundle. This method capitalizes on the variability of item prices in bundles that are available during the same time, and the influence of this variability on the utility one can expect to achieve when purchasing during this time. Each time a bundle b is about to expire, the buyer is instructed to make the purchase iff there does not exist a period of time (referred to as a comparison interval) such that the expected highest utility of all bundles available during that entire period (referred to as a comparison set) is higher than that of the current bundle purchase. The technique is shown to provide a minimum improvement of $\int_j^k (k-x)p_b(x)\,dx$ over the strategy of buying a bundle iff it has the highest expected utility over all bundles, where j is the highest expected utility over all bundles, k is the expected utility over all comparison sets, and p_b is the probability distribution function for the utility of buying the current bundle.

The problem is shown to be much more difficult when partial bundle purchases are permitted. The purchase procedure tree is introduced as a structure for modeling the sequence of decisions and purchases that must be made on the way to ultimately procuring a bundle. Conventional decision trees are shown to be unnecessarily large when used to solve decisions in the purchase procedure, and therefore a new type of decision tree referred to as the QR-tree is proposed. This tree models the future decisions and purchases that will affect the value of each choice at the present decision, but orders the purchase nodes by quote time, rather than the actual purchase (rescind) time. This reordering is done in such a way that the expected utility of the resulting tree is equal to that of the original purchase procedure tree. As a result, purchases for which the costs will be known at a particular decision time will be grouped together with that decision. These groupings are done for convenience in the proposed computation methods. There are three such methods: the restricted q-horizon method, the q-subset discretization method, and the classification tree method. Each uses Monte Carlo simulation to estimate the expected utility of each choice in a decision. The restricted q-horizon method simplifies the problem of computing the expected utility of a future decision by only considering the price outcomes for items that do not extend past subsequent decisions. The inaccuracy induced by ignoring this part of the q-horizon (i.e. the q-subhorizon of the next descendent decision node) is then resolved naïvely by using a greedy algorithm. The expected utility of this decision procedure is proven to be at least as high as the expected utility of using the greedy method of always pursuing the bundle with the highest expected utility. The q-subset discretization method considers each item represented by nodes in the q-subhorizons (i.e. the q-subsets) to have three discrete outcomes, as developed by Pearson and Tukey. This technique is shown experimentally to yield the most accurate estimates of all methods tested. The classification tree method uses data compiled by the q-subset discretization method at each decision node to build a classification tree (referred to as a decision tree in machine learning). The QR-tree is then simulated top-down, using these classification trees to classify outcomes into utilities to make decisions. This method is shown to provide an underestimate of the true utility when a reasonable restriction is placed on the execution of the purchase procedure. Testing shows that this estimate is also very close to the true expected utility.

The approach and techniques described in this thesis are novel in that the utility of future decision points are considered when assessing the utility of current choices. While decision trees accomplish this for less complex decision problems, for reasons described in this thesis they are not feasible in this context. Recent literature on decision-making in dynamic purchasing environments supports the claim that this approach is novel by arguing that computing expected utility of future choices is too difficult and error-prone, and therefore not worthwhile given the fact that ignoring future information has the desirable quality of guaranteeing a lower bound on expected utility (see [PBBP01, BPJ02, PBB03] for example). The techniques presented in this thesis therefore advance the state-of-the-art by demonstrating efficient and effective methods for including incomplete information on future decisions in utility computation of current choices, giving the purchaser more precise information as to the value of a choice.

7.2 Thesis Contributions

The main contributions of this thesis are as follows:

- The problem of deciding which of possibly many bundles to purchase, referred to as the online combinatorial purchasing problem (OCPP), is formalized. Techniques for determining utility functions in this domain, as well as computing expected utilities of bundle purchases and times during which multiple choices are available, are developed.

- The Prequote-Quote-Rescind (PQR) protocol is proposed. This protocol allows for the exchange of information necessary for a purchase agent to make informed decisions in the OCPP. It defines the communication rules under which buyers learn about a potential quote, the time a quote will be offered and time a quote will be rescinded, as well as the rules for communicating the quote itself and acceptance/rejection of the quote.

- A decision procedure is developed that leads the buyer to buy during the period of time during which the expected highest purchase utility of the bundles available during that entire time is maximized. This decision procedure is proven to always yield higher expected utility to the buyer than simply pursuing the bundle with the highest expected purchase utility.

- The QR-tree is introduced as a structure for making purchasing decisions. The tree is shown to be exponentially smaller than a conventional decision tree used in this domain. Three Monte Carlo (MC) algorithms are presented to solve the tree to closely estimate the buyer's expected utility of each choice. These methods are shown to have a run time that grows linearly with the size of the tree, and are thus more computationally feasible than any method that solves conventional decision trees. Basing decisions on estimates computed using the restricted q-horizon method is proven to always give the buyer a higher expected utility than the greedy approach of pursuing the best bundle. The classification tree method is shown to provide a tight lower bound on the true expected utility of any choice. Finally, the q-subset discretization method is shown to provide the most accurate estimate of all, with a standard error just $\frac{1}{26}$ that of the greedy method that judges the expected utility of a choice to be the maximum expected utility of all bundles that can potentially be procured as a result of that choice.

- Antithetic variate sampling is shown to be an effective variance reduction technique in Monte Carlo simulation in this domain. For the simple case of finding the expected highest utility of two simultaneous choices, antithetic variate sampling is shown to reduce the variance (and thus the number of simulations needed) to $\frac{1}{6}$ that of crude Monte Carlo.

7.3 Future Work

One idea for future work is to explore the addition of negotiation to the PQR protocol. Not only could the quoted prices be negotiated but also the quote and rescind times themselves. Consider the situation where a buyer may have to decide between buying item A or item B, and the decision must be made at A's rescind time, which happens to be just before B's quote time. Since it would be desirable to delay the decision until after B's quote time when both outcomes are known so a more informed decision can be made, then this delay must therefore increase expected utility. Thus it would be beneficial to the buyer to obtain either a later rescind time for A or an earlier quote time for B. In fact, a rational buyer should be willing to offer money in return for this time change, if the decrease in expected utility caused by this extra expenditure is less than the increase caused by the time change. Thus the opportunity for negotiation occurs.

Another addition to the PQR protocol that is worth investigation is to allow for more probabilistic parameters in information exchange. Thus far, the only such parameter considered is the price outcome, while t_q and t_r values are always known for certain. However, one could allow for the possibility that a quote may not be submitted at all after the prequote time. The supplier could say (or the buyer could subjectively decide), for example, that there is a 75% chance that a price will be quoted at time t_q. Another alternative is that a quote may be promised, but perhaps only a probability measure is known for the outcomes of the quote and rescind times. A supplier agent could say "I'll give you a quote sometime between day 3 and day 5, and it will be open for exactly 4 days from the time it is offered." A new theory would have to be developed to solve purchasing problems in this domain, since comparison sets, comparison intervals, q-horizons and q-subhorizons would be uncertain, but the work in this thesis would certainly provide a good starting point.

A different direction in which to take this research is into the domain of multiple auction decision making. Recent work by Preist *et al.* [PBB01], Anthony and Jennings [AJ02] and Byde *et al.* [BPJ02] has begun to concentrate efforts on determining strategies for participating in several auctions at once. Here the buyer has the problem of deciding whether to participate in an auction and how much to bid, based on the auctions into which he has already entered, and what new auctions are expected. These decisions are subject to the constraint that the buyer often prefers not to win more than one auction in which he has bid on the same item. Making this even more difficult is the fact that there may be different auction types (e.g. English, Dutch, sealed-bid). To see how the research presented in this thesis can be applied to such a problem, one can view each auction in the multi-auction problem as a quote. Like quotes, each auction has a known start time, a known finish time and can result in the purchase of an item. Unlike the OCPP, however, the buyer is not in complete control of whether he will get the item if he chooses. On the other hand, with enough knowledge of the type of auction and the behaviour of the participants, the buyer may be able to construct a reasonable probability measure on the outcome of the auction. Instead of judging the probability of price outcomes, as is done in the OCPP, one could instead judge the probability of winning the auction for each bid amount. This transformation could allow for the development of decision making techniques similar to those presented in this thesis.

Bibliography

[AA63] F. J. Anscombe and R. J. Aumann. A definition of subjective probability. *Annals of Mathematical Statistics*, 34:199–205, 1963.

[AJ02] P. Anthony and N. R. Jennings. Evolving bid strategies for multiple auctions. In *Proc. 15th European Conf. on AI (ECAI-2002)*, pages 178–182, Lyon, France, 2002.

[ATY00] A. Andersson, M. Tenhunen, and F. Ygge. Integer programming for combinatorial auction winner determination. In *International Conference on Multiagent Systems (ICMAS)*, pages 39–46, 2000.

[BAD03] A. Baklizi and W. Abu-Dayyeh. Shrinkage estimation of $p(y < x)$ in the exponential case. *Communications in Statistics, Simulation and Computation*, 32(1):31–42, 2003.

[Ber38] D. Bernoulli. Specimen theoriae novae de mensura sortis. *Commentarii Acasemia Scientiarum Imperialis Petropolitanae*, 5:175–192, 1738.

[Ber54] D. Bernoulli. Exposition of a new theory on the measurement of risk. *Econometrica*, 22, 1954. Translation of 1738 version [Ber38].

[BEY98] A. Borodin and R El-Yaniv. *Online Computation and Competitive Analysis.* Cambridge University Press, 1998.

[BPJ02] A. Byde, C. Preist, and N. R. Jennings. Decision procedures for multiple auctions. In *Proc. 1st Int Joint Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS'02)*, pages 613–620, Bologna, Italy, 2002.

[BRT88] D. E. Bell, H. Raiffa, and A. Tversky. *Decision Making: Descriptive, Normative, and Prescriptive Interactions.* Cambridge University Press, 1988.

[BS03a] S. Buffett and B. Spencer. A decision procedure for bundle purchasing with incomplete information on future prices. *International Journal of Electronic Commerce*, 2003. Accepted December 2003. To appear.

[BS03b] S. Buffett and B. Spencer. Efficient monte carlo decision tree solution in dynamic purchasing environments. In *Proc. International Conference on Electronic Commerce (ICEC2003)*, pages 31–39, Pittsburgh, PA, USA, 2003.

[BSZ02] A. Blum, T. Sandholm, and M. Zinkevich. Online algorithms for market clearing. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 971–980, 2002.

[CH99] B. Chandra and M. M. Halldórsson. Greedy local search and weighted set packing approximation. In *10th Annual SIAM-ACM Symposium on Discrete Algorithms (SODA)*, January 1999.

[Cre01] CreativeGood. Web page: http://www.creativegood.com. Date accessed: Aug 1, 2001, 2001.

[DDN01] H. M. Deitel, P. J. Deitel, and T. R. Nieto. *E-Business and E-Commerce: How to Program.* Prentice Hall, Inc., Upper Saddle River, NJ, USA, 2001.

[dVV00] S. de Vries and R. Vohra. Combinatorial auctions: A survey. url = "citeseer.nj.nec.com/devries01combinatorial.html", 2000.

[Fis65] P. C. Fishburn. Independence in utility theory with whole product sets. *Operations Research*, 13:28–45, 1965.

[Fis68] P. C. Fishburn. Utility theory. *Management Science*, 14:335–378, 1968.

[Fis70] P. C. Fishburn. *Utility Theory for Decision Making.* John Wiley and Sons, Inc., 1970.

[FLBS99] Y. Fujishima, K. Leyton-Brown, and Y. Shoham. Taming the combinatorial complexity of combinatorial auctions: optimal and approximate approaches. In *International Joint Conference on Artificial Intelligence (IJCAI-99)*, 1999.

[FR01] Inc Forrester Research. Web page: http://www.forrester.com. Date accessed: Aug 1, 2001, 2001.

[GC01] J. Greene and P. Cohen. Idc bulletin: ecommerce in canada: 2000-2005, icmm v7.3. Document #: CA070IEH, Publication Date: Dec 2001, 2001.

[GW99] P. Goodwin and G. Wright. *Decision Analysis for Management Judgment.* Chichester John Wiley & Sons, Ltd., New York, NY, 1999.

[Hal98] M. M. Halldórsson. Approximations of independent graphs in sets. In *The First International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, pages 1–14, 1998.

[HH64] J. M. Hammersley and D. C. Handscombe. *Monte Carlo Methods.* Wiley, New York, 1964.

[HL97] M. M. Halldórsson and H. C. Lau. Low-degree graph partitioning via local search with applications to constraint satisfaction, max cut, and 3-coloring. *Journal of Graph Algo. Applic.*, 1(3):1–13, 1997.

[HM56] J. M. Hammersley and K. W. Morton. A new Monte Carlo technique: antithetic variates. In *Cambridge Phil. Soc.*, volume 52, pages 449–475, 1956.

[Hoc83] D. S. Hochbaum. Efficient bounds for the stable set, vertex cover, and set packing problems. *Discrete Applied Mathematics*, 6:243–254, 1983.

[Hor02] J. Horton. Personal communication, Aug 12, 2002.

[HS65] R. F. Hespos and P. A. Strassmann. Stochastic decision trees for the analysis of investment decisions. *Management Science*, 11(10):B244–B259, 1965.

[KB83] D. L. Keefer and S. E. Bodily. Three point approximations for continuous random variables. *Management Science*, 29(5):595–609, 1983.

[Kee03] F. Keenan. The price is really right. *BusinessWeek Online*, March 31, 2003.

[Kor85] R. E. Korf. Depth-first iterative deepening: an optimal admissible tree search. *Artificial Intelligence*, 27:97–109, 1985.

[KR76] R. L. Keeney and H. Raiffa. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs.* John Wiley and Sons, Inc., 1976.

[Kre88] D. M. Kreps. *Notes on the Theory of Choice.* Westview Press, 1988.

[LBST00] K. Leyton-Brown, Y. Shoham, and M. Tennenholtz. An algorithm for multi-unit combinatorial auctions. In *AAAI*, pages 56–61, 2000.

[Lub01] F. Luban. Integrated decision analysis procedure for investment projects. In *Proceedings of the International Conference on Modeling and Simulation in Distributed Applications (MS2001)*, pages 776–780, Changsha, Hunan, China, 2001.

[Mag64] J. F. Magee. Decision trees for decision making. *Harvard Business Review*, 42 (July-August):126–139, 1964.

[MMV95] J. K. MacKie-Mason and H. R. Varian. Generalized vickrey auctions. Technical report, University of Michigan, 1995.

[MP68] R. F. Meyer and J. W. Pratt. The consistent assessment and fairing of preference functions. *IEEE Systems Science and Cybernetics*, SSC-4:270–278, 1968.

[MS73] W. Mendenhall and R. L. Scheaffer. *Mathematical Statistics with Applications*. Duxbury Press, Mass., 1973.

[MU49] N. Metropolis and S. Ulam. The Monte Carlo method. *Journal of the American Statistical Association*, 44(247):335–341, 1949.

[Par99] D. C. Parkes. *i*bundle: An efficient ascending price bundle auction. In *ACM Conference on E-commerce*, 1999.

[PBB01] C. Preist, A. Byde, and C. Bartolini. Economic dynamics of agents in multiple auctions. In *AGENTS'01*, pages 545–551, Montreal, Quebec, Canada, 2001.

[PBB03] C. Preist, C. Bartolini, and A. Byde. Agent-based service composition through simultaneous negotiation in forward and reverse auctions. In *Proceedings of the 4th ACM Conference on Electronic Commerce*, pages 55–63, San Diego, California, USA, 2003.

[PBBP01] C. Priest, A. Byde, C. Bartolini, and G. Piccinelli. Towards agent-based service composition through negotiation in multiple auctions. In *AISB'01 Symp. on Inf. Agents for Electronic Commerce*, 2001.

[PBF91] P.E. Pfeifer, S.E. Bodily, and S.C. Frey. Pearson-tukey three-point approximations versus monte carlo simulation. *Decision Sciences*, 22(1):74–90, 1991.

[PT65] E. S. Pearson and J. W. Tukey. Approximating means and standard deviations based on distances between percentage points of frequency curves. *Biometrika*, 52(3-4):533–546, 1965.

[PU00a] D. C. Parkes and L. H. Ungar. Iterative combinatorial auctions: Theory and practice. In *AAAI*, pages 74–81, 2000.

[PU00b] D. C. Parkes and L. H. Ungar. Preventing strategic manipulation in iterative auctions: proxy agents and price-adjustment. In *AAAI*, pages 82–89, 2000.

[Qui86] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[Rai68] H. Raiffa. *Decision Analysis: Introductory Lectures on Choices under Uncertainty.* Addison-Wesley Publishing Company Inc., Massachusetts, USA, 1968.

[RPH98] M. H. Rothkopf, A. Pekec, and R. M. Harstad. Computationally manageable combinatorial auctions. *Management Science*, 44(8):1131–1147, 1998.

[RSB82] S. J. Rassenti, V. L. Smith, and R. L. Bulfin. A combinatorial auction mechanism for airport time slot allocation. *Bell J. of Economics*, 13:402–417, 1982.

[San00] T. Sandholm. emediator: A next generation electronic commerce server. In *International Conference on Autonomous Agents (AGENTS)*, pages 73–96, June 2000.

[San02] T. Sandholm. An algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135:1–54, 2002.

[Sav54] L. J. Savage. *The Foundations of Statistics.* Wiley, New York, USA, 1954.

[SDP03] M.S. Shell, P.G. Debenedetti, and A.Z. Panagiotopoulos. An improved monte carlo method for the direct calculation of the density of states. *Journal of Chemical Physics*, 119(18):9406–9411, 2003.

[SS01] T. Sandholm and S. Suri. Market clearability. In *International Joint Conference on Artificial Intelligence (IJCAI-01)*, 2001.

[SSGL01] T. Sandholm, S. Suri, A. Gilpin, and D. Levine. Cabob: A fast optimal algorithm for combinatorial auctions. In *International Joint Conference on Artificial Intelligence (IJCAI-01)*, pages 1102–1108, 2001.

[Ten00] M. Tennenholtz. Some tractable combinatorial auctions. In *AAAI*, pages 98–103, 2000.

[Vic61] W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.

[vNM47] J. von Neumann and O. Morgenstern. *Theory of games and economic behaviour, 2nd ed.* Princeton University Press, Princeton NJ, USA, 1947.

[Wal50] A. Wald. *Statistical Decision Functions.* Wiley, New York, 1950.

[WWW98] P. R. Wurman, M. P. Wellman, and W. E. Walsh. The michigan internet auctionbot: A configurable auction server for human and software agents. In *AGENTS*, pages 301–308, 1998.

Appendix A

Tables

Table A.1: Item price means and standard deviations for each instance in tests described in section 6.3.3 (mean appears above standard deviation)

1	A	1.046	B	1.027	C	1.020	D	0.977	E	1.076	F	0.955
		0.080		0.183		0.111		0.328		0.314		0.043
	G	0.929	H	1.009	J	0.921	K	0.982	L	1.098	X	2.729
		0.008		0.321		0.208		0.259		0.162		0.007
2	A	0.975	B	1.098	C	0.996	D	0.982	E	1.043	F	1.027
		0.022		0.187		0.183		0.060		0.150		0.262
	G	1.043	H	1.017	J	0.933	K	1.059	L	1.015	X	2.754
		0.219		0.206		0.104		0.221		0.242		0.012
3	A	1.088	B	0.912	C	1.026	D	1.087	E	1.077	F	1.040
		0.071		0.066		0.076		0.082		0.069		0.184
	G	0.954	H	1.044	J	0.939	K	1.042	L	0.982	X	2.891
		0.047		0.141		0.199		0.153		0.258		0.002
4	A	1.050	B	1.076	C	0.989	D	1.092	E	0.927	F	0.982
		0.168		0.131		0.080		0.060		0.026		0.252
	G	0.954	H	0.913	J	0.968	K	0.989	L	0.921	X	2.776
		0.280		0.081		0.044		0.252		0.217		0.003
5	A	0.985	B	1.098	C	1.093	D	1.037	E	0.942	F	0.938
		0.224		0.267		0.178		0.194		0.029		0.323
	G	1.016	H	1.075	J	1.057	K	0.909	L	0.946	X	2.728
		0.332		0.004		0.014		0.229		0.320		0.006

6	A	0.947	B	1.063	C	1.057	D	1.042	E	0.992	F	1.081
		0.320		0.332		0.182		0.172		0.319		0.058
	G	1.017	H	0.944	J	1.086	K	1.056	L	1.085	X	2.676
		0.231		0.277		0.192		0.328		0.280		0.006
7	A	1.054	B	1.058	C	1.089	D	0.987	E	1.010	F	0.959
		0.005		0.199		0.155		0.012		0.147		0.209
	G	0.910	H	0.959	J	0.969	K	0.985	L	1.088	X	2.839
		0.121		0.268		0.031		0.033		0.319		0.011
8	A	1.016	B	1.023	C	1.052	D	0.991	E	1.060	F	0.901
		0.101		0.264		0.159		0.323		0.104		0.278
	G	0.911	H	1.028	J	0.997	K	0.932	L	1.027	X	2.699
		0.021		0.207		0.210		0.132		0.279		0.002
9	A	0.933	B	1.041	C	0.975	D	1.016	E	1.003	F	1.088
		0.029		0.085		0.273		0.105		0.018		0.305
	G	0.971	H	0.968	J	1.033	K	0.977	L	1.077	X	2.674
		0.021		0.262		0.299		0.167		0.257		0.006
10	A	1.095	B	1.006	C	0.939	D	1.021	E	1.053	F	1.031
		0.125		0.150		0.277		0.083		0.308		0.040
	G	0.972	H	1.037	J	1.061	K	0.924	L	1.016	X	2.716
		0.254		0.202		0.144		0.266		0.217		0.006
11	A	1.008	B	1.020	C	0.968	D	0.926	E	1.088	F	1.043
		0.174		0.029		0.170		0.105		0.227		0.294
	G	1.083	H	1.064	J	1.068	K	0.907	L	0.983	X	2.745
		0.081		0.137		0.239		0.198		0.233		0.013
12	A	1.037	B	0.978	C	1.058	D	1.051	E	1.005	F	0.929
		0.088		0.009		0.189		0.308		0.116		0.102
	G	1.071	H	1.009	J	0.987	K	0.901	L	1.056	X	2.806
		0.252		0.273		0.016		0.172		0.048		0.003
13	A	1.098	B	0.988	C	1.038	D	0.916	E	0.936	F	1.008
		0.296		0.218		0.019		0.034		0.096		0.331
	G	0.928	H	1.083	J	1.063	K	0.921	L	0.982	X	2.766
		0.095		0.056		0.131		0.047		0.319		0.003
14	A	1.061	B	0.952	C	1.002	D	0.921	E	0.985	F	0.950
		0.059		0.283		0.022		0.012		0.052		0.066
	G	0.928	H	0.952	J	1.091	K	1.068	L	0.921	X	2.712
		0.328		0.328		0.309		0.146		0.279		0.001
15	A	0.974	B	0.967	C	1.032	D	1.026	E	1.020	F	0.962
		0.108		0.162		0.249		0.034		0.305		0.320
	G	1.039	H	0.979	J	1.097	K	1.051	L	1.089	X	2.734
		0.156		0.264		0.071		0.078		0.295		0.017

16	A	0.916	B	1.094	C	1.025	D	0.990	E	1.017	F	1.074
		0.325		0.112		0.112		0.123		0.194		0.193
	G	1.026	H	0.980	J	1.081	K	1.094	L	0.931	X	2.743
		0.096		0.113		0.019		0.203		0.095		0.009
17	A	0.959	B	0.901	C	1.024	D	0.936	E	1.077	F	0.983
		0.070		0.006		0.291		0.165		0.085		0.044
	G	1.082	H	1.032	J	0.903	K	1.039	L	1.083	X	2.680
		0.220		0.027		0.253		0.136		0.243		0.001
18	A	1.085	B	1.037	C	0.954	D	0.944	E	1.046	F	1.014
		0.164		0.273		0.190		0.279		0.141		0.080
	G	0.947	H	0.909	J	0.923	K	1.068	L	0.935	X	2.688
		0.169		0.256		0.082		0.125		0.090		0.003
19	A	1.071	B	1.063	C	0.907	D	0.929	E	0.965	F	0.904
		0.061		0.079		0.018		0.043		0.084		0.177
	G	1.063	H	1.025	J	0.944	K	0.952	L	1.001	X	2.793
		0.206		0.332		0.032		0.159		0.270		0.012
20	A	1.034	B	1.028	C	1.070	D	0.934	E	1.015	F	1.046
		0.158		0.144		0.022		0.020		0.040		0.245
	G	1.063	H	1.093	J	0.972	K	1.099	L	0.940	X	2.803
		0.105		0.271		0.211		0.258		0.230		0.008
21	A	1.095	B	1.037	C	1.050	D	0.999	E	0.946	F	0.925
		0.032		0.020		0.185		0.080		0.020		0.141
	G	0.920	H	0.967	J	1.047	K	0.919	L	1.066	X	2.871
		0.282		0.054		0.326		0.133		0.168		0.001
22	A	1.057	B	1.002	C	1.014	D	1.003	E	0.995	F	1.067
		0.129		0.085		0.188		0.058		0.310		0.209
	G	0.921	H	1.036	J	1.075	K	1.066	L	1.005	X	2.792
		0.182		0.327		0.171		0.232		0.128		0.019
23	A	1.064	B	1.085	C	0.960	D	1.068	E	0.964	F	0.959
		0.119		0.261		0.109		0.035		0.191		0.282
	G	1.089	H	0.988	J	1.066	K	0.912	L	0.964	X	2.748
		0.031		0.236		0.224		0.282		0.236		0.008
24	A	0.973	B	1.098	C	0.944	D	0.993	E	1.081	F	1.029
		0.302		0.228		0.089		0.220		0.088		0.192
	G	0.950	H	0.960	J	0.908	K	1.026	L	0.988	X	2.654
		0.096		0.288		0.016		0.229		0.054		0.002
25	A	1.069	B	1.056	C	0.923	D	1.018	E	1.073	F	1.011
		0.294		0.220		0.291		0.054		0.039		0.201
	G	0.945	H	0.909	J	0.916	K	1.004	L	1.055	X	2.671
		0.055		0.089		0.304		0.314		0.010		0.002

26	A	1.051	B	1.082	C	1.030	D	1.023	E	1.075	F	0.958
		0.326		0.114		0.207		0.313		0.292		0.288
	G	1.011	H	0.969	J	0.973	K	1.019	L	0.977	X	2.679
		0.329		0.282		0.284		0.030		0.325		0.012
27	A	1.018	B	1.047	C	1.039	D	1.008	E	1.042	F	1.043
		0.288		0.091		0.123		0.129		0.271		0.034
	G	1.062	H	1.093	J	0.965	K	1.035	L	1.060	X	2.823
		0.325		0.105		0.099		0.114		0.150		0.007
28	A	0.920	B	1.042	C	1.026	D	0.973	E	0.942	F	0.979
		0.107		0.046		0.184		0.130		0.077		0.174
	G	0.905	H	1.052	J	1.084	K	1.095	L	0.919	X	2.707
		0.118		0.212		0.301		0.166		0.325		0.005
29	A	1.035	B	1.002	C	0.986	D	0.918	E	1.006	F	0.930
		0.228		0.020		0.037		0.317		0.117		0.227
	G	0.902	H	1.024	J	0.945	K	0.947	L	0.960	X	2.665
		0.130		0.216		0.136		0.194		0.072		0.003
30	A	0.924	B	0.979	C	1.000	D	1.003	E	1.070	F	0.969
		0.198		0.206		0.210		0.279		0.279		0.275
	G	0.949	H	0.928	J	0.944	K	0.958	L	1.088	X	2.582
		0.235		0.178		0.030		0.167		0.064		0.008
31	A	1.090	B	0.944	C	1.009	D	0.987	E	1.038	F	1.054
		0.160		0.183		0.127		0.095		0.258		0.021
	G	1.000	H	1.052	J	0.935	K	0.974	L	0.970	X	2.743
		0.231		0.001		0.179		0.251		0.221		0.005
32	A	0.947	B	0.953	C	0.943	D	1.073	E	1.050	F	0.991
		0.169		0.146		0.044		0.317		0.216		0.239
	G	1.007	H	1.070	J	1.047	K	0.924	L	0.953	X	2.620
		0.311		0.233		0.324		0.141		0.295		0.004
33	A	0.969	B	0.928	C	1.095	D	1.017	E	1.031	F	0.983
		0.184		0.069		0.245		0.188		0.129		0.117
	G	1.061	H	1.061	J	1.000	K	0.966	L	0.959	X	2.739
		0.130		0.328		0.113		0.052		0.159		0.005
34	A	1.071	B	0.922	C	1.082	D	0.982	E	0.931	F	0.919
		0.003		0.023		0.305		0.190		0.268		0.257
	G	0.904	H	1.021	J	0.919	K	1.026	L	1.019	X	2.720
		0.257		0.102		0.315		0.104		0.001		0.033
35	A	1.068	B	1.097	C	0.928	D	1.003	E	0.962	F	0.982
		0.270		0.120		0.074		0.019		0.056		0.266
	G	1.074	H	0.924	J	0.996	K	0.905	L	1.023	X	2.764
		0.266		0.025		0.224		0.071		0.295		0.008

36	A	0.925	B	0.906	C	1.063	D	1.089	E	0.975	F	0.918
		0.279		0.083		0.324		0.241		0.014		0.150
	G	1.080	H	0.943	J	0.908	K	1.055	L	1.006	X	2.613
		0.013		0.257		0.035		0.294		0.036		0.002
37	A	0.965	B	1.084	C	0.995	D	0.953	E	1.027	F	1.080
		0.090		0.321		0.176		0.131		0.016		0.072
	G	0.936	H	0.956	J	0.914	K	0.972	L	0.987	X	2.674
		0.218		0.221		0.038		0.148		0.024		0.001
38	A	1.073	B	1.097	C	0.979	D	1.078	E	0.944	F	0.950
		0.273		0.288		0.156		0.316		0.246		0.316
	G	0.997	H	0.927	J	0.957	K	1.086	L	1.011	X	2.679
		0.283		0.273		0.164		0.094		0.176		0.015
39	A	1.034	B	1.044	C	0.991	D	0.903	E	1.095	F	0.901
		0.204		0.253		0.153		0.282		0.134		0.292
	G	1.030	H	1.039	J	1.025	K	1.048	L	0.929	X	2.686
		0.008		0.152		0.056		0.284		0.024		0.010
40	A	1.090	B	1.052	C	0.926	D	0.988	E	0.908	F	1.038
		0.213		0.216		0.070		0.171		0.092		0.176
	G	1.037	H	1.057	J	0.937	K	0.999	L	0.911	X	2.739
		0.088		0.051		0.200		0.174		0.098		0.004
41	A	0.912	B	1.027	C	1.077	D	0.933	E	1.088	F	1.030
		0.228		0.139		0.271		0.042		0.005		0.054
	G	1.091	H	1.048	J	1.090	K	1.041	L	0.918	X	2.698
		0.241		0.146		0.323		0.085		0.295		0.001
42	A	1.021	B	1.040	C	1.062	D	1.058	E	1.005	F	0.900
		0.078		0.105		0.251		0.226		0.235		0.298
	G	0.952	H	1.020	J	0.933	K	0.955	L	0.941	X	2.721
		0.305		0.198		0.115		0.032		0.113		0.020
43	A	0.988	B	1.006	C	1.071	D	1.043	E	0.973	F	0.985
		0.040		0.083		0.167		0.042		0.214		0.043
	G	1.003	H	0.944	J	0.924	K	0.971	L	1.024	X	2.797
		0.301		0.021		0.264		0.116		0.271		0.013
44	A	1.024	B	0.977	C	1.035	D	1.088	E	0.933	F	1.090
		0.169		0.265		0.278		0.042		0.104		0.066
	G	0.996	H	1.095	J	0.992	K	1.018	L	0.910	X	2.713
		0.088		0.163		0.178		0.032		0.185		0.002
45	A	0.958	B	1.048	C	1.057	D	1.073	E	1.017	F	0.905
		0.310		0.181		0.263		0.278		0.295		0.296
	G	0.959	H	0.959	J	0.968	K	0.963	L	0.971	X	2.609
		0.062		0.265		0.043		0.313		0.129		0.012

46	A	0.920	B	1.013	C	1.021	D	1.077	E	1.067	F	1.047
		0.073		0.246		0.266		0.256		0.252		0.031
	G	0.960	H	1.027	J	0.996	K	0.910	L	0.910	X	2.640
		0.295		0.204		0.222		0.078		0.089		0.007
47	A	1.033	B	1.027	C	1.050	D	0.982	E	0.902	F	1.060
		0.122		0.125		0.159		0.278		0.171		0.285
	G	1.071	H	1.054	J	1.002	K	1.093	L	1.077	X	2.793
		0.123		0.140		0.317		0.011		0.105		0.014
48	A	1.017	B	0.999	C	1.088	D	1.027	E	0.932	F	0.946
		0.226		0.114		0.035		0.034		0.211		0.171
	G	1.022	H	1.059	J	0.914	K	0.978	L	1.073	X	2.820
		0.247		0.067		0.036		0.155		0.151		0.012
49	A	0.912	B	1.010	C	0.937	D	1.012	E	1.095	F	0.989
		0.146		0.011		0.277		0.038		0.042		0.048
	G	0.932	H	1.064	J	1.067	K	1.085	L	1.066	X	2.706
		0.023		0.134		0.251		0.145		0.043		0.001
50	A	1.067	B	0.956	C	0.903	D	1.057	E	0.929	F	0.997
		0.212		0.064		0.028		0.167		0.241		0.013
	G	0.915	H	0.920	J	0.972	K	0.978	L	1.040	X	2.687
		0.318		0.129		0.173		0.016		0.026		0.005
51	A	1.054	B	0.983	C	1.056	D	1.042	E	1.048	F	1.087
		0.109		0.203		0.189		0.263		0.183		0.110
	G	1.031	H	1.036	J	0.993	K	1.062	L	0.906	X	2.806
		0.021		0.205		0.124		0.051		0.206		0.009
52	A	0.935	B	0.933	C	0.933	D	0.970	E	0.930	F	0.918
		0.059		0.146		0.278		0.265		0.058		0.223
	G	1.059	H	0.972	J	1.005	K	0.983	L	1.082	X	2.566
		0.257		0.225		0.228		0.210		0.286		0.005
53	A	1.048	B	0.947	C	0.963	D	1.075	E	0.964	F	0.931
		0.108		0.025		0.170		0.287		0.311		0.144
	G	0.970	H	0.999	J	1.011	K	0.966	L	1.050	X	2.720
		0.084		0.161		0.324		0.142		0.301		0.013
54	A	1.010	B	0.912	C	0.927	D	1.080	E	1.014	F	1.073
		0.204		0.090		0.174		0.265		0.267		0.212
	G	0.971	H	1.068	J	1.039	K	0.913	L	0.988	X	2.609
		0.317		0.100		0.169		0.241		0.261		0.005
55	A	0.990	B	0.919	C	1.082	D	0.905	E	0.916	F	1.029
		0.071		0.217		0.083		0.179		0.119		0.207
	G	1.069	H	0.971	J	0.937	K	0.943	L	1.090	X	2.682
		0.083		0.232		0.196		0.048		0.132		0.005

56	A	0.901	B	1.079	C	0.997	D	1.070	E	1.062	F	1.038
		0.326		0.279		0.023		0.194		0.136		0.249
	G	1.015	H	1.087	J	1.035	K	0.926	L	0.988	X	2.664
		0.331		0.019		0.060		0.090		0.014		0.001
57	A	0.995	B	1.090	C	1.045	D	1.055	E	1.026	F	1.081
		0.246		0.176		0.265		0.198		0.300		0.126
	G	0.926	H	1.092	J	0.987	K	1.034	L	0.980	X	2.751
		0.107		0.085		0.275		0.080		0.239		0.011
58	A	0.937	B	1.018	C	1.039	D	1.020	E	1.023	F	0.992
		0.130		0.126		0.129		0.287		0.188		0.332
	G	0.913	H	1.093	J	0.971	K	1.077	L	1.023	X	2.675
		0.047		0.268		0.223		0.205		0.039		0.008
59	A	0.999	B	1.027	C	1.019	D	1.085	E	0.953	F	1.071
		0.004		0.002		0.292		0.013		0.168		0.227
	G	0.969	H	0.967	J	1.046	K	1.097	L	0.949	X	2.816
		0.223		0.051		0.004		0.080		0.311		0.030
60	A	0.961	B	0.992	C	0.982	D	1.089	E	1.003	F	1.013
		0.101		0.067		0.295		0.324		0.125		0.295
	G	0.970	H	1.036	J	1.075	K	1.044	L	1.088	X	2.689
		0.123		0.323		0.183		0.179		0.271		0.004
61	A	1.068	B	0.987	C	1.014	D	1.035	E	1.021	F	1.092
		0.119		0.267		0.269		0.164		0.287		0.113
	G	1.064	H	1.051	J	0.958	K	0.973	L	1.091	X	2.751
		0.013		0.315		0.022		0.217		0.310		0.009
62	A	0.903	B	0.964	C	0.902	D	0.919	E	0.955	F	0.911
		0.064		0.259		0.208		0.216		0.266		0.024
	G	1.069	H	1.008	J	1.001	K	1.021	L	0.929	X	2.553
		0.121		0.061		0.055		0.085		0.101		0.009
63	A	0.965	B	0.974	C	1.082	D	0.984	E	1.086	F	0.990
		0.271		0.135		0.038		0.276		0.246		0.253
	G	0.926	H	1.082	J	1.076	K	1.040	L	1.055	X	2.712
		0.298		0.177		0.170		0.294		0.040		0.006
64	A	0.927	B	0.938	C	1.046	D	0.989	E	0.936	F	0.933
		0.034		0.200		0.041		0.254		0.203		0.321
	G	0.926	H	0.951	J	0.904	K	0.987	L	0.994	X	2.626
		0.198		0.117		0.015		0.059		0.333		0.016
65	A	1.095	B	0.962	C	1.018	D	1.054	E	0.983	F	1.053
		0.136		0.024		0.219		0.332		0.056		0.041
	G	1.041	H	1.044	J	1.071	K	0.996	L	1.085	X	2.851
		0.182		0.325		0.085		0.151		0.055		0.001

66	A	1.024	B	1.032	C	0.994	D	1.059	E	0.950	F	0.925
		0.079		0.028		0.291		0.064		0.225		0.261
	G	0.954	H	1.052	J	1.083	K	1.073	L	1.002	X	2.766
		0.060		0.209		0.313		0.113		0.134		0.019
67	A	0.939	B	0.993	C	0.954	D	1.004	E	1.014	F	1.076
		0.200		0.002		0.144		0.250		0.265		0.314
	G	0.951	H	1.023	J	0.944	K	0.925	L	1.043	X	2.636
		0.071		0.213		0.148		0.141		0.215		0.010
68	A	1.048	B	1.006	C	0.938	D	0.929	E	0.931	F	0.946
		0.049		0.302		0.124		0.101		0.162		0.224
	G	1.012	H	0.995	J	0.987	K	1.039	L	1.067	X	2.692
		0.136		0.264		0.273		0.087		0.033		0.010
69	A	0.918	B	0.993	C	0.907	D	1.030	E	1.046	F	1.086
		0.165		0.325		0.162		0.005		0.306		0.076
	G	0.986	H	0.987	J	1.078	K	1.019	L	0.970	X	2.633
		0.020		0.328		0.037		0.156		0.158		0.005
70	A	1.026	B	1.007	C	1.092	D	0.955	E	0.963	F	0.956
		0.015		0.292		0.142		0.189		0.203		0.319
	G	1.047	H	0.959	J	0.921	K	0.933	L	0.994	X	2.711
		0.085		0.046		0.162		0.103		0.309		0.015
71	A	0.941	B	0.968	C	1.051	D	1.100	E	1.026	F	0.907
		0.136		0.266		0.328		0.043		0.243		0.039
	G	1.043	H	0.930	J	0.982	K	0.986	L	1.035	X	2.667
		0.097		0.050		0.330		0.130		0.056		0.000
72	A	1.043	B	0.962	C	1.038	D	1.065	E	0.926	F	1.043
		0.289		0.221		0.055		0.044		0.049		0.180
	G	1.087	H	1.089	J	0.901	K	1.046	L	1.094	X	2.796
		0.224		0.157		0.198		0.183		0.231		0.004
73	A	0.972	B	1.087	C	1.096	D	1.089	E	0.970	F	0.928
		0.242		0.102		0.181		0.286		0.204		0.019
	G	0.923	H	0.924	J	0.995	K	1.083	L	0.906	X	2.726
		0.208		0.022		0.236		0.018		0.105		0.012
74	A	1.080	B	0.901	C	1.080	D	1.015	E	1.087	F	0.998
		0.092		0.069		0.161		0.137		0.101		0.263
	G	0.984	H	1.095	J	1.038	K	0.997	L	0.930	X	2.798
		0.272		0.074		0.028		0.298		0.256		0.003
75	A	0.995	B	1.061	C	1.040	D	0.959	E	0.951	F	0.949
		0.213		0.043		0.014		0.324		0.141		0.316
	G	0.980	H	1.067	J	1.099	K	1.049	L	0.971	X	2.712
		0.284		0.077		0.196		0.325		0.003		0.009

76	A	0.977	B	1.087	C	0.948	D	0.964	E	0.952	F	1.096
		0.245		0.270		0.263		0.249		0.061		0.069
	G	1.046	H	0.927	J	0.956	K	1.064	L	1.094	X	2.635
		0.224		0.270		0.176		0.041		0.318		0.001
77	A	0.937	B	1.034	C	0.959	D	0.989	E	0.903	F	1.047
		0.176		0.235		0.260		0.215		0.282		0.045
	G	1.079	H	1.039	J	1.025	K	1.026	L	1.012	X	2.639
		0.282		0.149		0.068		0.231		0.167		0.011
78	A	1.046	B	1.047	C	0.943	D	0.914	E	1.020	F	0.917
		0.026		0.202		0.240		0.066		0.179		0.289
	G	0.940	H	1.010	J	1.067	K	0.940	L	1.035	X	2.701
		0.014		0.200		0.193		0.178		0.318		0.003
79	A	0.961	B	1.008	C	0.903	D	1.059	E	1.083	F	1.068
		0.147		0.176		0.093		0.253		0.219		0.009
	G	0.937	H	0.982	J	0.928	K	0.993	L	1.077	X	2.691
		0.025		0.204		0.028		0.205		0.065		0.000
80	A	0.943	B	1.034	C	1.011	D	1.064	E	1.084	F	1.016
		0.179		0.069		0.009		0.178		0.139		0.095
	G	1.002	H	1.042	J	0.986	K	1.008	L	0.999	X	2.791
		0.299		0.063		0.008		0.254		0.047		0.002
81	A	1.092	B	0.996	C	1.039	D	0.929	E	0.981	F	0.931
		0.012		0.065		0.274		0.309		0.029		0.130
	G	1.052	H	0.944	J	0.924	K	0.948	L	0.907	X	2.738
		0.242		0.018		0.276		0.063		0.080		0.003
82	A	0.993	B	0.990	C	1.026	D	1.094	E	1.081	F	1.067
		0.164		0.019		0.002		0.310		0.168		0.271
	G	1.095	H	0.903	J	1.043	K	1.008	L	0.966	X	2.815
		0.036		0.125		0.145		0.116		0.255		0.008
83	A	0.982	B	0.963	C	0.909	D	0.918	E	1.029	F	1.095
		0.134		0.254		0.124		0.238		0.230		0.167
	G	1.097	H	0.963	J	0.919	K	0.962	L	0.906	X	2.570
		0.256		0.270		0.167		0.245		0.167		0.007
84	A	1.091	B	0.977	C	1.093	D	1.044	E	0.946	F	0.966
		0.317		0.220		0.104		0.089		0.128		0.077
	G	0.957	H	0.934	J	0.961	K	1.076	L	0.930	X	2.705
		0.321		0.321		0.293		0.192		0.318		0.003
85	A	1.028	B	0.932	C	1.065	D	0.992	E	1.050	F	1.073
		0.307		0.078		0.168		0.224		0.298		0.226
	G	1.090	H	1.079	J	1.068	K	1.029	L	1.090	X	2.787
		0.304		0.044		0.015		0.060		0.006		0.010

86	A	0.936	B	0.946	C	1.049	D	0.917	E	0.967	F	0.943
		0.157		0.045		0.148		0.075		0.195		0.137
	G	0.951	H	1.050	J	0.911	K	0.943	L	0.914	X	2.669
		0.036		0.096		0.140		0.007		0.157		0.015
87	A	1.031	B	1.069	C	1.039	D	1.022	E	0.977	F	1.063
		0.065		0.238		0.194		0.040		0.166		0.328
	G	0.934	H	0.955	J	0.928	K	1.092	L	1.059	X	2.819
		0.087		0.010		0.077		0.061		0.078		0.011
88	A	0.960	B	0.940	C	1.015	D	1.026	E	0.944	F	0.935
		0.007		0.242		0.267		0.105		0.002		0.214
	G	1.047	H	0.963	J	0.940	K	0.968	L	0.976	X	2.626
		0.298		0.228		0.306		0.023		0.157		0.001
89	A	1.055	B	0.951	C	1.064	D	1.012	E	0.908	F	0.978
		0.260		0.139		0.176		0.108		0.166		0.091
	G	0.957	H	0.967	J	1.092	K	1.076	L	0.969	X	2.725
		0.074		0.294		0.188		0.013		0.049		0.009
90	A	1.039	B	0.923	C	0.902	D	1.056	E	0.937	F	1.064
		0.031		0.128		0.052		0.198		0.206		0.309
	G	0.908	H	0.972	J	1.004	K	0.961	L	0.965	X	2.659
		0.198		0.222		0.240		0.157		0.095		0.008
91	A	0.952	B	0.937	C	0.931	D	0.961	E	0.972	F	1.077
		0.080		0.007		0.278		0.220		0.125		0.231
	G	1.019	H	0.999	J	1.011	K	0.944	L	0.933	X	2.664
		0.026		0.157		0.002		0.089		0.111		0.006
92	A	0.914	B	0.902	C	1.095	D	0.966	E	1.022	F	1.019
		0.176		0.072		0.281		0.198		0.082		0.299
	G	1.020	H	1.067	J	1.039	K	1.091	L	1.071	X	2.696
		0.259		0.097		0.181		0.188		0.071		0.005
93	A	1.020	B	1.097	C	0.927	D	0.911	E	1.040	F	1.098
		0.194		0.272		0.255		0.038		0.244		0.117
	G	0.914	H	1.082	J	0.922	K	1.019	L	1.074	X	2.689
		0.149		0.088		0.020		0.226		0.110		0.003
94	A	1.004	B	1.069	C	0.965	D	0.956	E	1.070	F	0.984
		0.298		0.261		0.130		0.127		0.284		0.002
	G	1.065	H	1.087	J	1.034	K	0.904	L	1.071	X	2.713
		0.273		0.015		0.072		0.037		0.157		0.003
95	A	0.918	B	1.083	C	0.994	D	1.061	E	0.905	F	0.996
		0.014		0.140		0.023		0.150		0.203		0.257
	G	0.976	H	0.914	J	1.040	K	1.087	L	1.038	X	2.719
		0.100		0.284		0.168		0.302		0.173		0.015

96	A	1.080	B	1.048	C	0.958	D	1.066	E	0.954	F	0.938
		0.110		0.205		0.027		0.127		0.160		0.325
	G	0.976	H	0.982	J	1.068	K	1.034	L	0.977	X	2.760
		0.333		0.223		0.013		0.068		0.133		0.007
97	A	1.048	B	1.072	C	0.930	D	0.931	E	1.002	F	1.059
		0.212		0.029		0.205		0.205		0.013		0.045
	G	1.091	H	1.076	J	1.035	K	1.043	L	1.068	X	2.801
		0.318		0.098		0.083		0.160		0.188		0.002
98	A	0.959	B	1.031	C	1.045	D	1.095	E	0.948	F	1.068
		0.134		0.091		0.145		0.221		0.030		0.087
	G	1.037	H	0.992	J	0.933	K	1.026	L	0.966	X	2.776
		0.233		0.218		0.281		0.202		0.071		0.001
99	A	1.065	B	1.027	C	1.077	D	1.040	E	0.966	F	0.946
		0.152		0.291		0.118		0.060		0.192		0.239
	G	0.931	H	1.006	J	1.016	K	0.955	L	0.969	X	2.763
		0.123		0.209		0.119		0.240		0.110		0.006
100	A	1.027	B	0.991	C	1.040	D	1.045	E	1.016	F	1.088
		0.248		0.022		0.122		0.038		0.149		0.307
	G	0.932	H	1.075	J	1.044	K	0.902	L	0.908	X	2.769
		0.022		0.148		0.298		0.005		0.070		0.004

Table A.2: Expected utility of right subtree of Figure 6.8 in section 6.3.3 for each instance, as well as mean and variance of utility achieved over 10,000 runs/instance.

Instance	Greedy	Res q-horzn	Class Tree	q-subst Disc.	Achvd μ	Achvd σ^2
1	0.5279	0.6323	0.6366	0.6389	0.6390	0.0255
2	0.5550	0.6173	0.6257	0.6289	0.6280	0.0099
3	0.5144	0.5537	0.5538	0.5551	0.5578	0.0109
4	0.5377	0.6093	0.6122	0.6131	0.6194	0.0161
5	0.5070	0.6337	0.6380	0.6395	0.6473	0.0303
6	0.5334	0.6585	0.6633	0.6651	0.6635	0.0369
7	0.5003	0.5768	0.5813	0.5857	0.5839	0.0082
8	0.5150	0.6496	0.6518	0.6517	0.6572	0.0195
9	0.5606	0.6603	0.6631	0.6656	0.6680	0.0156
10	0.5653	0.6382	0.6429	0.6445	0.6424	0.0272
11	0.5520	0.6222	0.6319	0.6339	0.6411	0.0167
12	0.5310	0.5948	0.5985	0.5989	0.6009	0.0138
13	0.5471	0.6135	0.6177	0.6194	0.6251	0.0236
14	0.5704	0.6429	0.6443	0.6439	0.6449	0.0252
15	0.5161	0.6231	0.6412	0.6419	0.6531	0.0220
16	0.5566	0.6246	0.6303	0.6326	0.6346	0.0205
17	0.6013	0.6590	0.6609	0.6603	0.6586	0.0156
18	0.5312	0.6554	0.6564	0.6571	0.6574	0.0204
19	0.5593	0.5969	0.6097	0.6095	0.6156	0.0075
20	0.5305	0.5958	0.6000	0.6027	0.5991	0.0141
21	0.4967	0.5643	0.5657	0.5652	0.5678	0.0105
22	0.5037	0.5961	0.6116	0.6137	0.6165	0.0175
23	0.5215	0.6220	0.6278	0.6301	0.6385	0.0242
24	0.5669	0.6716	0.6732	0.6738	0.6765	0.0242
25	0.5313	0.6652	0.6677	0.6657	0.6674	0.0313
26	0.4801	0.6538	0.6642	0.6665	0.6676	0.0385
27	0.5047	0.5858	0.5889	0.5919	0.5898	0.0191
28	0.5744	0.6449	0.6495	0.6490	0.6510	0.0160
29	0.5511	0.6651	0.6659	0.6695	0.6711	0.0174
30	0.5729	0.7055	0.7113	0.7126	0.7159	0.0214
31	0.5369	0.6256	0.6309	0.6311	0.6326	0.0203
32	0.5898	0.6869	0.6896	0.6927	0.6942	0.0189

Instance	Greedy	Res q-horzn	Class Tree	q-subst Disc.	Achvd μ	Achvd σ^2
33	0.5410	0.6273	0.6334	0.6327	0.6333	0.0151
34	0.5640	0.6235	0.6512	0.6562	0.6602	0.0175
35	0.5351	0.6148	0.6214	0.6225	0.6294	0.0151
36	0.5566	0.6917	0.6935	0.6945	0.6964	0.0233
37	0.5838	0.6647	0.6638	0.6634	0.6667	0.0120
38	0.4985	0.6554	0.6643	0.6679	0.6707	0.0321
39	0.5373	0.6531	0.6597	0.6620	0.6661	0.0210
40	0.5644	0.6287	0.6322	0.6323	0.6352	0.0130
41	0.5532	0.6491	0.6510	0.6516	0.6516	0.0204
42	0.5086	0.6297	0.6436	0.6492	0.6528	0.0183
43	0.5227	0.5949	0.6046	0.6082	0.6046	0.0113
44	0.5897	0.6424	0.6456	0.6447	0.6469	0.0205
45	0.5402	0.6909	0.7004	0.7018	0.7076	0.0339
46	0.5496	0.6752	0.6803	0.6830	0.6804	0.0220
47	0.4970	0.5969	0.6069	0.6104	0.6129	0.0202
48	0.5207	0.5833	0.5944	0.5963	0.5974	0.0120
49	0.6097	0.6475	0.6470	0.6463	0.6464	0.0182
50	0.5813	0.6528	0.6561	0.6592	0.6573	0.0121
51	0.5316	0.5908	0.5965	0.6006	0.5977	0.0182
52	0.6068	0.7147	0.7180	0.7197	0.7229	0.0235
53	0.5622	0.6331	0.6432	0.6464	0.6481	0.0209
54	0.6245	0.6933	0.6954	0.6983	0.7029	0.0234
55	0.5839	0.6562	0.6607	0.6622	0.6619	0.0139
56	0.5437	0.6656	0.6686	0.6688	0.6751	0.0278
57	0.5170	0.6171	0.6258	0.6300	0.6277	0.0266
58	0.5557	0.6585	0.6638	0.6666	0.6716	0.0176
59	0.5350	0.5773	0.6033	0.6072	0.6137	0.0239
60	0.5431	0.6530	0.6571	0.6578	0.6586	0.0261
61	0.5128	0.6204	0.6268	0.6279	0.6294	0.0276
62	0.6419	0.7187	0.7260	0.7279	0.7297	0.0127
63	0.5138	0.6409	0.6456	0.6486	0.6473	0.0240
64	0.5900	0.6781	0.6910	0.6948	0.7016	0.0169
65	0.5122	0.5741	0.5742	0.5761	0.5761	0.0143
66	0.5284	0.6079	0.6285	0.6265	0.6351	0.0164

Instance	Greedy	Res q-horzn	Class Tree	q-subst Disc.	Achvd μ	Achvd σ^2
67	0.5780	0.6784	0.6835	0.6868	0.6897	0.0177
68	0.5339	0.6495	0.6558	0.6582	0.6602	0.0149
69	0.5945	0.6810	0.6860	0.6867	0.6874	0.0218
70	0.5489	0.6377	0.6499	0.6520	0.6525	0.0184
71	0.5499	0.6666	0.6699	0.6669	0.6711	0.0187
72	0.5085	0.5994	0.6023	0.6036	0.6035	0.0214
73	0.5181	0.6295	0.6394	0.6428	0.6436	0.0217
74	0.5412	0.6003	0.6017	0.6023	0.6099	0.0189
75	0.5084	0.6384	0.6461	0.6479	0.6515	0.0193
76	0.5657	0.6827	0.6837	0.6825	0.6832	0.0295
77	0.5283	0.6773	0.6834	0.6867	0.6865	0.0236
78	0.5468	0.6488	0.6541	0.6509	0.6639	0.0180
79	0.5996	0.6547	0.6538	0.6548	0.6547	0.0117
80	0.5220	0.6031	0.6043	0.6055	0.6048	0.0129
81	0.5583	0.6302	0.6322	0.6331	0.6329	0.0194
82	0.5049	0.5872	0.5958	0.5962	0.6000	0.0180
83	0.5905	0.7107	0.7161	0.7176	0.7160	0.0200
84	0.5739	0.6454	0.6483	0.6495	0.6498	0.0313
85	0.4868	0.6029	0.6076	0.6120	0.6130	0.0176
86	0.6182	0.6606	0.6695	0.6730	0.6737	0.0086
87	0.5099	0.5867	0.5945	0.5956	0.5997	0.0106
88	0.5701	0.6872	0.6883	0.6883	0.6935	0.0179
89	0.5865	0.6336	0.6387	0.6413	0.6402	0.0142
90	0.6074	0.6670	0.6727	0.6740	0.6809	0.0101
91	0.5941	0.6659	0.6712	0.6710	0.6787	0.0119
92	0.5406	0.6469	0.6537	0.6549	0.6583	0.0203
93	0.5734	0.6555	0.6555	0.6538	0.6586	0.0178
94	0.5309	0.6419	0.6435	0.6445	0.6431	0.0190
95	0.5562	0.6330	0.6441	0.6475	0.6498	0.0110
96	0.5119	0.6168	0.6230	0.6237	0.6329	0.0154
97	0.4930	0.5988	0.5985	0.5987	0.6009	0.0187
98	0.5273	0.6117	0.6092	0.6129	0.6129	0.0159
99	0.5192	0.6122	0.6206	0.6215	0.6246	0.0181
100	0.5425	0.6111	0.6167	0.6168	0.6219	0.0096

Table A.3: Average utility achieved by each method in each instance over 10,000 runs/instance (as described in Section 6.3.4)

Instance	Greedy	Res q-horzn	Class Tree	q-subst Disc.
1	0.6356	0.6370	0.6390	0.6393
2	0.6230	0.6252	0.6285	0.6286
3	0.5544	0.5551	0.5553	0.5574
4	0.6112	0.6130	0.6176	0.6185
5	0.6366	0.6390	0.6450	0.6463
6	0.6617	0.6628	0.6641	0.6640
7	0.5813	0.5827	0.5843	0.5845
8	0.6507	0.6520	0.6565	0.6564
9	0.6629	0.6644	0.6666	0.6678
10	0.6414	0.6423	0.6432	0.6430
11	0.6281	0.6314	0.6396	0.6404
12	0.5968	0.5980	0.6007	0.6008
13	0.6164	0.6185	0.6233	0.6245
14	0.6434	0.6438	0.6449	0.6448
15	0.6326	0.6380	0.6519	0.6521
16	0.6286	0.6306	0.6341	0.6347
17	0.6596	0.6596	0.6586	0.6589
18	0.6563	0.6566	0.6572	0.6574
19	0.6031	0.6067	0.6152	0.6152
20	0.5991	0.6000	0.6005	0.5999
21	0.5647	0.5653	0.5678	0.5674
22	0.6047	0.6089	0.6165	0.6168
23	0.6261	0.6289	0.6357	0.6375
24	0.6727	0.6736	0.6757	0.6761
25	0.6654	0.6658	0.6674	0.6671
26	0.6602	0.6629	0.6678	0.6680
27	0.5888	0.5897	0.5905	0.5904
28	0.6469	0.6481	0.6510	0.6509
29	0.6673	0.6684	0.6690	0.6710
30	0.7091	0.7110	0.7153	0.7157
31	0.6283	0.6297	0.6325	0.6326
32	0.6898	0.6912	0.6931	0.6942

Instance	Greedy	Res q-horzn	Class Tree	q-subst Disc.
33	0.6300	0.6312	0.6334	0.6334
34	0.6398	0.6469	0.6604	0.6609
35	0.6186	0.6212	0.6280	0.6286
36	0.6931	0.6939	0.6956	0.6962
37	0.6640	0.6664	0.6652	0.6646
38	0.6616	0.6645	0.6700	0.6708
39	0.6575	0.6600	0.6650	0.6659
40	0.6305	0.6317	0.6348	0.6349
41	0.6504	0.6508	0.6516	0.6517
42	0.6395	0.6440	0.6519	0.6531
43	0.6015	0.6036	0.6060	0.6057
44	0.6435	0.6444	0.6469	0.6467
45	0.6964	0.6996	0.7067	0.7071
46	0.6791	0.6803	0.6814	0.6812
47	0.6037	0.6068	0.6124	0.6131
48	0.5897	0.5926	0.5976	0.5977
49	0.6469	0.6467	0.6469	0.6470
50	0.6560	0.6570	0.6579	0.6579
51	0.5958	0.5973	0.5988	0.5986
52	0.7173	0.7187	0.7217	0.7226
53	0.6397	0.6426	0.6479	0.6483
54	0.6957	0.6975	0.7000	0.7024
55	0.6592	0.6604	0.6621	0.6622
56	0.6672	0.6688	0.6741	0.6743
57	0.6235	0.6256	0.6286	0.6286
58	0.6625	0.6650	0.6697	0.6712
59	0.5926	0.5993	0.6133	0.6140
60	0.6554	0.6565	0.6586	0.6587
61	0.6241	0.6259	0.6293	0.6295
62	0.7233	0.7254	0.7295	0.7298
63	0.6448	0.6461	0.6477	0.6478
64	0.6865	0.6909	0.7000	0.7012
65	0.5751	0.5755	0.5755	0.5762
66	0.6170	0.6223	0.6349	0.6345

Instance	Greedy	Res q-horzn	Class Tree	q-subst Disc.
67	0.6826	0.6847	0.6884	0.6896
68	0.6539	0.6560	0.6597	0.6602
69	0.6838	0.6851	0.6875	0.6875
70	0.6448	0.6478	0.6529	0.6530
71	0.6667	0.6674	0.6711	0.6704
72	0.6015	0.6023	0.6036	0.6037
73	0.6361	0.6389	0.6436	0.6440
74	0.6013	0.6029	0.6074	0.6088
75	0.6431	0.6456	0.6508	0.6513
76	0.6826	0.6831	0.6832	0.6827
77	0.6820	0.6838	0.6866	0.6869
78	0.6499	0.6523	0.6639	0.6620
79	0.6547	0.6547	0.6547	0.6547
80	0.6043	0.6047	0.6050	0.6050
81	0.6316	0.6322	0.6330	0.6330
82	0.5917	0.5941	0.5996	0.5997
83	0.7141	0.7153	0.7166	0.7166
84	0.6475	0.6484	0.6498	0.6499
85	0.6074	0.6094	0.6121	0.6132
86	0.6668	0.6694	0.6737	0.6741
87	0.5912	0.5937	0.5991	0.5994
88	0.6877	0.6888	0.6928	0.6927
89	0.6374	0.6388	0.6406	0.6407
90	0.6705	0.6730	0.6794	0.6802
91	0.6685	0.6707	0.6778	0.6777
92	0.6510	0.6531	0.6578	0.6581
93	0.6546	0.6581	0.6581	0.6555
94	0.6432	0.6435	0.6436	0.6434
95	0.6402	0.6435	0.6495	0.6501
96	0.6202	0.6230	0.6312	0.6317
97	0.5987	0.6006	0.5987	0.5991
98	0.6123	0.6125	0.6123	0.6130
99	0.6169	0.6192	0.6243	0.6245
100	0.6139	0.6159	0.6213	0.6213

Vita

Scott Christopher Buffett

Universities Attended

1998 - Present	University of New Brunswick Candidate for Doctor of Philosophy
1996-1998	University of New Brunswick Master of Computer Science (May 1998)
1991-1996	University of New Brunswick Bachelor of Computer Science (May 1996)

Publications

S. Buffett and B. Spencer. A Decision Procedure for Bundle Purchasing with Incomplete Information on Future Prices. In the *International Journal of Electronic Commerce.* To appear.

S. Buffett, K. Jia, S. Liu, B. Spencer and F. Wang. Negotiating Exchanges of P3P-labeled Information for Compensation. In *Computational Intelligence*, 20(4), 2004. To appear.

S. Buffett and A. Grant. A Decision-Theoretic Algorithm for Bundle Purchasing in Multiple Open Ascending Price Auctions. In Proceedings of the *Seventeenth Canadian Conference on Artificial Intelligence (AI'2004)*, London, ON, Canada, 2004. To appear.

S. Buffett and B. Spencer. Efficient Monte Carlo Decision Tree Solution in Dynamic Purchasing Environments. In Proceedings of the *5th International Conference on Electronic Commerce (ICEC'03)*, Pittsburgh, PA, USA, pp. 31-39, 2003.

S. Buffett, K. Jia, S. Liu, B. Spencer and F. Wang. Negotiating Exchanges of P3P-labeled Information for Measurable Benefits. In Proceedings of the *2nd Business Agents and the Semantic Web (BASeWEB'03) workshop*, Halifax, NS, Canada, pp. 25-34, 2003.

S. Buffett and B. Spencer. Planning and Procurement in Multi-Agent Systems. In proceedings of the *Novel E-Commerce Applications of Agents workshop*, Ottawa, ON, Canada, pp. 29-36, June 8, 2001.

S. Buffett and B. Spencer. Reducing the Search Space Required in Implicit AND/OR Tree Solution Search. In Proceedings of the *APICS Mathematics, Statistics and Computer Science Conference*, St. Mary's University, Halifax, NS, Canada, Oct 1998.

S. Buffett. Investigating Iterative Deepening in Top Down Automated Reasoning. Master's Thesis, University of New Brunswick, 1998.