Interval Analysis for Unknown Dependencies and Genetic Algorithm Emulation of Markets

Market Interactions and Market Power
Final Project Report

Power Systems Engineering Research Center

A National Science Foundation
Industry/University Cooperative Research Center
since 1996

Power Systems Engineering Research Center

Interval Analysis for Unknown Dependencies and Genetic Algorithm Emulation of Markets

Market Interactions and Market Power
Final Project Report

Project Team

Gerald B. Sheblé, Project Leader
Daniel Berleant
Mei-Ping Chong
Jianzhong Zhang
Iowa State University

Robert J. Thomas
Cornell University

PSERC Publication 03-33

November 2003

Information about this Project

For information about this project contact:

Gerald B. Sheblé
Iowa State University
Department of Electrical and Computer Engineering
1115 Coover Hall
Ames, IA 50011
Phone: 515-294-3046
Fax: 515-294-4263
Email: gsheble@iastate.edu

Power Systems Engineering Research Center

This is a project report from the Power Systems Engineering Research Center (PSERC). PSERC is a multi-university Center conducting research on challenges facing a restructuring electric power industry and educating the next generation of power engineers. More information about PSERC can be found at the Center's website: http://pserc.org.

For additional information, contact:

Power Systems Engineering Research Center
Cornell University
428 Phillips Hall
Ithaca, New York 14853
Phone: 607-255-5601
Fax: 607-255-8871

Notice Concerning Copyright Material

Acknowledgements

The Power Systems Engineering Research Center (PSERC) sponsored the work described in this report. We express our appreciation for the support provided by PSERC's industrial members and by the National Science Foundation under grant NSF EEC-9908690 received under the Industry/University Cooperative Research Center program. Our thanks are also given to MidAmerican Energy for its support of this project.

Thanks are also given to our industry advisors:

- O. Dale Stevens, II, MidAmerican Energy Co.
- John Thomas Chatelain, MidAmerican Energy Co.

Preface

The Power Systems Engineering Research Center sponsored the research project titled "Market Interactions and Market Power (M-3)." This project had two parts:

- Part I: Interval Analysis for Unknown Dependencies (in Chapters 1-5)
- Part II: Adaptive Agent Market Emulation (in Chapters 6-8)

This report includes both parts.

Executive Summary

The project's objectives were to assess the interactions between the operational and commercial aspects of electricity markets; to identify key requirements in the reform of market structure and rules; and to propose market design modifications that will enable realization of the full the benefits of competition in electricity. Two parallel paths were pursued to achieve the objectives. The first path was to identify a technique suitable for using adaptive agents to value electricity bids. The second path was to emulate market interactions by adaptive agents in multiple markets. This emulation used a genetic-algorithm market simulator and a market simulator based on a genetic programming algorithm, including information from repeated bidding. The new techniques provide considerable insight into market behavior with repeated bidding. Based on simulation results, we identify recommendations for further use and development of the techniques; more definitive recommendations require additional work.

The first part of the report extends the applications of decision analysis to problems where the random variables may be dependent but the correlation is unknown. This part describes a new interval-based method to handle decisions under uncertainty when the two variables are of unknown dependency. Correlation is often confused with dependency. Correlation is only a measure of a special type of dependency. The technique developed enables decision analysis even if the dependency or correlation is unknown. The technique uses Linear Programming optimization to solve for the intervals given the nonlinear relationship between the two variables of unknown dependency. This work can be extended to more complex commodity pricing problems.

The second part of this report details the market simulations using adaptive agents. Initial market design recommendations are based on these results; however, more work needs to be done before final recommendations can be made. In particular, differences exist with the pricing rules and with the number of players. Additional testing is required, especially because the forward markets are a more critical element than previously understood.

The adaptive agent software was tested with a number of test sets and variations. The results provide interesting conclusions, even though the markets implemented were limited in scope. The results demonstrate the usefulness of the genetic algorithm to emulate market behavior at a level of complexity equal to and sometimes exceeding the market designs of experimental economics. Adaptive agent modeling and experimental economics can complement each other in assessing market design alternatives. More research is needed to compare and contrast the two analysis approaches for developing recommendations on market designs.

Another area for future research is enhancement of the adaptive agent with interval analysis methodology to provide a more robust decision model to play the market over many trials. Additional market rules, and complete emulation of the network physical capabilities and limitations should be included as they have been already. The next major step is to identify a means to model forward and ancillary markets without emulating all markets simultaneously.

Table of Contents

Table of Figures

Table of Tables

1 Decision-Making Under Uncertainty with Dependent Variables

A basic premise of competition is the ability of buyers and sellers to make rational decisions. The rational decision assumption requires each player to know the relative value of each resource needed, the relative value of each product produced, and the relative value of all services necessary to produce the product. The sellers have to get the product to the customer and maintain a competitive product portfolio. One major goal of deregulation is to identify the market design to provide the proper price signaling for valuation of assets and, thus, the proper value for bidding in the spot market, in the futures market, and in the expansion (or infrastructure planning) market of the electric energy system. The ideal is to project the incentives needed under market-based decision-making for market participants to invest in the careful design, operation and maintenance of the overall electric energy system. The premise is that asset valuation requires information of the principal components of future cash flows, ideally based on real option analysis by suppliers and buyers.

As the US moves toward competitive markets in electric power generation, considerable attention is focused on issues of ancillary service markets, and potential system security consequences of differences between contracted energy services and the physical supply of services. Such service requirements introduce uncertainties that must be included to determine the delivery possibilities. These issues relate primarily to the transportation capability of the system, on an hourly time scale. The proper valuation of the resources to produce and transport the electric energy has to include the impact of all ancillary services as well as of the primary fuel markets.

This project intended to achieve two goals. The first goal was the characterization of market interaction between primary energy exchange and ancillary service requirements, given market forecast uncertainty to provide improved estimates of bidding, and thus of asset valuation. The primary effort evaluated the effect of uncertainties in price bidding and matching in systems for settling contracts for energy delivery. Recent tools of probability and statistics theory (such as interval analysis, second order uncertainty modeling, and parametric programming) apply to such problems.

The second goal was to advance proposals of specific market designs and of their related market power philosophies, based on adaptive agent simulations that offer quantitative schemes for ensuring system expansion to enhance the system performance under a wide range of operating condition uncertainties. Standardized market interaction and baseline strategy to facilitate secure addition of competitive resources from all agents' units was assumed for all simulations. The benefits of properly assessing uncertainties, even when such uncertainties cannot be precisely determined, enable the proper pricing of the primary product, energy, and all supportive ancillary services.

Prototype designs for market models for primary energy delivery and for contingent services (such as transmission capability, load balance, regulation, spinning reserves, and ready reserves) for competitive adaptive agents were developed. Test-bed systems indicating

system benefits of these uncertainty evaluation tools when interacting with competitively driven bidding of energy dispatch and ancillary service provisions were developed.

The objective of previous research was to guarantee that a single player, or the contract to use any given asset, did not dominate market play. Clearly, contract behavior and market control challenges arise as large numbers of independent, for-profit companies trade contracts based on the latest equipment and system capability information. The project was to consolidate and extend these preliminary results, considering their application to more widespread supportive markets including ancillary services. This project did tailor auction and bidding model designs to a competitive power system environment, as well as examine results in trajectory sensitive markets as contracts changed based on new information.

This project was conducted in two distinct paths. The first path was the extension of techniques for decision-making under uncertainty for use by adaptive agents. The goal was to identify means to reduce the range of valuation for price signals for variables that are dependent on each other, but cannot be quantified precisely. As an example, the spark spread is the difference between the fuel price and the electric energy price. As the fuel price increases, the price of electric energy increases. This is often cited as a key indicator. The value of this indicator could then be found by estimating the correlation between the price of fuel and the price of electric energy. Previous studies have shown that this correlation varies between -0.4 and +0.9. This range of correlation shows that the spark spread is hard to forecast and is hard to include as a valuation factor for generation asset management. Such wide variations also demonstrate the lack of clear dependency, even when it is known to exist based on microeconomic analysis. Chapters 1 through 5 detail this work.

The second path, simulation of markets to demonstrate the price signaling for asset valuation, was the extension of simulation techniques from previous artificial life market simulators, market power assessment, and classical market simulation. The technique of handling uncertain intervals was to be included in these new agent models. Chapters 6 through 8 detail this work.

The decision-making under uncertainty technique called Interval-Based Distribution Analysis (IBDA) [Berleant and Goodman-Strauss, 1998] discretizes input distributions by using a list of numerical intervals that span the range of the distribution and, associated with each interval, the probability that a sample drawn from the distribution would fall within it. A relatively straightforward convolution process does a sum, product, or other operation on two inputs that are independent. In the case of unknown dependency, an optimization process calculates strategic points on the envelopes describing the outputs. This extension to IBDA, named Distribution Envelope Determination (DEnv), is both simpler and more flexible than the Probabilistic Arithmetic technique addressing the problem described by Williamson and Downs [1990]. It is also capable of incorporating information about Pearson correlation which copula-based techniques like Probabilistic Arithmetic do not do.

This report identifies the methods to implement this assessment and uses case studies used as examples. The concluding chapter identifies other applications.

1.1 Dependence and Time Series Analysis

Time series analysis, computationally reduced by functional dependence, yields more efficiently simulated risk models. Time series analysis extends the possibility of multiple period assessments instead of single period assessment. Cash flows over time for production or for services are combined to forecast representative project values for business valuations. Traditional project valuation uses net present value to determine if a project is a positive investment. Dependence is a characteristic between economic drivers to reduce the number of data series to analyze and develop as part of the process to build future cash flows.

Time series analysis enables the valuation of projects that provide a potential cash flow over long periods. The desired outcome of risk analysis is a stochastic assessment of the investment under analysis. Many business-setting analyses of the technical, political, or other economic drivers are typically based on probabilistic financial assessment of the investment opportunity. Financial analysis uses a net present value (NPV) technique, or alternatively internal rate of return or discounted return on investment. Such financial considerations require evaluation of the future cash flows over a long period, commonly a number of years up to over two decades.

Econometric analysis is the general study of series analysis of economic drivers. Economic drivers are the price signals that identify future opportunities, if any. Some economic drivers indicate the resource needs or costs for projects, such as labor, fuel, or other raw materials. Other economic drivers are the price signals indicating the value for products or services that buyers are assigning based on the application of the product or service to produce a subsequent product or service. The desired relationship for a risk assessment is a distribution of each variable coefficient.

The distribution for a variable, such as NPV, is found by combining input variables that are distributions. Several methods to generate distributions for single values exist. Typically, a separate distribution is established for each variable in each period. This method is more complex than representing time series as a stochastic function because distributions are generated for each period. The changing dependency of each variable between periods is another factor to model and to generate (that is, forecast). One method to reduce the number of parameters to forecast is to use a single basic distribution variable that changes in each future period. An example is to use the Normal distribution where the mean and the standard deviation change from one period to the next according to another functional calculation. A typical functional relationship would be for the mean to increase at 10% each period while the standard deviation increases by 5% each period.

Risk analysis generally terminates in a calculated variable that is represented by a distribution for each period into the future. Such a distribution can then be used by Monte-Carlo-based risk tools to generate scenarios for the calculation of the probable risk profile.

Once dependence between two variables is established, one variable is designated the independent variable and the other the dependent variable. Then, a Monte Carlo process first

randomly samples the independent variable. Next, depending on the degree of correlation between the two variables, the process establishes a dependent-variable value that is reasonable for the independent variable coefficient. This process is applied each iteration.

The distribution is sampled each iteration, across several periods within an iteration. If the distribution being sampled, period by period, has a dependent relationship with other variables, then those relationships need to be used across periods, not just across iterations.

1.1.1 Dependence, Correlation, and Valuation

Parameters that are co-contributors to a risk assessment may be independent. An example of two independent variables is the risk free interest rate and the product value for a given application. It is often valid to separate the international demand from the domestic demand due to societal or cultural differences. Such examples have no restrictions on the values of the variable based on the other variable. Specifically, there are valid variable pairs. However, many variables exhibit dependent relationships. The dependency of interest free rate on government budgetary deficits restricts the variable pairs that may be generated. A Monte Carlo analysis would have to generate values for each variable based on the dependency between the two variables. The concept of dependence is to model the relationship between two or more variables to generate valid future characteristics.

The first task is to identify the dependent and the independent variable. Specifically, the independent variable may be defined as the variable that may be considered as a driver for the other (dependent) variables. Alternatively, the independent variable may be defined as the variable that controls the dependent variables.

The economic interpretation of the variables may lead to a rational description of the dependence sequence for the given problem. When it cannot be easily discovered which is the independent (controlling) variable and which is the dependent variable, the independent variable is selected by the degree of confidence to generate a given variable or the ability of a variable to encapsulate the economic price signal.

The strength of the relationship is the next item to be identified. There are several techniques to estimate the strength of the relationships. One method is to use the degree of correlation between variables. Correlation is a description of how closely one variable follows another variable. Correlation is often used as an abbreviated reference to Pearson correlation, which varies from a negative relationship with a value of negative one (-1) to a positive relationship with a value of one (+1). Independence is indicated when the correlation is “close” to zero (0). A pair of variables is perfectly correlated if the correlation is positive or negative one. Regression techniques are often used to estimate the relationship strength. Linear and nonlinear regression techniques are generally available.

There are many pairs of variables in a risk assessment that may be considered dependent. It is often true that one variable can have a number of related dependent variables. It is important to define the dependent variable based on a truly independent variable to avoid circular dependency.

If there is a dependency between two variables, if we assign one variable as the independent variable and the other variable as the dependent variable, and if we find the correlation coefficient for this relationship, then the dependent variable will follow, in some partially characterized sense, the independent variable. A Monte Carlo analysis will first generate the independent variable by a random selection, and then generate a value for the dependent variable by random selection from the distribution determined in part by the value of the first variable.

If the risk assessment model is to value a bid or a project properly (i.e., emulate the real world), then the relationships indicated by economic laws and physical processes must be consistent with the variable chosen to be dependent and the variables chosen to be independent.

1.1.2 Decision Theory, Trees, and Analysis

The basic decision process is often described by a tree structure as shown in Figure 1-1.

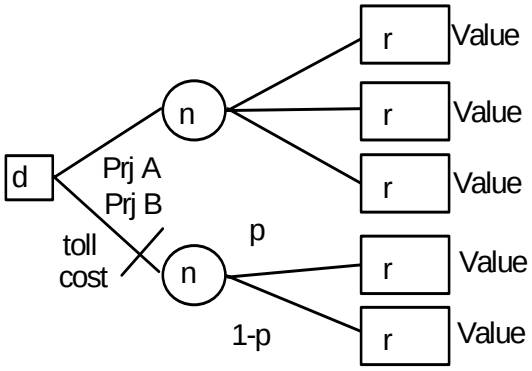

Figure 1-1 Decision Tree Format.

Each decision node is represented by a square. Natural nodes are represented by a circle. Result node (leafs) are represented by rectangles. The cost of any action on a given path is represented by a line (tollgate) perpendicular to the path. The probability of any natural node is adjacent to the branch connecting the natural node with another node in the tree. The tree is often evaluated using:

$$EMV = \sum_{i=1}^{n} (p_i * V_i)$$

$$\sum_{i=1}^{n} p_i = 1.0$$

where Expected Monetary Value is EMV, probability of event is p_i, value (benefit or cost) after event is V_i, and number of events is n. The decision rule is to take the option with the highest payoff (maximum profit) or lowest cost.

There are six basic steps used to define the basic decision tree process:

- Generate Decision Tree
- Define Decision Criterion
- Determine Value of Each Path
- Assess Probabilities, Correlations
- Complete Mathematical Evaluation
- Assess Solution Sensitivity

The key is to state the problem to draw the tree with each type of node identified (Decision, Natural, Result), to find all joint or independent probabilities, to calculate the total cost and benefit for the given conditions for each independent variable, and to include all uncertainties.

As an example, we consider the case where there are two unknown economic drivers. Each economic driver is a random variable. The problem with this approach (as addressed by this research) arises from the complication when one random variable is dependent on another one random variable. In Figure 1-2, each random variable is the status of a generating unit. The node in the first column represents the availability of unit 1, the node in the second column represents unit 2, etc. Traditional probabilistic production costing assumes that the availability of units is independent. Thus, the sequence of nodes can be arbitrarily exchanged. However, when the units have common characteristics (such as coal supply, water supply, common labor crew, or common maintenance cycles), the availabilities are dependent. The output of one unit is dependent on a common variable not represented in the tree calculations. Indeed, such a common variable that the units are dependent upon may not be included in the calculations as stated in the tree or available for measurement.

Classical decision analysis assumes that the correlation is known. This research was directed to the relaxation of this methodology constraint.

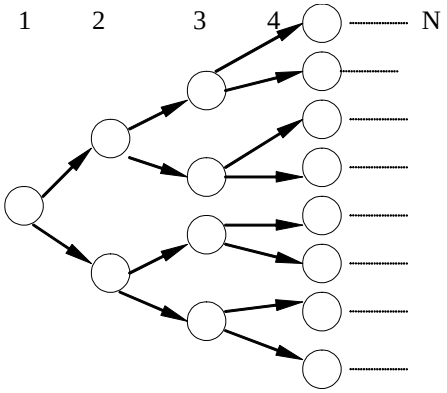

Figure 1-2 Unit Availability Tree.

Another example of unknown dependency is shown in the following game tree. The single agent decision process can be described in a game theory context [Aliprantis and Chakrabarti, 1998]. Generally, a *decision graph* is employed to describe the sequential decision process of a single person. A decision graph is directed graph having a unique root *R*, in the sense that *R* is the only node with no edge ending into it. There is at least one path from *R* to *N* for every node *N* other than *R*. There is at least one terminal node. There is at least one path from *N* to a terminal node from every non-terminal node *N*. This is illustrated in Figure 1-3.

When X is a decision node, all other nodes are natural nodes (or consequence), Hi indicates High Investment, Lo indicates Low Investment, M indicates "Market" decision, DM indicates the decision to "Do not Market," p is the probability of success under high investment, and q is the probability of success under low investment.

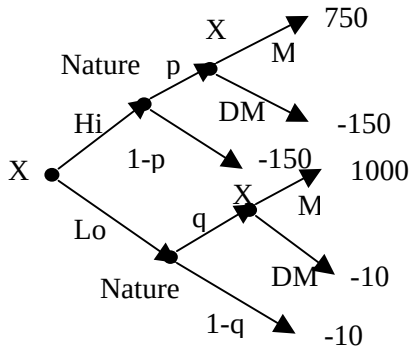

Figure 1-3 Game Response Tree.

Thus, basic decision tree process and game theory process modeling share the assumption that the correlation between variables is known. Often, the correlation is assumed to be zero. Based on economic theory, this is known to be a false assumption when the complete

industrial or decision/consequence model is evaluated. Thus, the need for a less restrictive assessment is needed.

1.2 Adaptive Agent Emulation of Energy Markets

Adaptive agents use structures similar to the decision trees to emulate the analysis performed by the trading staff of generation companies (GENCOs) and energy management companies (EMCOs). The problem with present decision trees is insufficient representations of the trade analysis commonly performed. However, this simplicity demonstrates the behavior of trading agents that can be understood from basic economic principles. The tree complexity is enhanced when the dependencies do not have to be known, as provided by the interval analysis methodology. More discussion of this enhancement is in the section on developing the genetic algorithm machines.

1.3 Intervals, Uncertainty, and Distributions

Uncertainty is frequently present in our knowledge of the real world. Handling uncertainty is therefore an important problem. Probability is a common approach. Probability density functions (PDFs) or their integrals, and cumulative distribution functions (CDFs) are used in a probabilistic approach. These are used to describe random variables. Often such random variables are defined based on combinations of other random variables. These are called derived random variables, and their distributions are called derived distributions [Springer, 1979]. For example, samples of a derived random variable could be defined as the sum, the max, or some other function of samples of two other random variables. Derived random variables are recognized in such fields as decision analysis and risk analysis.

A variety of methods have been developed to address this topic. There are two classes of such methods: analytical and numerical. Analytical methods are restricted to specific classes of input distribution, and usually make simplifying assumptions, such as independence. For example, normal distributions are often used. If two random variables are normal and independent, the sum of these two random variables still is normal. It is also possible to obtain derived distributions for specified dependency relationships other than independence, such as perfect positive rank correlation. However, it is often not easy to obtain analytical results for arithmetic operations on random variables without assumptions and it is not always reasonable to make these assumptions though it may be convenient. Sometimes, we do not have information about the dependency. Nevertheless, an advantage of analytical methods is accuracy. Unlike analytical methods, numerical methods only give numerical results. However, this is suitable for a wide class of distributions. Numerical methods are widely used in real applications if approximate results within specific tolerances are acceptable.

Monte Carlo simulation is one of the best-known numerical methods. However, the traditional approach of Monte Carlo has some limitations. It assumes that the distribution of the random variables is known, and that their dependency relationship is independent or known [Ferson, 1996]. If either the probability distributions or the dependency relationship

of the random variables is not available, some assumptions are usually necessary to process it. If the assumptions do not hold in reality, dependability of results can be degraded.

A discrete convolution approach may be used to calculate the result given independence of the input random variables [Ingram et al., 1968; Colombo and Jaarsma, 1980; Kaplan, 1981]. Interval analysis may be used to solve this problem while providing error bounds. (Intervals become closer to point values, as the intervals get narrower.) Interval mathematics became an identifiable field in the 1960's [Moore, 1966].

Intervals have the potential for bounding the result of an operation. Discretization error coming from discretizing distributions may be bounded by interval-based discretization [Berleant, 1993]. If the dependency is not specified, result bounds will include the entire range of possible dependencies. These bounds will normally be wider than if a particular dependency is specified. The Distribution Envelope Determination (DEnv, also referred to as "IBDA") technique for computing bounds on derived distributions without specifying dependency was described by Berleant and Goodman-Strauss [1998]. This approach has fundamental similarities with the copula-based approach [Frank et al., 1987] which was significantly extended by Williamson and Downs [1990]. These two methods have been implemented in software. The copula-based approach, termed probabilistic arithmetic, is implemented in the commercial software RiskCalc [Ferson et al., 1998]. DEnv is implemented as Statool [Berleant and Goodman-Strauss, 1998] which extends the previous tool [Berleant and Cheng, 1998] by eliminating the independence assumption. DEnv (and Statool) thus can handle the case where a dependency relationship is unknown or unspecified by not making assumptions about the dependency relationship between operands. Yet, partial dependence information might be available in some cases. If we can use this information in the calculation, we will get more accurate results than can be obtained without using this information.

DEnv and its implementation support a variety of dependency relationships, such as independence, unknown dependence, and correlation values. The algorithm extension to support correlation is a significant improvement. In the implementation, we have found that using the transportation simplex method speeds up computing significantly over the classical simplex method. Recent advances in the algorithm now allow cascaded operations and monotonic binary functions to be supported. These new functions, and use of correlation as a constraint, are the main recent advances in DEnv and its implementation. Among the other contributions reported here are addressing example application problems. Recently we have shown, using bounds on derived distributions in a GENCO competitive bidding scenario, that:

- the unjustified assumption of independence between uncertain quantities has a partially quantifiable dollar cost,
- the information that uncertain quantities are independent, if true, has a quantifiable dollar value, and
- a decision tree approach can identify optimal bids even when the dependency relationship between uncertain quantities is unknown.

2 Uncertainty Intervals

Uncertainty exists frequently in our knowledge of the real world. Probability is a common way to understand uncertainty and is intrinsic to the concept of the random variable. Intuitively, a random variable is a function that takes as input an experiment and returns a value that expresses the result of the experiment. Thus, if the experiment is to measure a voltage, the value returned is the number of volts. Typically, the result of the experiment cannot be determined in advance. A probability distribution function is often used to describe the way the experiment is more likely to return some values more often than others. Sometimes random variables are defined such that samples of their distributions are derived from arithmetic operations on samples of the distributions of other random variables. Determining the distribution of these derived random variables has been the goal of a considerable body of work.

As noted in the previous section, numerical analysis using the Monte Carlo technique has some drawbacks. Major ones include undependable conclusions about unusual situations, however important those situations may be, because in a simulation run, random number generation may not generate such situations at the same rate that they would actually be expected to occur, or might even fail to generate any example of such a situation at all.

To alleviate the problems with Monte Carlo analysis, one method that has been described is Distribution Envelope Determination (DEnv), developed by Berleant and Goodman-Strauss. Another approach is the copula-based approach called Probabilistic Arithmetic. These two methods have been implemented in software.

Interval mathematics has become a well-developed, sophisticated area since it became identifiable as an area in the 1960's. Influential reviews, fairly comprehensive at the time of writing, include for example [Alefeld and Herzberger, 1983]. Such works provide important basics and truths that retain their relevance today.

An interval can be used to bound the range for a value. Arithmetic operations on intervals have been defined in the literature. For example, if interval X is the interval [1, 2], and interval Y is the interval [3, 4], then interval $Z=X+Y$ is the interval [1+3, 2+4] = [4, 6]. Some additional explanations of interval properties are provided in the Appendix. One use for intervals in computations on random variable distributions is to partition the domain of the distribution into a set of intervals, with a probability associated with each. This partitioning is the basis for discretizing distributions and extending binary operations from intervals to (discretized) distributions.

Given two random variables X and Y, to get the exact distribution for a function of samples of X and Y, we must know the joint distribution of X and Y. The joint distribution is constrained (though not determined) by the correlation for these two random variables. Consider an example. The following table gives the interval-based discretization for two distributions, one for random variable X and one for Y.

Table 2.1 Discretized distributions for X and Y.

	X			Y		
Range	[1,2]	[2,3]	[3,4]	[2,3]	[3,4]	[4,5]
Probability	0.25	0.5	0.25	0.5	0.3	0.2

Sharing of interval endpoints between adjacent intervals in a discretization is of no practical consequence unless distributions have impulses at those shared endpoints. In that case, discretizations would need to contain partially open intervals, e.g., [1, 2]. Discretizations contain no information about distributions of probabilities over their corresponding intervals. However, each interval does limit its probability to within its endpoints. We also do not know the dependency relationship between X and Y. In other words, we do not know the joint distribution for X and Y.

We consider the addition $Z = X + Y$. Because we do not have the joint distribution for X and Y, it is impossible to find the precise distribution for Z. However, we can put these two random variables into a "joint distribution tableau," as shown in the following table.

Table 2.2 Joint distribution tableau showing marginal distributions for X and Y.

$z \in [3,5]$ $p_{11} = ?$	$z \in [4,6]$ $p_{12} = ?$	$z \in [5,7]$ $p_{13} = ?$	$y \in [2,3]$ $p_{Y1} = 0.5$
$z \in [4,6]$ $p_{21} = ?$	$z \in [5,7]$ $p_{22} = ?$	$z \in [6,8]$ $p_{23} = ?$	$y \in [3,4]$ $p_{Y2} = 0.3$
$z \in [5,7]$ $p_{31} = ?$	$z \in [6,8]$ $p_{32} = ?$	$z \in [7,9]$ $p_{33} = ?$	$y \in [4,5]$ $p_{Y3} = 0.2$
$x \in [1,2]$ $p_{X1} = 0.25$	$x \in [2,3]$ $p_{X2} = 0.5$	$x \in [3,4]$ $p_{X3} = 0.25$	$\leftrightarrow$ $\updownarrow$ X Y

The last row in the table is the distribution for X and last column is the distribution for Y. We do not know the values of probabilities $p11$ through $p33$ because we do not know the joint distribution. For the simple case of X and Y independent, we can fill in the missing values as in the following table.

Table 2.3 Joint distribution for independence.

$z \in [3,5]$	$z \in [4,6]$	$z \in [5,7]$	$y \in [2,3]$
$p_{11} = 0.125$	$p_{12} = 0.25$	$p_{13} = 0.125$	$p_{Y1} = 0.5$
$z \in [4,6]$	$z \in [5,7]$	$z \in [6,8]$	$y \in [3,4]$
$p_{21} = 0.075$	$p_{22} = 0.15$	$p_{23} = 0.075$	$p_{Y2} = 0.3$
$z \in [5,7]$	$z \in [6,8]$	$z \in [7,9]$	$y \in [4,5]$
$p_{31} = 0.05$	$p_{32} = 0.1$	$p_{33} = 0.05$	$p_{Y3} = 0.2$
$x \in [1,2]$	$x \in [2,3]$	$x \in [3,4]$	$\leftrightarrow$ $\updownarrow$
$p_{X1} = 0.25$	$p_{X2} = 0.5$	$p_{X3} = 0.25$	X Y

Because the joint distribution is affected by the dependency relationship between *X* and *Y*, if we do not know the dependency relationship between *X* and *Y*, we cannot determine the joint distribution in this tableau. Nevertheless, we can infer some things about the result or dependent variable *Z=X+Y* from this matrix. For example, consider *z* = 5. This is possible only in the grey cells in the next table.

Table 2.4 Graying indicates cells in which *z* might equal five.

$z \in [3,5]$	$z \in [4,6]$	$z \in [5,7]$	$y \in [2,3]$
$p_{11} = ?$	$p_{12} = ?$	$p_{13} = ?$	$p_{Y1} = 0.5$
$z \in [4,6]$	$z \in [5,7]$	$z \in [6,8]$	$y \in [3,4]$
$p_{21} = ?$	$p_{22} = ?$	$p_{23} = ?$	$p_{Y2} = 0.3$
$z \in [5,7]$	$z \in [6,8]$	$z \in [7,9]$	$y \in [4,5]$
$p_{31} = ?$	$p_{32} = ?$	$p_{33} = ?$	$p_{Y3} = 0.2$
$x \in [1,2]$	$x \in [2,3]$	$x \in [3,4]$	$\leftrightarrow$ $\updownarrow$
$p_{X1} = 0.25$	$p_{X2} = 0.5$	$p_{X3} = 0.25$	X Y

As previous stated, we do not know the exact probability for *z* <= 5. However, we can ask what the possible probabilities are for *z*<=5. As this matrix shows, only grey cells contribute to the probability of *z*<=5. We would like to determine the maximum probability and the minimum probability. To get the maximum value, all cells in which *z* can be <= 5 will have their probabilities summed. To obtain the minimum value, only cells in which Z must be <=

5 will have their probabilities summed. For example, we consider cell p_{12}. When we calculate the maximum value for z<=5, this cell is counted because z can be <= 5 in this cell. Nevertheless, for the minimum value, we do not count this cell because z might not be <= 5 in this cell. This way, we can find the possible range of cumulative probabilities for various values of z, a sample of random variable Z. We can find the maximum probability and minimum probability for every value of z, and connect all these points to get 2 curves: a left curve and a right curve. All the CDFs that are possible for Z must belong between these two curves.

In this example, suppose Z's range is from 3 to 9. It is clear that the probability for z<3 is 0 and for z>9 is 1. For example, consider the probability of $z \leq 4$.

- Maximum. First, find all the cells in which this situation may occur. From the previous table, these cells are p_{11}, p_{12}, and p_{21}. So the maximum value is the maximum value for the sum of p_{11}, p_{12}, and p_{21}.
- Minimum. First, find all the cells in which z must be <= 4. In this table, there are none. Although p_{11}, p_{12}, and p_{21} may satisfy $z \leq 4$, they also might not. For example, the whole probability for the cell might be concentrated at the high bound of its range. So there is no cell in which z must be <= 4.

Summarizing the above analysis, we can define a way to tell which cells contribute to the maximum and minimum probability values. The maximum probability is found from all the cells in which the low bound is not greater than the value of z contribute to the max value. The minimum probability is found from all the cells in which the high bound is not greater than the value of z contribute to the min value.

After finding all the cells satisfying the max (or min) condition, we must calculate the sum of the probabilities of these cells. Based on the previous table, there exist constraints for the probabilities p_{ij}. It is clear that the sum of the p_{ij}'s in a row or column cannot go over the marginal probability of that row or column. These constraints can be described as follows.

Row Constraints: $\sum_{j=1}^{3} p_{ij} = p_{Yi}$ for i=1 to 3.

Column Constraints: $\sum_{i=1}^{3} p_{ij} = p_{Xj}$ for j=1 to 3.

Therefore, the question becomes how to find the maximum and minimum value for the sum of cells under these constraints. For the case $z \leq 4$, we can describe these questions as follows.

Maximum - make the sum of the specified cells' value big enough, that is, find

max ($p_{11} + p_{12} + p_{21}$) such that:

$\sum_{j=1}^{3} p_{ij} = p_{Yi}$ for i=1 to 3

and $\sum_{i=1}^{3} p_{ij} = p_{Xj}$ for j=1 to 3.

Minimum - make the sum of specified cells' values small enough, that is, find

min ($\sum_{i=1}^{3}\sum_{j=1}^{3} p_{ij}$) such that:

$\sum_{j=1}^{3} p_{ij} = p_{Yi}$ for i=1 to 3

and $\sum_{i=1}^{3} p_{ij} = p_{Xj}$ for j=1 to 3

For these two optimization questions, linear programming is a suitable tool to. Thus, we can find the probability range for the specified value of *z*. Table 2-5 shows the probabilities for various values of *z*.

From this table, we can draw two curves, a top curve and a bottom curve, using the maximum and minimum probabilities shown for various values of *z*. These two curves also can be called envelopes for the CDF of derived random variable *Z* because the CDF for *Z* must be between these two curves whatever the dependency relationship between *X* and *Y* is. Figure 2-1 shows the result.

Table 2.5 Probabilities for result variable *z*.

z range	Maximum probability	Minimum probability
z<3	0	0
3<z<=4	0.25	0
4<z<=5	0.75	0
5<z<=6	1	0
6<z<=7	1	0.25
7<z<=8	1	0.55
8<z<=9	1	0.8
z>9	1	1

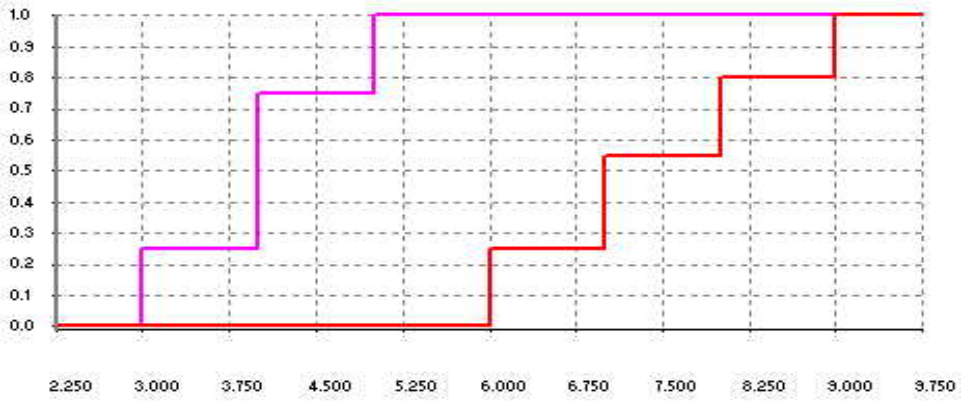

Figure 2-1 Probability Bounds for Random Variable *Z*.

3 Narrowing the Envelopes Around Results Using Correlation

The previous chapter discussed the core DEnv algorithm. This algorithm uses linear programming based on the row and column constraints on the p_{ij}'s. The previous chapter also mentioned an important factor: correlation. If one knows something about correlation, it would be good to be able to use it to narrow the separation of envelopes. We have recently identified how to convert information about the correlation into constraints that can supplement the row and column constraints of the core algorithm, resulting in many cases in narrowing of the space between the left and right envelopes [Berleant and Zhang (in press)]. We describe this new augmentation to the algorithm in this chapter.

3.1 Facts about Correlation

Correlation measures the degree of correspondence between random variables. To describe this kind of relationship, there are a number of methods. For example, we can consider many possible relationships, such as a linear relationship between two random variables, a linear relationship of the squares of the random variables, among other possibilities. By far the most popular correlation coefficient is called Pearson correlation, or Pearson product-moment correlation. It measures the strength of the linear relationship between two random variables. It is defined as

$$\rho = \frac{E\left[\left(X - E(X)\right)\left(Y - E(Y)\right)\right]}{\sqrt{D(X)D(Y)}}$$

were $D(X)$ is the variance of X and $D(Y)$ is the variance of Y. E is the expectation function.

The Pearson correlation has the potential range $-1 \le \rho \le 1$. Correlation values can be classified into three types: (1) positive correlation, meaning there is a positive direct linear correlation between the random variables; (2) negative correlation, meaning there is an inverse linear correlation between the random variables; and (3) zero correlation, meaning there is no apparent linear relationship between the random variables.

3.2 Joint Distributions

A joint distribution describes the detailed dependency between two random variables. From the joint distribution, we can get the correlation. However, correlation does not imply a specific joint distribution, so in general we cannot get the joint distribution from a value of the correlation.

3.2.1 Interval-Valued Correlations

When the correlation is unknown, we use linear programming to find CDF envelopes, as described in the previous chapter. If we know the correlation between two operands, we would like to use that information to determine additional constraints for the linear

programming problem. In another words, we wish to decrease the feasible solution space and get a better solution.

According to the definition of correlation, for samples *x* and *y* of two random variables *X* and *Y*, the correlation is

$$\rho = \frac{E[(x - Ex)(y - Ey)]}{D(x)D(y)} = \frac{E[(x - Ex)(y - Ey)]}{\sqrt{E[(x - Ex)^2]}\sqrt{E[y - E(y)^2]}}$$

where Ex and Ey are the means for *x* and *y*.

Using the following formulas, we can simplify terms:

$$E[(x - Ex)(y - Ey)] = E[xy - yEx - xEy + Ex * Ey]$$
$$= Exy - Ey * Ex - Ex * Ey + Ex * Ey = Exy - Ex * Ey$$

where *Exy* is the expectation of *x***y*. Also,

$$E[(x - Ex)^2] = E[x^2 - 2xEx + Ex * Ex] = Ex^2 - 2Ex * Ex + Ex * Ex$$
$$= Ex^2 - (Ex)^2$$

so the previous equation becomes

$$\rho = \frac{E[(x - Ex)(y - Ey)]}{\sqrt{E[(x - Ex)^2]}\sqrt{E[y - E(y)^2]}} = \frac{Exy - Ex * Ey}{\sqrt{(Ex^2 - (Ex)^2)(Ey^2 - (Ey)^2)}}$$

Using the definition of mean, when variable *x* is discrete,

$Ex = \sum_i x_i * p_i$ where $p_i = p(x = x_i)$.

When *x* is continuously distributed, and has density function *f*, then

$Ex = \int xf(x)dx$.

In the DEnv algorithm, we do not care if a random variable is discrete or continuous. We use bars to discretize the distribution. This method has the following characteristics:

- Bars may overlap (this is a generalization from the previous chapter).
- Histograms are a special case of collections of bars.
- A bar describes the probability of an interval containing the value of a variable.
- No assumption is made about the distribution of the probability associated with a bar over the interval of the bar.

We now extend the definition of mean to intervals. $E(X) = \sum_i X_i * P_i$ where $E(X)$ is the (interval-valued) mean of a collection of intervals $X=\{X_1,...X_m\}$, and $P_i(x \in X_i) = P_i$. The mean Ex is in $E(X)$.

When x is continuous, we also can get the mean based on the following argument.

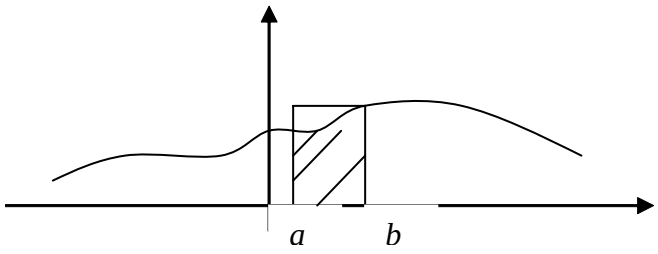

Figure 3-1 Probability Bounds for Random Variable *Z*.

We consider some bar in the discretization of variable x whose distribution function is $f(x)$. The probability that it is in $[a,b]$ is the area of $f(x)$ between a and b.

$$P(a \le x \le b) = \int_a^b f(x)dx$$

We can partition the domain of variable x into many intervals such as this one, denoting them X_i. The mean of variable x now becomes

$$Ex = \int xf(x)dx = \sum_i \int_{Xi} xf(x)dx$$

Consider one item in the previous formula, assuming X_i is $[a,b]$ as in the previous figure.

$$\int_{X_i} xf(x)dx = \int_{[a,b]} xf(x)dx \ge \int_{[a,b]} af(x)dx = a * \int_{[a,b]} f(x)dx = a * p_i .$$

Similarly, we also get

$$\int_{X_i} xf(x)dx = \int_{[a,b]} xf(x)dx \le \int_{[a,b]} bf(x)dx = b * \int_{[a,b]} f(x)dx = b * p_i .$$

Therefore, $\int_{X_i} xf(x)dx$ must belong to X_i*p_i. Thus, $E(x)$ is in the mean of interval variable X.

If the intervals overlap, the width of the mean is wider than for the non-overlapped case. Thus, the mean of the non-overlapped intervals is a subset of that of the overlapped intervals. Hence, $E(x)$ belongs to mean of interval variable X in this case, too.

3.2.2 Legal and Illegal Correlation Values

In the software tool developed in this project, the user can input any value of correlation from –1 to 1. However, for some marginal distributions, there are correlation values that will not be exhibited by any joint distribution. In fact, the constraints derived from setting the correlation to an impossible value should conflict with the constraints derived from the marginals of the joint distribution matrix (i.e., the row and column constraints).

From the definition of correlation, and algebraic rearrangement, we obtain this equation for $E(xy)$: $Exy = Ex * Ey + \rho\sqrt{(Ex^2 - (Ex)^2)(Ey^2 - (Ey)^2)}$. Let $f(x,y)=E(xy)$, so $f(x,y)$ is a real function of x and y. We can rewrite $E(xy)$ with intervals X and Y.

$$f(X,Y) = EX * EY + \rho\sqrt{(EX^2 - (EX)^2)(EY^2 - (EY)^2)}$$
$$= \sum_i X_i P_{xi} \sum_j Y_j P_{yj} + \rho\sqrt{(\sum X_i^2 P_{xi} - (\sum_i X_i P_{xi})^2) * (\sum Y_j^2 P_{yj} - (\sum_j Y_j P_{yj})^2)} \quad .$$

This is the interval extension of the corresponding real function

$$f(x,y) = \sum_i x_i p_{xi} \sum_j y_j p_{yj} + \rho\sqrt{(\sum x_i^2 p_{xi} - (\sum_i x_i p_{xi})^2) * (\sum y_j^2 p_{yj} - (\sum_j y_j p_{yj})^2)}$$

where $x_i \in X_i$ and $y_j \in Y_j$. If ρ is an interval, it becomes another variable in function f.

3.2.3 Solution

The implemented software tool should provide a way to help the user to set a reasonable correlation. To do this, first, the software must figure out the range of possible correlations for the current random variables. Then, the software can display this information. It then only accepts values intersecting with this range.

As mentioned before, there are two kinds of constraints, one coming from the marginals of the joint distributions matrix and another coming from the correlation setting. The joint distribution matrix marginals are assumed correct. Therefore, the constraints arising from them are a given. If constraints coming from a correlation setting conflict with them, they must be in error. Constraints coming from the matrix are primary and constraints coming from correlation should be considered secondary.

Consider a joint distribution matrix for an operation $\otimes$.

Table 3.1 Joint distribution matrix.

	Y1	…	*Ym*	
X1	p_{11}	…	p_{1m}	p_{x1}
…	…	…	…	…
Xn	p_{n1}		p_{nm}	p_{xn}
	p_{y1}	…	p_{ym}	

We can get $E(xy)$ as follows:

$Exy \in \sum_{i=1}^{n}\sum_{j=1}^{m} X_i Y_j p_{ij}$ where X_i and Y_j are interval values.

Probability p_{ij} is the probability assigned to cell *ij*. We use underlining to indicate the low bound of an interval and overlining to indicate the high bound of an interval. We can get the bounds of $E(xy)$ as follows:

$$\underline{E(xy)} = \underline{x_1 y_1} p_{11} + \underline{x_1 y_2} p_{12} + \underline{x_1 y_3} p_{13} + ... + \underline{x_2 y_1} p_{21} + \underline{x_2 y_2} p_{22} + \underline{x_2 y_3} p_{23} + ... + \underline{x_n y_m} p_{nm}$$

$$\overline{E(xy)} = \overline{x_1 y_1} p_{11} + \overline{x_1 y_2} p_{12} + \overline{x_1 y_3} p_{13} + ... + \overline{x_2 y_1} p_{21} + \overline{x_2 y_2} p_{22} + \overline{x_2 y_3} p_{23} + ... + \overline{x_n y_m} p_{nm}$$

From this, we get two linear programming problems:

$$Min\underline{E(xy)} = \underline{x_1 y_1} p_{11} + \underline{x_1 y_2} p_{12} + \underline{x_1 y_3} p_{13} + ... + \underline{x_2 y_1} p_{21} + \underline{x_2 y_2} p_{22} + \underline{x_2 y_3} p_{23} + ... + \underline{x_n y_m} p_{nm}$$

subject to:

row constraints: $\sum_{j=1}^{m} p_{ij} = p_{xi}$ for i=1 to n;

column constraints: $\sum_{i=1}^{n} p_{ij} = p_{yj}$ for j=1 to m.

$$Max\overline{E(xy)} = \overline{x_1 y_1} p_{11} + \overline{x_1 y_2} p_{12} + \overline{x_1 y_3} p_{13} + ... + \overline{x_2 y_1} p_{21} + \overline{x_2 y_2} p_{22} + \overline{x_2 y_3} p_{23} + ... + \overline{x_n y_m} p_{nm}$$

subject to:

row constraints: $\sum_{j=1}^{m} p_{ij} = p_{xi}$ for i=1 to n;

column constraints: $\sum_{i=1}^{n} p_{ij} = p_{yj}$ for j=1 to m.

Solving these two linear programming problems, we can get the bounds of $E(xy)$. Call these numbers $\underline{k}$ and $\overline{k}$. We also know that

$$E(xy) = \sum_i x_i p_{xi} \sum_j y_j p_{yj} + \rho \sqrt{(\sum x_i^2 p_{xi} - (\sum_i x_i p_{xi})^2)(\sum y_j^2 p_{yj} - (\sum_j y_j p_{yj})^2)}$$

where $x_i \in X_i$ and $y_j \in Y_j$.

In this equation, only ρ is an unknown range. Now the problem becomes solving for $E(xy)$. The minimum should be the minimum value of ρ. The maximum should be the maximum value of ρ. Thus, the problem is transformed into finding the root range of a nonlinear function.

3.2.4 Approximate Solution

From $f(x,y)= \bar{x} \cdot \bar{y} + \rho\sqrt{(\overline{x^2} - \bar{x}^2)(\overline{y^2} - \bar{y}^2)}$, in most cases, $\bar{x} \cdot \bar{y}$ is greater than $\sqrt{(\overline{x^2} - \bar{x}^2)(\overline{y^2} - \bar{y}^2)}$. Therefore, we just consider $\bar{x} \cdot \bar{y}$. It is obvious that it is an increasing function of x and y. Assigning the minimum values to x and y, and the maximum value possible for $f(x,y)$, we can obtain the maximum value of ρ. Assigning the maximum values to x and y, and the minimum value to $f(x,y)$, we can get minimum value of ρ.

3.2.5 Additional Constraints from Correlation

When the user sets the correlation range, we know the range of every variable in equation f(x,y). Under this situation, we can get the range of f(x,y). This range of f(x,y) is thus controlled by the user. At the same time, we know another range for f(x,y) which is derived from the joint distribution matrix. As previous noted, the range derived from the joint distribution matrix is considered given. Therefore, it is always correct. The range coming from the user must be intersected with this range. From this restriction, we can get additional constraints for linear programming.

Since formula $f(x,y)$ is non-linear, we use non-linear optimization to do minimization and maximization on it. Using a penalty function transforms a constrained optimization problem to a non-constrained problem. We also can get the first and second derivative for this function. Call the values obtained *fmin* and *fmax*. Thus, we get $f(x,y)$=[*fmin*,*fmax*].

In the previous section we obtained another range for $f(x,y)$, namely $[\underline{k}, \overline{k}]$, from the joint distribution matrix. These two ranges must intersect; otherwise, the user input is not possible.

These two ranges both are intervals. If the following conditions are satisfied, these two intervals will intersect as required:

$fmax \geq \underline{k}$ and $fmin \leq \overline{k}$.

Since

$$\underline{k} = \underline{x_1 y_1} p_{11} + \underline{x_1 y_2} p_{12} + \underline{x_1 y_3} p_{13} + ... + \underline{x_2 y_1} p_{21} + \underline{x_2 y_2} p_{22} + \underline{x_2 y_3} p_{23} + ... + \underline{x_n y_m} p_{nm}$$
$$\overline{k} = \overline{x_1 y_1} p_{11} + \overline{x_1 y_2} p_{12} + \overline{x_1 y_3} p_{13} + ... + \overline{x_2 y_1} p_{21} + \overline{x_2 y_2} p_{22} + \overline{x_2 y_3} p_{23} + ... + \overline{x_n y_m} p_{nm}$$

we know *fmin* and *fmax*. Therefore, we get an additional two linear constraints for the linear programming problems based on correlation:

$$\underline{x_1 y_1} p_{11} + \underline{x_1 y_2} p_{12} + \underline{x_1 y_3} p_{13} + ... + \underline{x_2 y_1} p_{21} + \underline{x_2 y_2} p_{22} + \underline{x_2 y_3} p_{23} + ... + \underline{x_n y_m} p_{nm} \leq fmax$$
$$\overline{x_1 y_1} p_{11} + \overline{x_1 y_2} p_{12} + \overline{x_1 y_3} p_{13} + ... + \overline{x_2 y_1} p_{21} + \overline{x_2 y_2} p_{22} + \overline{x_2 y_3} p_{23} + ... + \overline{x_n y_m} p_{nm} \geq fmin$$

3.3 Nonlinear Optimization to Remove Excess Width

From the term $\rho * \sqrt{D(X) * D(Y)} + E(X)E(Y)$, we defined the corresponding function *f*(*x*,*y*): $f(x, y) = Ex * Ey + \rho\sqrt{(Ex^2 - (Ex)^2)(Ey^2 - (Ey)^2)}$. Recall that if we replace *x* and *y* with intervals *X* and *Y* in *f*(*x*,*y*), an interval function results.

Based on the rule "cancellation or reduction of the number of occurrences of a variable before interval evaluation" for eliminating excess width, if each variable occurs only once then evaluating an interval function cannot result in excess width. However, it is impossible to use this rule for this function. Instead, we avoid the excess width problem by evaluating this function in the real domain using real numbers *x* belonging to interval *X* and *y* belonging to *Y*. We can use the minimum value and the maximum value of this real function as the way to get bounds on the interval.

Rewrite the formula with intervals *X* and *Y*:

$$f(X, Y) = E(X) * E(Y) + \rho\sqrt{(E(X^2) - (E(X))^2)(E(Y^2) - (E)Y))^2)}$$
$$= \sum_i X_i P_{xi} \sum_j Y_j P_{yj} + \rho \sqrt{(\sum X_i^2 P_{xi} - (\sum_i X_i P_{xi})^2)(\sum Y_j^2 P_{yj} - (\sum_j Y_j P_{yj})^2)}$$

The corresponding real function is

$$f(x, y) = \sum_i x_i p_{xi} \sum_j y_j p_{yj} + \rho \sqrt{(\sum x_i^2 p_{xi} - (\sum_i x_i p_{xi})^2)(\sum y_j^2 p_{yj} - (\sum_j y_j p_{yj})^2)}$$

Here $x_i \in X_i$ and $y_j \in Y_j$. If ρ is an interval number, it becomes another variable for function *f*.

Obviously, *f*(*x*,*y*) is a non-linear function. We use non-linear optimization to figure out the minimum and maximum. However, this optimization question is restricted to a special region, the intervals for the x_i's and y_j's.

3.4 Improving Results by Adding Constraints to LP

Based on the above discussion, we get another two constraints for LP after calculating the interval *k*. From the joint distribution matrix in the following table,

Table 3.2 Joint distribution for *X* and *Y*.

	[…]	…	[…]	X
[…]	p_{11}	…	p_{1n}	px_1
…	…	…	…	…
[…]	p_{m1}	…	p_{mn}	px_m
Y	py_1	….	py_n	1

we get the LP model:

Minimize $Z = \sum_{i,j\in\Omega} p_{ij}$

subject to:
$$\begin{cases} \sum_j p_{ij} = p(x_i), i = 1...m \\ \sum_i p_{ij} = p(y_j), j = 1...n \\ p_{ij} \geq 0, p(x_i) \geq 0, p(y_j) \geq 0, \\ \sum p(x_i) = 1, \sum p(y_j) = 1 \end{cases}$$

To these we add the two constraints implied by the correlation. When the two constraints, $\sum \overline{a_{ij}} p_{ij} = \overline{k}$ and $\sum \underline{a_{ij}} p_{ij} = \underline{k}$, are added to the LP, the transportation simplex method cannot handle this augmented model because we cannot put these two constraints into the balanced transportation tableau.

Therefore, we use the traditional simplex method to solve the problem. The speed of calculation is very important. This is discussed later.

3.5 Simplex Method

Consider the standard LP question:

Min $Z = CX$

Subject to: $AX = b$, $x_i \geq 0$ and $b_i \geq 0$ for *i*=1 to *n*.

Here $C = (c_1,...,c_n)$ is a row vector, $X = \begin{pmatrix} x_1 \\ ... \\ x_n \end{pmatrix}$ is a column vector. $A = (P_1,...,P_n)$ and $P_i = \begin{pmatrix} a_{1i} \\ ... \\ a_{mi} \end{pmatrix}$. Therefore, *A* is an *m***n* matrix and $b = \begin{pmatrix} b_1 \\ ... \\ b_m \end{pmatrix}$ is a column vector.

We can transform the maximization problem to a minimization problem through the following approach.

Max $Z = CX$ $\Leftrightarrow$ Min $Y = -Z = -CX$

The constraints are unchanged.

Based on the simplex method, *A* is split into $(A_B \quad A_N)$. A_B has the coefficients for the basic variables (assuming there are m basic variable from x_1 to x_m), and A_N has the coefficients for the non-basic variables (from x_{m+1} to x_n). *X* is also separated into $\begin{pmatrix} X_B \\ X_N \end{pmatrix}$.

As a result, *AX*=*b* becomes $(A_B \quad A_N)\begin{pmatrix} X_B \\ X_N \end{pmatrix} = b$.

$A_B * X_B + A_N * X_N = b$

$X_B = A_B^{-1} * (b - A_N * X_N)$

Here A_B^{-1} means the inverse matrix of A_B. In other word, $A_B * A_B^{-1} = I$ where I is the unit matrix. For example, $\begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix}$ is a 3*3-unit matrix.

Thus, the objective function becomes

$$Z = C * X = (C_B \quad C_N) * \begin{pmatrix} X_B \\ X_N \end{pmatrix}$$

$$= C_B * X_B + C_N * X_N$$

$$= C_B * A_B^{-1}(b - A_N * X_N) + C_N * X_N$$

$$= C_B A_B^{-1} b + (C_N - C_B A_B^{-1} A_N) X_N$$

Here is an example.

Minimize $z = 3x_1 - x_2 - 7x_3 + 3x_4 + x_5$

Subject to: $\begin{cases} 5x_1 - 4x_2 + 13x_3 - 2x_4 + x_5 = 20 \\ x_1 - x_2 + 5x_3 - x_4 + x_5 = 8 \\ x_i \geq 0, i = 1...5 \end{cases}$

Here $C = \begin{pmatrix} 3 & -1 & -7 & 3 & 1 \end{pmatrix}$, $X = \begin{pmatrix} x_1 \\ ... \\ x_n \end{pmatrix}$, $A = \begin{vmatrix} 5 & -4 & 13 & -2 & 1 \\ 1 & -1 & 5 & -1 & 1 \end{vmatrix}$ and $b = \begin{pmatrix} 20 \\ 8 \end{pmatrix}$.

If we assume x_1 and x_2 are the basic variables, we get $X_B = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$, $X_N = \begin{pmatrix} x_3 \\ x_4 \\ x_5 \end{pmatrix}$,

$C_B = \begin{pmatrix} 3 & -1 \end{pmatrix}, C_N = \begin{pmatrix} -7 & 3 & 1 \end{pmatrix}, A_B = \begin{vmatrix} 5 & -4 \\ 1 & -1 \end{vmatrix}, A_N = \begin{vmatrix} 13 & -2 & 1 \\ 5 & -1 & 1 \end{vmatrix}$. We also get

$$A_B^{-1} = \begin{vmatrix} 1/3 & -2/3 \\ 1/6 & -5/6 \end{vmatrix}, \; X_B = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = A_B^{-1}(b - A_N X_N).$$

The following discussion is based on the previous definition and equations, and in part on Qian and Murty [1985].

3.5.1 How to Find the Initial Feasible Solution

For the standard LP question, if you can find a unit $m \times m$ matrix in A, you let this matrix be A_B by multiplying one row by a constant and adding it to another row, repeating as needed. Set X_N (non-basic variables) to zero (that is $x_{m+1}, x_{m+2}, ..., x_n$ all equal 0). Then, $X_B = A_B^{-1}(b - A_N X_N) = A_B^{-1} * b = I * b = b$ because since A is a unit matrix, as is A^{-1}. Then, $x_i = b_i \geq 0, (i = 1...m)$ is a feasible solution, although it is probably not the optimal solution.

If you cannot find a unit matrix, you can choose a sub-matrix ($m \times m$) of A which is nonsingular (meaning that the determinant of the matrix does not equal zero and the rank of the matrix is m), and every x_i of $X_B = A_B^{-1} * b$ is not less than 0. Under this condition, it is a feasible solution.

However, frequently, it is necessary to add artificial variables. To $AX = b$, we add the artificial variables $Y = \begin{pmatrix} y_1 \\ ... \\ y_m \end{pmatrix} \geq 0$, and revise the equation to $AX + IY = b$. In the objective function, the coefficients of Y should be very large positive real numbers so that the minimization objective function will be unaffected by artificial variables Y. Yet we can still use Y as the initial feasible solution. Importing the artificial variables just provides a convenient way to get an initial feasible solution.

3.5.2 How to Decide the Termination Condition and Entering Variable

Now consider optimization of Z. Let $W = C_N - C_B A_B^{-1} A_N = (w_1, ..., w_{n-m})$. Here w_i is the coefficient of x_{m+i} and describes the coefficient of a non-basic variable in the objective function.

If we want to make $Z = C * X = C_B A_B^{-1} b + (C_N - C_B A_B^{-1} A_N) X_N = C_B A_B^{-1} b + W X_N$ smaller, we must hope to find the negative elements of W because all elements of X_N are positive. From this discussion, we can derive the termination rule for an iterative optimization process.

- If every element w_i of W is not less than 0, then the current solution is optimal.
- If at least one element of W is negative, we continue to search for the optimal solution.

Let $w_k = \min(w_i \mid w_i < 0)$. This means if every non-basic variable changes by the same factor, value $w_k * x_{m+k}$ will have the maximum effect in minimizing the value of Z. Therefore, let non-basic variable x_{m+k} be the entering variable (entering the basic variable set from non-basic variable set).

If $A_B^{-1} P_{m+k} \leq 0$, there is no solution (k is the entering variable index, and P_{m+k}, belonging to $A_N = (P_{m+1}, ..., P_n)$, is the coefficient for non-basic variable x_{m+k}).

Proof:

From $X_B = A_B^{-1} * (b - A_N * X_N)$, assuming the entering variable x_{m+k} does not equal 0 and other non-basic variables still equal 0, let x_{m+k} equal α and be greater than 0. Then

$$X_B = A_B^{-1} * b - A_B^{-1} A_N * X_N$$
$$= A_B^{-1} * b - A_B^{-1}(P_{m+1},...,P_n)\begin{pmatrix} x_{m+1} \\ ... \\ x_n \end{pmatrix}$$
$$= A_B^{-1} * b - A_B^{-1} P_{m+k} x_{m+k}$$
$$= A_B^{-1} * b - \alpha A_B^{-1} P_{m+k}$$

Because $A_B^{-1} P_{m+k} \le 0$, X_B still are greater than 0, and $X_N = 0$ except for $x_{m+k} = \alpha$. Therefore, it is a feasible solution. Consider the objective function:

$$Z = C_b A_B^{-1} b + (C_n - C_b A_B^{-1} A_N) X_n$$
$$= C_b A_B^{-1} b + (w_1,...,w_{n-m})\begin{pmatrix} x_{m+1} \\ ... \\ x_n \end{pmatrix}$$
$$= C_b A_B^{-1} b + w_k * \alpha$$

Because w_k is less than 0, if $\alpha \to +\infty$, $Z \to -\infty$.

Therefore, there is no minimum value for the objective function.

Here is an example:

Minimize $z = -x_1 - x_2$

Subject to: $\begin{cases} -2x_1 + x_2 + x_3 = 4 \\ x_1 - x_2 + x_4 = 2 \\ x_i \ge 0, i = 1...4 \end{cases}$

We choose x_2 and x_4 as basic variables. Then $A_B = \begin{vmatrix} 1 & 0 \\ -1 & 1 \end{vmatrix}$, $A_N = \begin{vmatrix} -2 & 1 \\ 1 & 0 \end{vmatrix}$, $C_B = \begin{pmatrix} -1 & 0 \end{pmatrix}$ and $C_N = \begin{pmatrix} -1 & 0 \end{pmatrix}$. $A_B^{-1} = \begin{vmatrix} 1 & 0 \\ 1 & 1 \end{vmatrix}$, so $W = C_N - C_B A_B^{-1} A_N = \begin{pmatrix} -3 & 1 \end{pmatrix}$. We can choose x_1 as the entering variable. We get $A_B^{-1} P_1 = \begin{vmatrix} 1 & 0 \\ 1 & 1 \end{vmatrix}\begin{pmatrix} -2 \\ 1 \end{pmatrix} = \begin{pmatrix} -2 \\ -1 \end{pmatrix} < 0$. Now let x_1 equal $\beta > 0$. So

$$X_B = A_B^{-1} b - A_B^{-1} A_N * X_N = \begin{pmatrix} 4 \\ 6 \end{pmatrix} - \begin{vmatrix} 1 & 0 \\ 1 & 1 \end{vmatrix}\begin{vmatrix} -2 & 1 \\ 1 & 0 \end{vmatrix}\begin{pmatrix} x_1 \\ x_3 \end{pmatrix} = \begin{pmatrix} 4 \\ 6 \end{pmatrix} - x_1\begin{pmatrix} -2 \\ -1 \end{pmatrix} = \begin{pmatrix} 4 + 2x_1 \\ 6 + x_1 \end{pmatrix}$$

Here X_B is a feasible solution if $x_1 \geq 0$. But

$Z = \begin{pmatrix} -1 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = -x_1 - (4 + 2x_1) = -4 - 3x_1$. If $x_1 = \beta \to \infty$, then $Z \to -\infty$. Thus, there is no minimum value for *Z*.

Based on the previous discussion, three conditions can occur during the iterative procedure.

- A solution is found.
- Continuing to search for a solution that minimizes Z.
- No minimization solution is found.

3.5.3 How to Determine the Leaving Variable

Let X_B be a feasible solution. Thus, $A_B * X_B = b$. Here $A_B = \begin{pmatrix} P_1 & \ldots & P_m \end{pmatrix}$. We know A_B is nonsingular, so P_1 to P_m are the independent vectors. The other vectors P_{m+1} to P_n are linearly dependent on P_1 to P_m. Therefore, we can get

$$P_{m+j} = \sum_{i=1}^{m} \alpha_{i,m+j} * P_i$$

$$\Rightarrow P_{m+j} - (P_1, \ldots, P_m) * \begin{pmatrix} \alpha_{1,m+j} \\ \ldots \\ \alpha_{m,m+j} \end{pmatrix} = 0$$

From $A_B * X_B = b$, we get

$\begin{pmatrix} P_1 & \ldots & P_m \end{pmatrix} * X_B = b$.

Let β a positive real number. Then

$$(P_1, \ldots, P_m) * X_B + \beta(P_{m+j} - (P_1, \ldots, P_m) * (\alpha_{1,m+j}, \ldots, \alpha_{m,m+j})') = 0$$
$$\Rightarrow (P_1, \ldots, P_m) * (X_B - \beta(\alpha_{1,m+j}, \ldots, \alpha_{m,m+j})') + \beta P_{m+j} = 0.$$

Let x_{m+j} replace a variable in X_B. We can get a new feasible solution if we set suitable values for X and ensure $x_i \geq 0$. We can get a suitable solution from the previous formulation by setting the new X_B to $X_B - \beta(\alpha_{1,m+j}, \ldots, \alpha_{m,m+j})'$. We will let one element that equals 0 to be replaced by x_{m+j}. To assure the other variables in X_B stay positive, choose a suitable β. Let

$$\beta = \min\left(\frac{x_i}{\alpha_{i,m+j}} \mid \alpha_{i,m+j} > 0\right) = \frac{x_l}{\alpha_{l,m+j}}.$$

This implies that x_l is the leaving variable and entering variable $x_{m+j} = \dfrac{x_l}{\alpha_{l,m+j}}$.

Now we can apply this result. Based on

$$X_B = A_B^{-1} * (b - A_N * X_N)$$
$$= A_B^{-1} b - A_B^{-1} A_N * X_N$$
$$= A_B^{-1} b - x_{m+j} * A_B^{-1} P_{m+j}$$

We know x_{m+j} is the entering variable. Determine the leaving variable by choosing the minimum β using the equation

$$\beta = \min\left(\frac{(A_B^{-1} b)_i}{(A_B^{-1} P_{m+j})_i} \mid (A_B^{-1} P_{m+j})_i > 0\right) = \frac{(A_B^{-1} b)_l}{(A_B^{-1} P_{m+j})_l}.$$

This implies that the leaving variable is x_l.

3.5.4 Decreasing the Computational Cost

The simplex method is a good way to solve linear programming, but can take considerable computing resources.

From the previous discussion, we can see the main complexity problem focuses on the inverse matrix A_B. If we can find a better way to compute it, we can get better efficiency. A relatively straightforward approach is to find the relationship between the two A_B's in the closing steps. If we can use the previous A_B to speed computing the next A_B, it will help. If the original $A_B = (P_1 \quad ... \quad P_m)$, the new A_B is $A_B = (P_1 \quad ... \quad P_{l-1} \quad P_{m+k} \quad P_{l+1} \quad ... \quad P_m)$.

There is only one different column so the coefficient of the leaving variable is replaced by that of the entering variable. We can guess there is a relationship between these two A_B. From $X_B^{old} = A_B^{old\,-1} b$, $X_B^{new} = A_B^{old\,-1} b - x_{m+k} * A_B^{old\,-1} P_{m+k} = A_B^{new\,-1} b$, and basic variable x_l is replaced with x_{m+k}, and $A_B^{new} = A_B^{old} C$. Then $A_B^{new\,-1} = C^{-1} A_B^{old\,-1} = D A_B^{old\,-1}$.

Therefore, if we can find *D*, the inverse of *C*, we will speed computing the inverse of A_B. From the relationship of the original and new X_B, $x_i^{new} = x_i^{old} - x_{m+k}^{new} a_{ik}, i = 1...m, i \neq l$ and $x_{m+k}^{new} = x_l^{old} / a_{lk}$. Here $a_{ik} = (B^{-1} P_{m+k})_i$, *i*=1…*m* (*i* refers to the *i*th element of the vector $B^{-1} P_{m+k}$). We note that

$D = (e_1,...,e_{l-1},Ek,e_{l+1},..,e_m)$ and $e_i = \begin{pmatrix} 0 \\ ... \\ 1 \\ ... \\ 0 \end{pmatrix}$, and only element of *i* row is 1, while the others are 0. $Ek = (-a_{1k}/a_{lk},...,-a_{(l-1)k}/a_{lk},1/a_{lk},-a_{(l+1)k}/a_{lk},..,-a_{mk}/a_{lk})'$.

This way, the previous inverse matrix is used to calculate the new inverse matrix. Sposito (1989) gives a similar description of this method.

3.5.5 Applying the Method

For our case:

Min $Z = CX$

subject to: $AX = b$ and $X>=0$.

Using artificial variables $X_{av} = \begin{pmatrix} x_{n+1} \\ ... \\ x_{n+m} \end{pmatrix}$, the equation becomes $AX + IX_{av} = b$. Let *Con* be a very large positive real number based on the Big-M method so that the objective function becomes $Z = CX + Con*(1 \quad ... \quad 1)X_{av}$.

Based on the previous discussion, *X* are non-basic variables. A_B equals *I*. It is easy to compute. It is not needed to calculate the inverse matrix. But artificial variables are in the objective function and must be removed. If an artificial variable is removed from the basic variables, it will be removed from the objective function. This means the coefficient of the artificial variable becomes zero instead of one. After changing the coefficient of an artificial variable to zero, the artificial variable is in effect not present. When the optimum is reached, the coefficients of the artificial variables must be zero. Otherwise, there is no optimum.

3.5.6 An Example

Minimize $Z = x_1 - 3x_2 + 2x_3$

subject to: $\begin{cases} 3x_1 - x_2 + 2x_3 = 7 \\ -2x_1 + 4x_2 = 12 \\ x_i \geq 0, i = 1,2,3 \end{cases}$

Solution is found since using artificial variables x_4 and x_5 for an initial feasible solution.

The problem changes to:

minimize $Z = x_1 - 3x_2 + 2x_3 + M*(x_4 + x_5)$

subject to: $\begin{cases} 3x_1 - x_2 + 2x_3 + x_4 = 7 \\ -2x_1 + 4x_2 + x_5 = 12 \\ x_i \geq 0, i = 1,...,5 \end{cases}$

To remove the effects of the artificial variables, we set the coefficient *M* of the artificial variables in the objection function to a very large real number, such as 100,000.

Iteration 1:

$C = \begin{pmatrix} 1 & -3 & 2 & 100000 & 10000 \end{pmatrix}$, $b = \begin{pmatrix} 7 \\ 12 \end{pmatrix}$. $A = \begin{vmatrix} 3 & -1 & 2 \\ -2 & 4 & 0 \end{vmatrix} = \begin{pmatrix} P_1 & P_2 & P_3 \end{pmatrix}$. $A_B^0 = I$.

Variables x_4 and x_5 are the basic variables. $A_N = A$.

$$W = C_N - C_B A_B^{-1} A_N = \begin{pmatrix} 1 & -3 & 2 \end{pmatrix} - \begin{pmatrix} 100000 & 100000 \end{pmatrix} I \begin{vmatrix} 3 & -1 & 2 \\ -2 & 4 & 0 \end{vmatrix}$$

$$= \begin{pmatrix} 1 & -3 & 2 \end{pmatrix} - \begin{pmatrix} 100000 & 300000 & 200000 \end{pmatrix} = \begin{pmatrix} -99999 & -300003 & -199998 \end{pmatrix}$$

As a result, x_2 is the entering variable. In the next step, the leaving variable is determined.

$A_B^{-1} b = \begin{pmatrix} 7 \\ 12 \end{pmatrix}$, $A_B^{-1} P_2 = \begin{pmatrix} -1 \\ 4 \end{pmatrix}$. Thus the leaving variable is x_5. $a_{lk} = 4$.

Therefore, $E_k = (1/4, 1/4)$. We get $A_B^{1\,-1} = EA_B^{0\,-1} = \begin{vmatrix} 1 & 1/4 \\ 0 & 1/4 \end{vmatrix}$.

Iteration 2:

Now x_4 and x_2 are basic variables, and x_5 is discarded. $C = \begin{pmatrix} 1 & -3 & 2 & 100000 \end{pmatrix}$.

$A_N = \begin{vmatrix} 3 & 2 \\ -2 & 0 \end{vmatrix}$

$$W = C_N - C_B A_B^{-1} A_N = \begin{pmatrix} 1 & 2 \end{pmatrix} - \begin{pmatrix} 100000 & -3 \end{pmatrix} \begin{vmatrix} 1 & 1/4 \\ 0 & 1/4 \end{vmatrix} \begin{vmatrix} 3 & 2 \\ -2 & 0 \end{vmatrix}$$

$$= \begin{pmatrix} 1 & 2 \end{pmatrix} - \begin{pmatrix} 250001.5 & 200000 \end{pmatrix} = \begin{pmatrix} -250000.5 & -199998 \end{pmatrix}$$

As a result, x_1 is the entering variable.

$A_B^{-1}b = \begin{pmatrix} 10 \\ 3 \end{pmatrix}$, $A_B^{-1}P_1 = \begin{pmatrix} 2.5 \\ -0.5 \end{pmatrix}$. Thus, the leaving variable is x_4. $a_{lk} = 2.5$.

Therefore, $E_k = (2/5, 1/5)$. We get $A_B^{2^{-1}} = EA_B^{1^{-1}} = \begin{vmatrix} 2/5 & 1/10 \\ 1/5 & 3/10 \end{vmatrix}$.

Iteration 3:

Now x_1 and x_2 are basic variables, and x_4 is discarded. $C = \begin{pmatrix} 1 & -3 & 2 \end{pmatrix}$. $A_N = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$.

$$W = C_N - C_B A_B^{-1} A_N = (2) - \begin{pmatrix} 1 & -3 \end{pmatrix} \begin{vmatrix} 2/5 & 1/10 \\ 1/5 & 3/10 \end{vmatrix} \begin{pmatrix} 2 \\ 0 \end{pmatrix}$$

$$= 2 - (-2/5) = 2.4.$$

The optimal solution becomes:

$$X_B = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = A_B^{-1}b = \begin{pmatrix} 4 \\ 5 \end{pmatrix}, \; Z = C_B X_B = \begin{pmatrix} 1 & -3 \end{pmatrix} \begin{pmatrix} 4 \\ 5 \end{pmatrix} = -11.$$

3.6 Nonlinear Optimization

In most cases, there is a function *f*(*x*), called the objective function, which belongs to C^2, meaning that the function *f*(*x*) has a second derivative. We want to find the minimum or maximum value of *f*(*x*). We can describe this question as follows:

$\min_x f(x)$

Subject to:

$x \in R^n$

where R^n is the *n*-dimension real domain.

For the maximization question, we convert it to the minimization problem according to the following formulation:

$$\max_x f(x) = -\min_x(-f(x))$$

Now only the minimization question needs to be addressed.

In this case, the variable *x* belongs to the *n*-dimension real domain. The number of dimensions may vary from 1 to *n*. This kind of minimization problem is called *unconstrained optimization.*

If any constraints are applied to the variable *x*, we have the following situation:

$\min_x f(x)$

The *p* equality constraints are: $e_i(x) = 0$ for i=1,2…*p*

and the *q* inequality constraints are: $w_j(x) \geq 0$ for j=1,2,…*q*.

This is a *constrained optimization*.

All points x satisfying all the constraints are feasible and all others are non-feasible. All feasible *x's* form the feasible region. All non-feasible *x's* form the non-feasible region. For unconstrained optimization, the feasible region is the real domain.

3.6.1 Local and Global Optimums

A local maximum is a point in the feasible region that is higher than all other points within its immediate vicinity, but not necessarily the whole feasible region. The global maximum is the maximum for the whole feasible region. The following figure illustrates local optimums:

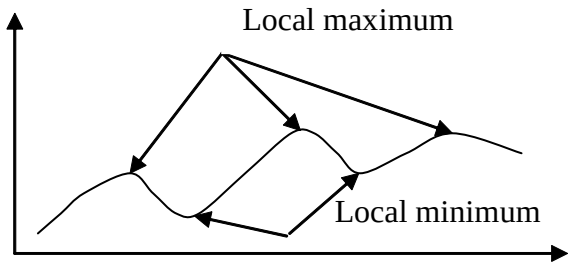

Figure 3-2 Local Optimums.

This figure illustrates the following points about the global and local optimums.

- There may be more than one local optimum for the function and their values perhaps are not the same.
- The global optimum must be a local optimum.
- A local optimum may be the global optimum.
- It is possible that there is more than one global minimum or maximum, if the function values are the same.
- The global optimum is the best of all the local optimums and is the solution for the problem.

3.6.2 Classical Theory of Unconstrained Optimization

Given a function $f(x)$, for vector x, and assume all the first derivatives $\frac{\partial f}{\partial x_i}$ exist at all points in the domain of f.

A *necessary* statement for a minimum of $f(x)$ is:

$$\frac{\partial f}{\partial x_1} = \frac{\partial f}{\partial x_2} = ... = \frac{\partial f}{\partial x_n} = 0 .$$

The condition "necessary" means that where the function is at a minimum, the equation holds. A sufficient condition also must be stated. A *sufficient* condition for a point to be a minimum of $f(x)$ is that the second derivatives of function $f(x)$ exist at the optimum point and $D_i > 0$.

$$D_i = \begin{vmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_i} \\ \cdots & \cdots & \cdots \\ \frac{\partial^2 f}{\partial x_i \partial x_1} & & \frac{\partial^2 f}{\partial x_i^2} \end{vmatrix}$$

When the derivatives of the function $f(x)$ are discontinuous, the classical theory is not fully applicable.

3.6.3 Finding a Solution Iteratively

Almost all numerical optimizations methods use iterative techniques. They start at an initial point x_0 and proceed by generating a sequence of points $x_1,\ldots x_m$ (each x_i is an n-dimension vector). Let $f(x_{i+1}) \leq f(x_i)$. Then, the minimum of $f(x)$ is approached more closely with each iteration. Clearly, the choice of x_0 is very important.
Defined by $x_{i+1} = x_i + d_i s_i$, d_i is a direction vector for finding the next x and s_i is the step size or distance to move. Here, a suitable choice of direction d_i is very important. How to search for the next x is an important issue. Typically, methods are classified into two classes: direct search and gradient methods.

3.6.4 Search Methods: Direct and Gradient

Direct search methods do not require the explicit evaluation of any derivatives of the function, but rely solely on values of the objective function $f(x)$ and information gained from earlier iterations. Some use function values to obtain numerical approximations of the derivatives.

Gradient methods select the direction using the values of the derivatives of the function $f(x)$. Usually, these methods use first order derivatives.

3.6.5 Converting Constrained to Unconstrained Optimization

For constrained optimization problems, it can be useful to make use of unconstrained optimization methods. Thus, converting to an unconstrained optimization problem is the first task. Many methods have been developed for transforming the optimization problem. The following methods are widely used:

1. Transfer functions
2. LaGrangian multipliers
3. Penalty functions.

3.6.5.1 Transfer Functions

Its basic idea is to extend the restricted feasible region to the whole real domain. For example, to minimize $f(x)$, subject to $x>a$, we can define a new variable y. Let

$$x = a + y^2$$

Using this equation, we can convert $f(x)$ to $f(y)$, and then minimize $f(y)$. Here variable y does not have any restriction. Thus, this is now an unconstrained optimization problem.

3.6.5.2 LaGrangian Multipliers

This is a very common method for transforming optimization problems. If a minimization problem has many equality constraints

$$e_i(x) = 0 \text{ for } i=1,2\ldots p$$

A new objective function to minimize can be defined with a new variable λ

$$h(x,\lambda) = f(x) + \sum_{j=1}^{p} \lambda_j e_j(x).$$

For the first derivatives of this function,

$$\frac{\partial h(x,\lambda)}{\partial x_i} = \frac{\partial f(x)}{\partial x_i} + \sum_{j=1}^{p} \lambda_j \frac{\partial e_j(x)}{\partial x_i} = 0$$

$$\frac{\partial h(x,\lambda)}{\partial \lambda_i} = e_j(x) = 0$$

The solution will satisfy the constraints $e_i(x) = 0$.

For the inequality constraints

$$w_j(x) \geq 0 \text{ for } j=1,2,\ldots q$$

we can introduce new variables called slack variables, $x_{n+1} \ldots x_{n+q}$. Let

$w_j(x) = x_{n+j}^2 \geq 0$.

Now we can transform the inequality into equality:

$$w_j(x) - x_{n+j}^2 = 0$$

Using this method, we can handle the constrained optimization problem.

3.6.5.3 Penalty Functions

The basis for the penalty function method is to define a new objective function such as:

$$h(x) = f(x) + p(c(x))$$

where $f(x)$ is the original objective function, and $p(c(x))$ is the penalty function based on the equality and inequality constraints.

For a minimization problem, the main point is to choose the penalty function to make sure that it is zero for all feasible points and is very high for all non-feasible points. Then, the minimum of $h(x)$ is equivalent to the minimum of $f(x)$.

3.6.6 Our Case

For our problems, the optimization question is defined as follows:

Find the minimum and maximum of function

$$f(x,y) = \sum_i x_i p_{xi} \sum_j y_j p_{yj} + \rho \sqrt{(\sum x_i^2 p_{xi} - (\sum_i x_i p_{xi})^2 (\sum y_j^2 p_{yj} - (\sum_j y_j p_{yj})^2}$$

subject to:

$$l \le x \le u$$
$$s \le y \le p$$

where $l = (l_1 ... l_n)$, $u = (u_1 ... u_n)$, $s = (s_1 ... s_m)$ and $p = (p_1 ... p_m)$. l_i, u_i, s_i and p_i are real numbers, not infinity. This kind of question is called box-constrained optimization or bound-constrained optimization.

We only discuss the minimization problem. For maximum problems, we can use the previous formulation to convert them to minimization problems.

Next, we need to convert the problem to an unconstrained optimization. We use the three methods introduced previously.

3.6.6.1 Transfer Function

The constraints for variable *x* and *y* are $l \le x \le u$, and $s \le y \le p$. This means that *x* lies between l and *u*, and *y* lies between *s* and *p*. so we need to introduce a new variable to replace *x* and make sure *x* satisfies the constraint.

Defining

$$x = l + (u - l)\sin^2 u \text{, and } y = s + (p - s)\sin^2 v$$

we will get the new objective function $f(u,v)$.

For this function, *y* belongs to the whole real domain, so it is unconstrained. But this function is very complicated. It is tricky use the first derivative to get the solution because *y* has many solutions.

3.6.6.2 LaGrangian Multipliers

For $l \le x \le u$, $s \le y \le p$, we can convert to:

$x - l \ge 0$, $u - x \ge 0$, $y - s \ge 0$, and $p - y \ge 0$.

Using the previous methods, we can get a new objective function. But this method introduces many slack variables and equalities. To solve these equalities is costly.

3.6.6.3 Penalty Function

We will design a suitable penalty function. Based on the constraints, we introduce this penalty function:

$$p(x) = \lambda * \max(0, l_1 - x_1, x_1 - \mu_1, ..., ..., l_n - x_n, x_n - \mu_n, s_1 - y_1, y_1 - p_1, ..., ..., s_m - y_m, y_m - p_m)$$

Here, λ will be chosen as a very large positive real number. So the new objective function is

$$h(x, y) = f(x, y) + p(x, y).$$

From this function, we see that if $l \le x \le u$, and $s \le y \le p$, then x and y belong to the feasible region and $h(x,y)$ equals $f(x,y)$, but if constraints are violated, $h(x,y)$ will become very large, clearly far from the minimum value.

3.6.6.4 Search Method

Our objective function has a good attribute; both the first derivatives and second derivatives exist. As a result, gradient search (Luenberger, 1984, pp. 384) can be applied to our case. Furthermore, gradient searching methods provide efficient direction information in searching for the next x. In view of the previous discussion, gradient search is used in our case.

3.6.6.5 Solution

Find the minimum value of function $f(x)$, stated by

Min $$f(x, y) = \sum_i x_i p_{xi} \sum_j y_j p_{yj} + \rho \sqrt{(\sum x_i^2 p_{xi} - (\sum_i x_i p_{xi})^2)(\sum y_j^2 p_{yj} - (\sum_j y_j p_{yj})^2)}$$

subject to:

$l \le x \le u$
$s \le y \le p$

where $l = (l_1 ... l_n)$, $u = (u_1 ... u_n)$, $s = (s_1 ... s_m)$ and $p = (p_1 ... p_m)$. l_i, u_i, s_i and p_i are real numbers, not infinity.

We use a penalty function to convert this problem to an unconstrained problem. The new objective function $h(x,y)$ is constructed as:

$$h(x, y) = f(x, y) + \lambda * \max(0, l_1 - x_1, x_1 - \mu_1, ..., ..., l_n - x_n, x_n - \mu_n, s_1 - y_1, y_1 - p_1, ..., ..., s_m - y_m, y_m - p_m)$$

Thys, the problem is to find the minimum value for function $h(x,y)$. For this unconstrained optimization problem, the iterative technique is adopted. First, we define some terms:

$$\bar{x} = \sum_{i}^{n} x_i p_{xi}$$

$$\bar{y} = \sum_{j}^{m} y_j p_{yj}$$

$$D_x = \sum x_i^2 p_{xi} - (\sum_i x_i p_{xi})^2$$

$$D_y = \sum y_j^2 p_{yj} - (\sum_j y_j p_{yj})^2$$

$$D_{xy} = D_x * D_y \quad .$$

Now we get the first derivative of $h(x,y)$ through $f(x,y)$ and penalty function $p(x,y)$.

$$\frac{\partial f}{\partial x_i} = p_{xi}\bar{y} + \rho * D_{xy}^{-1/2} * \tfrac{1}{2} * D_y * (2x_i p_{xi} - 2\bar{x}p_{xi})$$

$$\frac{\partial f}{\partial y_j} = p_{yj}\bar{x} + \rho * D_{xy}^{-1/2} * \tfrac{1}{2} * D_x * (2y_j p_{yj} - 2\bar{y}p_{yj})$$

$$\frac{\partial f}{\partial \rho} = D_{xy}^{1/2}$$

$$\frac{\partial p}{\partial x_i} = \begin{cases} 0 \quad others \\ \lambda \quad x_i - u_i = \max(0, l_1 - x_1, x_1 - \mu_1, ..., ..., l_n - x_n, x_n - \mu_n, s_1 - y_1, y_1 - p_1, ..., ..., s_m - y_m, y_m - p_m) \\ -\lambda \quad l_i - x_i = \max(0, l_1 - x_1, x_1 - \mu_1, ..., ..., l_n - x_n, x_n - \mu_n, s_1 - y_1, y_1 - p_1, ..., ..., s_m - y_m, y_m - p_m) \end{cases}$$

$$\frac{\partial p}{\partial y_j} = \begin{cases} 0 \quad others \\ \lambda \quad y_{ji} - p_j = \max(0, l_1 - x_1, x_1 - \mu_1, ..., ..., l_n - x_n, x_n - \mu_n, s_1 - y_1, y_1 - p_1, ..., ..., s_m - y_m, y_m - p_m) \\ -\lambda \quad s_j - y_j = \max(0, l_1 - x_1, x_1 - \mu_1, ..., ..., l_n - x_n, x_n - \mu_n, s_1 - y_1, y_1 - p_1, ..., ..., s_m - y_m, y_m - p_m) \end{cases}$$

Along the direction determined by the derivatives, the next x and y are defined. Through iteration, the numerical solution can be found.

Next, finding the maximum value of function $f(x)$,

Maximize $f(x, y)$

subject to:

$l \le x \le u$,
$s \le y \le p$.

Based on the formulation $\max_x f(x, y) = -\min_x(-f(x, y))$, we can transform this problem to:

Minimize $-f(x, y)$

subject to:

$l \leq x \leq u$
$s \leq y \leq p$

Using the previous method, we can get the minimum value *fmin*, and negate to get the maximum value of *f*(*x*), *–fmin*.

4 Enhancement of Functions

Advances in the technique were required in order to meet the needs of the problems in the electric power domain that we have been addressing. The following extensions and augmentations were developed for this purpose.

- Use of the transportation method to speed linear programming
- Cascading operations to support more than two variables
- Relational operations
- Evaluation of $f(x,y)$ for monotonic functions f

4.1 Transportation Method

In the previous version, only the standard simplex method is provided to solve linear programming. This method is much slower than that of the transportation simplex method.

4.1.1 Background on the Transportation Simplex Method

Many companies need to determine how to optimally transport goods from different warehouses to different destinations. Isomorphic problems are found in other situations unrelated to transportation, such as the assignment problem and production scheduling. [Hillier and Lieberman, 2001] give background information.

4.1.1.1 Model

In general, this kind of problem involves two different types of location: sources and destinations. Sources supply some resource and destinations accept it. Costs for transferring resources between each source and destination may be different. The aim is to minimize the total cost to transfer resource from these sources to those destinations. In most cases, the total supply for all sources is equal to the total demand for all destinations. If we have M sources, N destinations, the supply at source i is S_i, and the demand at destination j is D_j, we have that $\sum_{i=1}^{M} S_i = \sum_{j=1}^{N} D_j$. Let C_{ij} be the unit cost of moving resources from source i to destination j. Table 4.1 displays the relationship between sources and destinations.

We can describe this mode as a standard linear programming problem.

$$\min \quad Z = \sum_{i=1}^{M} \sum_{j=1}^{N} C_{ij} x_{ij}$$

Subject to:

$$\sum_{j=1}^{N} x_{ij} = S_i \quad \text{for } i=1 \ldots M$$

$$\sum_{i=1}^{M} x_{ij} = D_j \quad \text{for } j=1 \ldots N$$

and $x_{ij} \geq 0$ for all i and j.

Table 4.1 Parameter table for transportation model.

	Cost per unit distributed					
	Destination					
Source	1	2	3	……	N	Supply
1	C_{11}	C_{12}	C_{13}		C_{1N}	S_1
2	C_{21}	C_{22}	C_{23}		C_{2N}	S_2
……						
M	C_{M1}	C_{M2}	C_{M3}		C_{MN}	S_M
Demand	D_1	D_2	D_3		D_N	

If total supply is not equal to total demand, it is called an unbalanced model. For these cases, we can use dummy sources or destinations to make the model balance. If total supply is greater than total demand, we can create dummy destinations to demand extra resources and set the unit cost from each source to any dummy destinations to be very small. This way, extra resources will be transferred to dummy destinations. If total supply is less than total demand, we make up some dummy sources and set the unit cost from each dummy source to any destinations to be very large. If these unit costs are large enough, no destination will want to get resources from these dummy sources. As a result, the solution will be for resources from actual sources rather than dummy sources.

4.1.1.2 Solution

The transportation problem is a special type of linear programming problem. We can use general methods for linear programming such as the simplex method. If the simplex method is used, the simplex tableau will be complex and consists of $M+N+1$ rows and $(M+1)(N+1)$ columns. To handle this will require much computation.

For this special type of linear programming problem, there is an efficient method called the transportation simplex method. This method uses a tableau, but it only has M rows and N

columns. It is not necessary to use artificial variables to get an initial solution. It has just $M+N$-1 basic variables (not $M+N$), so a degree of freedom will be removed.

To solve transportation problems, generally two steps are necessary.

Step One: Initialization to get an initial basic feasible (BF) solution. There are 3 common methods for this step.

- Northwest corner rule
- Russell's approximation method
- Vogel's approximation method
-

Russell and Vogel's methods consider costs in generating an initial solution. The solutions are better than for the Northwest corner method. Hillier and Lieberman (2001) compare these three methods.

Step Two: Optimality testing. In this step, every solution is a feasible solution. Our aim is to find the best solution. The DEnv algorithm incorporates a loop to do the following.

- Get the two variables u_i and v_j from each basic variable's equation ($C_{ij}=u_i+v_j$).
- Calculate the related cost CC_{ij} of each non-basic variable according to $CC_{ij}=C_{ij}-u_i-v_j$.
- Get the entering non-basic variable, the one with the minimum CC_{ij} of all non-basic variables with negative CC_{ij}.
- Determine whether the solution is optimal. If all CC_{ij} are not less than 0, the solution is optimal.
- Get the leaving basic variable. This is done in a loop whose calculations use the entering non-basic variable and other basic variables. This loop identifies the cell whose assigned flow is the minimum and whose order to the entering cell is odd. This cell will be the leaving variable.
- Adjust the flow of the loop. For all the cells adjacent to the entering cell or an odd distance from it in the loop, subtract the minimum flow and for all cells an even distance, add the minimum.
- Get the new basic variable set, marking the entering cell basic variable and the leaving cell non-basic variable. Begin the loop again from step 1.

4.1.2 Exceptions in Finding the Initial Solution

Handling degeneracy when it occurs is important in finding the initial solution in a transportation simplex problem. Degeneracy means there are not enough basic variables in the initial feasible solution, such as when there are 5 basic variables for 2*3 tables. In fact, maybe only 4 variables are found for some initialization methods for some problems. This situation occurs when there are too many choices for which ones are basic. The northwest corner method does not have this kind of problem. This method always can find enough basic variables, although values of some of them may be zero. But Russell's method will have this kind of problem for some cases. Usually, the initial solution found by Russell's method is

closer to the optimal solution than that found by the northwest corner method. Thus, computing time is less for Russell's method. Therefore, there is a tradeoff.

4.1.3 Adaptation to the Unknown Dependency Case

For the unknown dependency case, the marginal distribution table for variables X and Y is really a transportation tableau. Here you can consider X as the sources and Y as the destinations. The total supply is one and total demand is one. Next, we use an example to illustrate this situation.

Example:

X distribution: $P([0,1]) = 0.2$, $P([1,2]) = 0.2$, $P([2,3]) = 0.2$, $P([3,4]) = 0.4$.

Y distribution: $P([1,2]= 0.25$, $P([2,3]) = 0.25$, $P([3,4]) = 0.2$, $P([4,5]) = 0.3$.

Consider $X+Y$ for the case of unknown dependency. We get the joint distribution tableau shown in Table 4.2.

Table 4.2 Marginal distribution.

X / *Y*	[0,1]	[1,2]	[2,3]	[3,4]	Prob.
[1,2]	[1,3] p_{11}	[2,4] p_{12}	[3,5] p_{13}	[4,6] p_{14}	0.25
[2,3]	[2,4] p_{21}	[3,5] p_{22}	[4,6] p_{23}	[5,7] p_{24}	0.25
[3,4]	[3,5] p_{31}	[4,6] p_{32}	[5,7] p_{33}	[6,8] p_{34}	0.2
[4,5]	[4,6] p_{41}	[5,7] p_{42}	[6,8] p_{43}	[7,9] p_{44}	0.3
Prob.	0.2	0.2	0.2	0.4	1

The next question is how to assign the distribution to p_{11} …p_{44} to give some subset a maximized probability. For example, to find the upper bound for $X+Y$ at 1 (the previous chapter discussed finding the subset), we get the linear programming problem

Maximize $f=p_{11}$

subject to:

$p_{11}+p_{12}+p_{13}+p_{14} = 0.25$

$p_{11}+p_{21}+p_{31}+p_{41} =0.2$

To find the upper bound for the CDF at 2, we get the problem

Maximize $f=p_{11}+p_{12}+p_{21}$

subject to:

$p_{11}+p_{12}+p_{13}+p_{14} = 0.25$

$p_{11}+p_{21}+p_{31}+p_{41} =0.2$

For every point in the support of the result distribution, we will get a linear programming problem. Through solving these problems, the upper bound of the CDF will be obtained.

The low bound of the CDF is found similarly. To speed finding the solution, we use the transportation method to solve these linear programming problems.

From the previous example, it can be seen that these linear programming problems use transportation tables. However, it is necessary to maximize the value of the objective function rather than minimize it, as in real transportation problems. To solve these problems, we use negation to transform a goal of maximization to one of minimization. The C_{ij}'s are important in transforming the problems. For the objective function, C_{ij} must be 1, 0, or -1. To transform the problem from one of maximization to one of minimization, use C_{ij}=-1 for all items that will contribute to the objective function, with others zero. For the previous two cases we will get:

Minimize $-f=-p_{11}$

subject to:

$p_{11}+p_{12}+p_{13}+p_{14} = 0.25$

$p_{11}+p_{21}+p_{31}+p_{41} =0.2$

C_{11} = -1, other C_{ij}=0

Minimize $-f=-p_{11}-p_{12}-p_{21}$

subject to:

$p_{11}+p_{12}+p_{13}+p_{14} = 0.25$

$p_{11}+p_{21}+p_{31}+p_{41} =0.2$

$C_{11}=C_{12}=C_{21}$=-1, other C_{ij}=0

This illustrates a way to transform an unknown dependency case to a transportation problem.

It includes two steps:

- get the transportation table from the marginal distribution table; and
- set the cost attribute for cells contributing to the objective function to –1, and set other cells' cost to zero.

Because the balance of supply and demand is a basic requirement for the transportation problem, we must keep marginal sum of X and Y equal to 1, and the same for Y.

4.1.4 Test Result

Consider an example:

X, $p([0,0.333])=0.2$, $p([0.333,0.667])=0.4$, $p([0.667,0.999])=0.4$
Y, $p([0,0.5])=0.5565437$, $p([0.5,1])=0.4434564$.

Consider $X+Y$ under the unknown dependency condition.

Table 4.3 Lower bound.

Result interval	Simplex	Transportation
[-0.25,0.833]	-1.490116E-08	0
[0.833,1.167]	-1.490116E-08	0
[1.167,1.333]	0.1565436	0.1565437
[1.333,1.5]	0.2	0.1999999
[1.5,1.667]	0.5565436	0.5565436
[1.667,2]	0.6	0.6
[2,2.25]	1	1

Table 4.4 Upper bound.

Result interval	Simplex	Transportation
[-0.25,0]	0	0
[0,0.333]	0.2	0.2
[0.333,0.5]	0.5565437	0.5565437
[0.5,0.667]	0.6	0.6
[0.667,0.833]	07565437	07565438
[0.833,1.167]	1	1
[1.167,2.25]	1	1

For this example, we obtain almost the same answer with both methods.

4.2 Cascading Operations

Previously, the DEnv algorithm only supported binary operations (two operands). But in real applications, there are often over 2 operands to be calculated, for example, $x+y+z$, Max(x,y,z), etc.

Association may be used to solve many such problems. For example, for $x+y+z$, we can first calculate $x+y$, and save the result to temporary variable $w=x+y$, then calculate $w+z$. However, the operation must support association and commutation.

To extend DEnv to handle cascaded operations, the following capability was added: the CDF envelopes for the result of two variables' operation were converted into a marginal of a joint distribution tableau, call it w, for use in the second step of the solution process. Thus, we need to be able to convert CDF envelops to a set of intervals and associated probabilities.

4.2.1 Solution

This section describes how to transform upper and lower envelops into a set of intervals and associated probabilities. The probability of each envelope is its top-to-bottom height. For example, four intervals will be obtained from the following CDF envelopes.

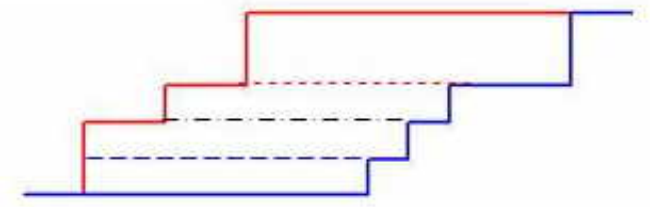

Figure 4-1 Converting CDF Envelopes to a Set of Intervals and Associated Probabilities.

Figure 4-1 shows the CDF envelopes resulting from an operation on two variables. Figure 4-2 shows the procedure to calculate using multiple operands (e.g., $x + y + z$). First, we get the result of $x + y$. This is shown in Figure 4-3. Then that result is loaded as a new operand (top panel above) and operates on it and z, which is shown in the middle panel. Figure 4-4 shows $x + y + z$.

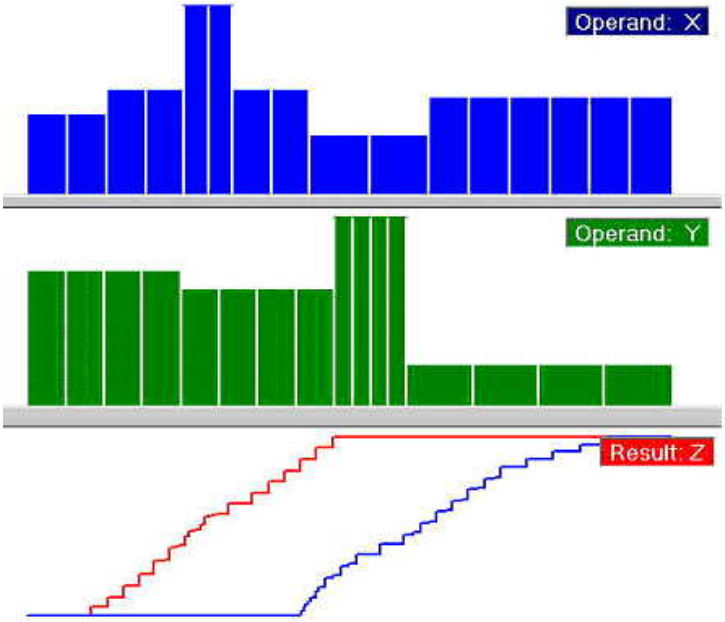

Figure 4-2 Result for Operation.

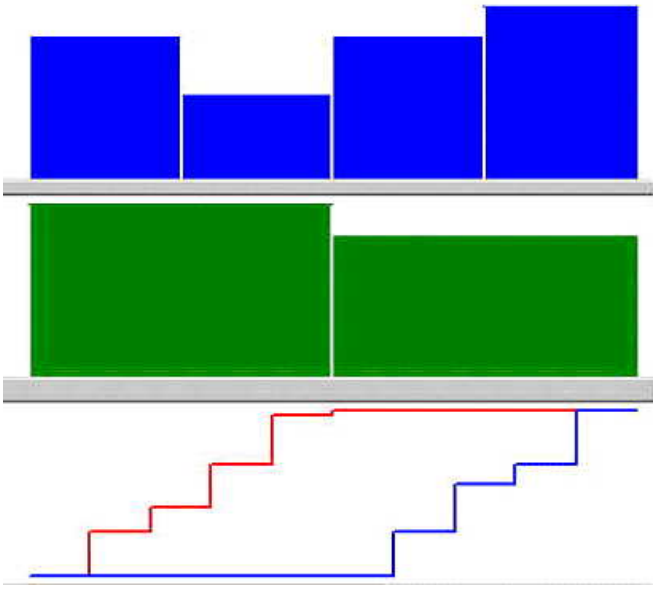

Figure 4-3 Result for $x+y$.

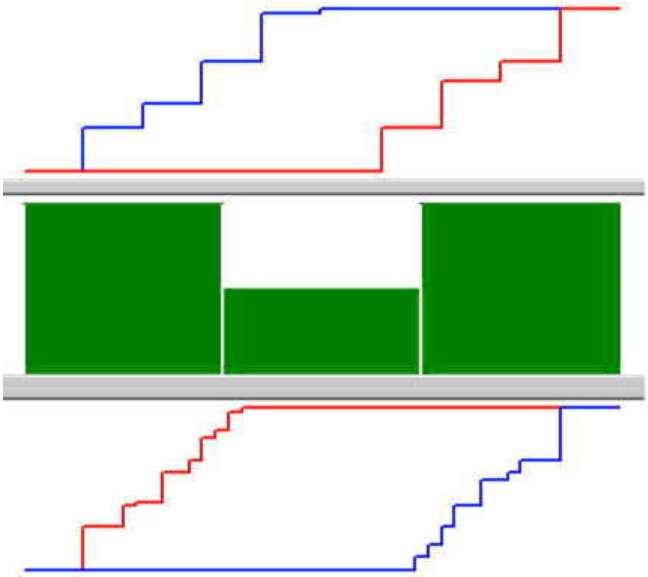

Figure 4-4 Result for $x+y+z$.

4.3 Relational Operations

Relational operations describe the relationship between two operands. DEnv handles these operations if the operations are defined to return a numerical value. We have used the value 1 (or [1,1]) to indicate the statement is true, 0 (or [0,0]) to indicate it is false, and the interval [0,1] to indicate uncertainty about whether it is true or false. The software implementation supports these four relational operations: >, >=, <, and <=.

4.3.1 Relational Operations on Intervals

Consider two real numbers x and y. We define the interval value to describe the relationship between x and y. The value [0,0] indicates that the relationship is false. The value [1,1] indicates the relational operation is true. The value [0,1] means the value of the relational operation is not determined or is uncertain.

For interval A, A-left means the left (or low) bound of A, and A-right means the right (or high) bound. Now consider two intervals A and B.

$$A > B = \begin{cases} [1,1] & A-left > B-right \\ [0,0] & A-right \le B-left \\ [0,1] & otherwise \end{cases}$$

$$A \ge B = \begin{cases} [1,1] & A-left \ge B-right \\ [0,0] & A-right < B-left \\ [0,1] & otherwise \end{cases}$$

$$A < B = \begin{cases} [1,1] & A - right < B - left \\ [0,0] & A - left \geq B - right \\ [0,1] & otherwise \end{cases}$$

$$A \leq B = \begin{cases} [1,1] & A - right \leq B - left \\ [0,0] & A - left > B - right \\ [0,1] & otherwise \end{cases}$$

4.3.2 Relational Operations on Random Variables

Consider two random variables X and Y. We consider $P(X>Y)$. In the DEnv algorithm, random variables X and Y are split into intervals that are assigned probabilities. Therefore, operation $X>Y$ is transformed into a series of interval operations. Here is an example.

Table 4.5 Distribution for X and Y.

Y \ X	[0,1]	[1,2]	[2,3]	Prob.
[1,2]	p_{11}	p_{12}	p_{13}	0.25
[2,3]	p_{21}	p_{22}	p_{23}	0.5
[3,4]	p_{31}	p_{32}	p_{33}	0.25
Prob.	0.5	0.25	0.25	1

Consider the relational operation $X>Y$. It is transformed into an interval relational operation between intervals of X and intervals of Y. For example, the result of [0,1] > [1,2] is [0,0], so [0,0] will be put into cell p_{11}. Similarly, [0,1] will be put into cell p_{12}. The following table gives the result.

Table 4.6 Interval value for a relational operation.

Y \ X	[0,1]	[1,2]	[2,3]	Prob.
[1,2]	[0,0] p_{11}	[0,1] p_{12}	[0,1] p_{13}	0.25
[2,3]	[0,0] p_{21}	[0,0] p_{22}	[0,1] p_{23}	0.5
[3,4]	[0,0] p_{31}	[0,0] p_{32}	[0,0] p_{33}	0.25
Prob.	0.5	0.25	0.25	1

Based on the DEnv algorithm, the probability for $X>Y$ may not be obtained. It is clear that all cells whose interval bounds include 1 are consistent with $X>Y$. To get the maximum value of $P(x>y)$, the sum of all cells including 1 will be maximized. For this case, we maximize $(p_{12}+p_{13}+p_{23})$. To get the minimum value of $P(x>y)$, the sum of all cells with the value [1,1]

will be minimized. All cells whose value is [0,1] will be discarded since for them, perhaps $x<=y$.

In summary, the value of each cell should be one of [0,0], [0,1], and [1,1]. Here [0,0] means the relationship does not hold. The value [0,1] means the relationship is not certain. The value [1,1] indicates the relationship must hold. To get the maximum value, maximize the sum of all cells whose bound include 1. To get the minimum value, minimize the sum of all cells whose values are [1,1].

4.4 Complex Expressions

It is useful to be able to calculate any arithmetic expression. The user of the implementation software should be able to input the expression desired. To solve this problem, DEnv needs arithmetic expression parsing to provide the functionality to interpret an arithmetic formula. We decided to support expressions containing arithmetic operators +,-,*, /, and exponentiation, and to support association through using ( ).

4.4.1 Expression Editing

To implement this expression editor, first a grammar definition of allowed expressions was written. This grammar is context-free, and is as follows.

<expression>::=<term> | <term> + <expression> |<term> - <expression>
<term>::=<factor> | <factor> * <term>| <factor> / <term>
<factor>::=(<expression>) | <number> | <variable>
<number>::= <integer> | <integer>.<integer>
<integer>::=<integer>|v
<variable>::=x|y

Here v indicates the numbers from 0 to 9.

Based on this grammar, arithmetic expressions such as $(a*X+b*Y)/(c*X+d*Y)$ are allowed. Parsing generates a parse tree: a diagram of the complete grammatical structure of the string being parsed. For this case, it is not very complex. Every operator needs two operands. '( )' will increase the priority of operation. Therefore, expression tables could be used. In such a table, the operands and operators are recorded. Every row describes an operator. Software must analyze the input string, and generate the expression table according to the order of calculations. Then using this table, the result may be calculated.

4.4.2 Limitations on Evaluating Expressions

Division by zero cannot be handled without using extended interval arithmetic, and even then there are interesting wrinkles that would need special handling. For example, in the case of $x/0>y/0$, the proper answer per the previous discussion would be the interval [0,1] since the expression evaluator does not know how to evaluate the value of such expressions.

Therefore, operands X and Y cannot include zero in their support if the user wants to use the expression parser as presently implemented.

4.4.3 Excess Width in Expressions

A typical expression is $P(x,y) = f(x,y)/g(x,y) = (aX+bY)/(cX+dY)$. For this kind expression, there may be excess width in interval calculations. This is a risk whenever a random variable is used more than once in an expression. To solve the problem, it is necessary to remove excess width in calculating this type of expression.

4.4.4 Removing Excess Width

The easiest way to handle excess width is to simplify the expression so that each random variable is used only once. But this is a very restrictive constraint, and many expressions cannot be simplified to meet this kind of condition.

For some expressions, we can employ another method to remove excess width. This method is to use the low and high bounds of the interval operands to calculate the expression. Then, from these calculated values, the result bound is determined. For two variables, there are four combinations of bounds so four candidate result values are obtained. We select the minimum of the four as the low bound of the result interval, and the maximum of the four as the high bound of the result interval. Here is an example.

Suppose: $x = [1,2]$, $y = [2,3]$, and $F(x,y) = (8.4x + 7.2y)/(0.04x + 0.02y)$.

First: let $x=1$, $y=2$, and calculate $F(x,y)$, obtaining the value 285.

Second: let $x=1$, $y=3$, and calculate $F(x,y)$, obtaining the value 300.

Third let $x=2$, $y=2$, and calculate $F(x,y)$, obtaining the value 260.

Finally, let $x=2$, $y=3$, and calculate $F(x,y)$, obtaining the value 274.3.

Thus, the interval for $F(x,y)$ is [260,300].

If we calculate the expression by using interval addition to obtain intervals for the numerator and denominator independently, then divide the two resulting intervals, the interval [162.9,480] results. This has excess width. The endpoint method can remove excess width for this expression.

4.4.4.1 Limitation of the Endpoint Method for Removing Excess Width

Although the method of selecting the min and max value to get the result bound works for the example given, there are limitations to this method. When the expression is monotonic over the 2-D box defined by the interval endpoints, the method works. When it is not monotonic over that region, another method must be used. One approach, shown to be practical by the implementation of DEnv, uses a sampling method. In this method, a grid is

placed over the box and interior values of the intervals are sampled using points on the grid. The min and max of all sampled points provides the low and high bounds of the answer. For example the expression $(x+y)^x$ was analyzed this way (Berleant and Zhang, 2003).

5 Applications

We are exploring applications of DEnv for analysis of economic dispatch, value at risk (VaR), and reliability problems as summarized in this section.

5.1 Economic Dispatch: Applying the Interval-Based DEnv Algorithm

A common way to model uncertainty in the value of a quantity is to use a probability density function (PDF) or its integral, a probability distribution function (CDF). When two such values are combined to form a new value equal to their sum, product, max, etc., the new value is termed a {\it derived distribution} [Springer, 1979]. It is well known that derived distributions may be obtained by numerical convolution, Monte Carlo simulation, and analytically for specific classes of input distributions, under the assumption that the input distributions are independent. It is also possible to obtain derived distributions for specified dependency relationships other than independence. However, it is not always the case that the dependency relationship is known. Thus, there is a need for obtaining solutions that are validated with respect to uncertainty about the dependency relationship.

Numerical approaches have the advantage of applicability to a very wide class of distributions. Two numerical algorithms have been implemented in software for obtaining solutions to combining distributions that are validated with respect to uncertainty about their dependency. Both also validate their solutions with respect to discretization of the input distributions by using intervals to account for the inexactness of the discretization, eventually producing results that incorporate that inexactness into the separation of the envelopes. One algorithm is Probabilistic Arithmetic [Williamson and Downs, 1990], which is implemented in the commercially available software tool RiskCalc [Ferson et al., 1998]. The second algorithm is Distribution Envelope analysis (DEnv) [Berleant and Goodman-Strauss, 1998].

The DEnv algorithm is implemented in a tool, Statool that extends our previous tool by eliminating the need to assume independence. While the Statool and RiskCalc tools have fundamental similarities [Regan et al., submitted] a difference that is relevant to the present problem is that the DEnv algorithm supports, and Statool implements, excess width removal in the underlying interval calculations for expressions in which the true bounds of the expression occur at corners of the rectangle defined by the input intervals. This simple approach frequently works, as for example in the present application. More sophisticated approaches to excess width removal, if implemented, could be incorporated into the software without difficulty since the details of the interval calculations are decoupled from other parts of the software. The result of handling excess width is inferred envelopes that are closer together than they would be if excess width was not handled [Berleant, 1993]. In this section we apply the DEnv algorithm to generalize a solution to the well-known economic dispatch problem in electric power generation to the case where the dependency relationship between the fuel costs of two generators is unspecified.

5.1.1 The Problem

Interval methods have continued to draw the attention of researchers in the power generation community [e.g., Wang and Alvarado, 1992; Shaalan and Broadwater, 1993; Shaalan, 2000]. One electric power problem that, as traditionally formulated, is well understood is the economic dispatch problem. In this problem, it is desired to determine how much power should be generated by each of two generators, to meet a given level of demand such that total generation cost is minimized. One of a number of approaches to solving this problem is termed LaGrangian Relaxation [Wood and Wollenberg, 1996].

We added a new dimension to this problem by incorporating uncertainty into the LaGrangian Relaxation technique for solving the problem, by modeling uncertainty in the cost of fuel to run the generators with probability distributions, postulating in addition that the dependency between the two fuel costs of the two generators is unknown (as might occur if one generator burns oil and the other coal). The uncertainties are then propagated through the algebraic expression derived by the LaGrangian Relaxation technique.

First, the cost equations are specified as

$$F_1 = v_1(8P_1 + 0.024P_1^2 + 80)$$
$$F_2 = v_2(6P_2 + 0.04P_2^2 + 120)$$

[Wood and Wollenberg, 1996] where P_1 and P_2 are the power outputs of generators 1 and 2 in megawatts; v_1 and v_2 are the fuel costs for generators 1 and 2 in $ per M Btu; and F_1 and F_2 are the generation costs for given power output levels and fuel cost rates. Therefore, generation costs change nonlinearly with power output according to the following equations.

$$\frac{dF_1}{dP_1} = v_1(8 + 0.048P_1)$$
$$\frac{dF_2}{dP_2} = v_2(6 + 0.08P_2) \qquad (5.1)$$

Solving the problem requires minimizing an objective function,

$$F = F_1 + F_2 = v_1(8P_1 + 0.024P_1^2 + 80) + v_2(6P_2 + 0.04P_2^2 + 120)$$

subject to the constraint

$$P = P_1 + P_1^2$$

P is the total customer demand for electric power that for this example we take as 400 megawatts. This gives a constraint function

$P=P_1+P_2=400$.

by the method of LaGrangian multipliers from calculus, at an extreme value of this objective function,

$$\frac{dF_1}{dP_1} = \frac{dF_2}{dP_2} = \lambda \qquad (5.2)$$

for some λ. This is derived from the Lagrange function L that relates objective function F and constraint (5.1) according to $L = F + \lambda P$, which implies $\frac{\partial L}{\partial P_1} = \frac{dF_1(P_1)}{dP_1} - \lambda = 0$ for generator 1 and similarly for generator 2. From (5.1) and (5.2),

$v_1(8+0.048P_1) = \lambda = v_2(6+0.08P_2)$
$P_2 = 400 - P_1$

and solving simultaneous equations for P_1 gives

$$P_1 = \frac{38v_2 - 8v_1}{0.08v_2 + 0.048v_1} \qquad (5.3)$$
$$P_2 = 400 - P_1$$

as the most economical amounts of power to generate from generators 1 and 2 to meet the demand (assuming those amounts are within the capacity of both generators). P_1 and P_2 are easily calculated for real values of v_1 and v_2, but given distribution functions for v_1 and v_2 the problem requires evaluating an expression on random variables v_1 and v_2 involving a sum, difference and quotient. Solving it by dividing a difference of random variables by a sum results in excessively wide envelopes on the CDFs for P_1 and P_2 because the same operands occur in both terms, leading to excess width in the underlying interval calculations. Instead, the entire expression must be treated as a single binary operation on v_1 and v_2. Figure 5-1 shows the results given PDFs describing v_1 and v_2.

The CDF for optimum power generation from generator 1 will be within the interior envelopes if the inputs v1 and v2 are independent, and within the exterior envelopes regardless of the dependency relationship between inputs v1 and v2. When the dependency relationship is not known, the exterior envelopes might be sufficient for a decision, or might point out the need for additional information gathering to sharpen the input distributions and/or identify their dependency relationship sufficiently to support a decision. [From Berleant et al., 2002.]

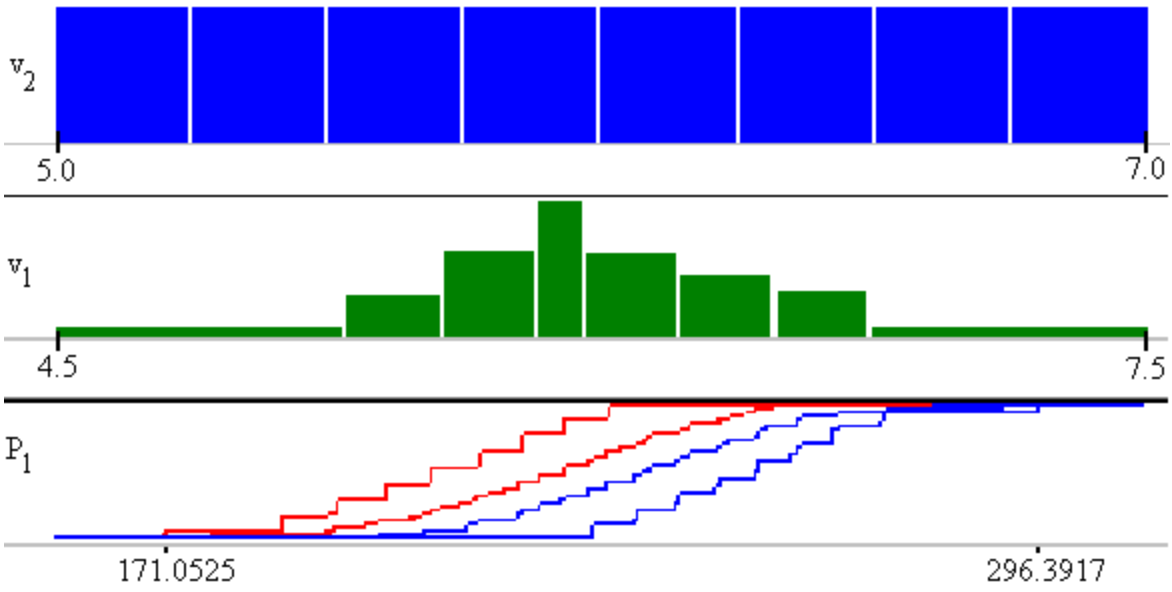

Figure 5-1 Solution, Given the Histogram-Discretized PDFs for v_1 and v_2 Shown.

5.1.2 Discussion

The present software implementation has certain limitations. One planned extension is to handling of asymptotic PDF tails. The process of discretizing a PDF into a histogram does not presently allow for the case where a PDF tail trails off to plus or minus infinity. Yet this implies setting definite bounds, though any specific such bounds might be hard to justify. Indeed unusual and extreme values can occur in the electric power domain, as happened for example in the California power crisis. The solution is to allow the discretization to include open intervals with an end point at +/- infinity. This in turn would require the arithmetic operations to be defined on such intervals. Fortunately, this is straightforward.

The overhead in time complexity due to use of interval calculations is a relevant consideration. For DEnv, time complexity overhead is attributable mainly to the increased time complexity of computing interval operations in place of what would otherwise be numerical ones. Thus, a complex excess width removal algorithm would have a correspondingly great effect on run time. The simple method employed in Statool typically adds approximately 25% to the run time, as tested by doing elementary arithmetic operations with and without the excess width-handling algorithm, when the dependency relationship between the operands is considered unknown. However, when the operands are assumed to be independent, using the excess width algorithm leads to a slowdown by an approximate factor of 10, because a higher proportion of the computations done by the program in this case are interval operations, and therefore, slowing them has a correspondingly greater effect.

These results (25% and 10x) suggest comparing the speed of computation when the dependency relationship as unknown (that is, when the DEnv algorithm is used) with the

speed when the operands are assumed independent. When the operands are each discretized into 16 intervals (a 16x16 problem), simple arithmetic operations on the operands take about 30 seconds, or 25% more with excess width handling, on an Intel-based PC running at 500MHz.
In comparison, assuming independence allows the same problems to run interactively without noticeable delay when excess width handling is not used, and in about 2.5 seconds when it is used.

5.2 Bounding the Composite Value at Risk for Energy Management Company Operation with DEnv

Deregulation in the power industry drives competition. It also increases the risk of doing business. Therefore, it is important to manage and assess the risk. Value at risk (VaR) analysis has been used in financial institutions to evaluate portfolios of assets for some time, but the application of the approach in the power industry has not been established. The VaR of serving customer demand using the energy purchased on the auction market is our focus. In this report, the risks of the energy management company (EMCO) are identified and the contract specifications and the VaR reviewed. In describing the difference in the business environments between the power and financial industries, the VaR analysis that has been used in the financial industry has been remodeled to best describe the assumed deregulated power environment. The pros and cons of the VaR levels are presented. Because of the interval-based computational core of DEnv (Distribution Envelope Determination), results are validated with respect to two sources of potential error.

Given the cumulative distributions of random variables, a derived random variable, which is an arithmetic combination of the given random variables, will have a single defined cumulative distribution only if the joint distribution of the given distributions is fully defined. If the joint distribution is not defined, a verified characterization of the result will be envelopes bounding the space of cumulative distribution curves that correspond to the members of the set of all the possible joint distributions. Distribution Envelope Determination (DEnv) [Berleant and Goodman-Strauss, 1998] provides those envelopes, so that uncertainty in results due to uncertainty about dependencies among model variables is bounded.

The distributions of input random variables can be discretized in DEnv in order to avoid the problem of finding envelopes for arbitrary input distributions analytically. Discretization typically involves approximation, but DEnv can avoid this by bounding each input distribution with envelopes such that the discretized form of an input is a pair of envelopes enclosing it. While the input distribution is likely to be a continuous curve, the envelopes are staircase-shaped. This representation for the input curves propagates into wider envelopes around the space of possible result curves because those envelopes bound the space of results not only with respect to different dependency relationships between the inputs (as described in the previous section), but also with respect to the space of curves consistent with the envelopes around an input.

Results are valuable because insufficient data are typically present to specify the relevant dependencies accurately.

5.2.1 Background

Calls for competition in the power industry, from the wholesale level to the retail level, have made deregulation an attractive option around the world. New market structures have been studied to search for a good one that can ultimately satisfy regulatory bodies, customers, and suppliers. One approach that has been tried is the brokerage system. To accomplish it, the vertically integrated utilities are converted into a horizontal structure. The framework of the energy market is shown in Sheblé [1999]. Since the emphasis of this report is on the value at risk (VaR) of serving customer demand, the energy management company (EMCO), which serves customers, is discussed while leaving the rest to Sheblé [1999].

The EMCO collects its revenue from the customers of the energy and ancillary services it provides. It can also act as a wholesaler, reselling electric energy to other EMCOs, generating companies, etc. To obtain the desired electric energy to serve its purposes, the EMCO may purchase it through the auction market, or utilize the reserves that it has accumulated through load management programs or ownership of generation units.

In the deregulated environment, customers are free to choose among EMCOs. In addition, energy purchased by EMCOs from the auction market bears the risk of market price fluctuation. These, from the demand factors to the supply factors, are risks that the EMCO has to take in the new market structure. Since deregulation will render governmental financial protection largely obsolete, risk management and assessment tools should be considered and applied.

Ng and Sheblé [2000] introduce the different risk management and assessment tools available to assist an EMCO. This account emphasizes Value at Risk (VaR) analysis.

5.2.2 VaR Analysis Basics

VaR is the maximum amount of money that may be lost on a portfolio over a given period, with a given level of confidence [Best, 1998]. VaR calculations are important because exceeding an appropriately defined maximum loss would be a major or even irrecoverable blow to the company. Thus, business decisions need to be made with the objective of keeping the probability of such a loss below a relatively low level of probability deemed acceptable. Consequently determining the probability of such a catastrophic loss should be done carefully and, for dependability, should be validated with respect to lack of knowledge about the dependencies among the variables factoring into the calculation.

There are currently three techniques that can be used to evaluate VaR of an EMCO. The first technique is historical simulation, which applies historical data to evaluate the VaR. The second technique is the covariance technique. To apply the covariance technique, the correlation matrix, C, of the uncertain factors is assumed available. The third technique is

Monte Carlo simulation. Monte Carlo simulation involves artificially generating a very large set of events from which VaR is derived [Best, 1998].

The covariance technique is the easiest and fastest technique among the three. However, the technique assumes that the uncertain factors are normally distributed. Since normal distributions do not necessarily apply to all situations, the technique is consequently limited. Historical simulation and Monte Carlo simulation can supplement the covariance technique in such cases. Since historical simulation uses historical data to evaluate the VaR, there is no need to assume the form of the probabilistic distribution function of the uncertain factors. However, when historical data is limited, solving the VaR using the historical simulation method can be problematic. The Monte Carlo simulation method requires assuming the probability distributions of the uncertain factors (often that they are normal, but uncertain factors that are not normally distributed can be handled). For instance, in determining the VaR of holding option contracts (whose prices are not normally distributed), the option sensitivities (normally distributed) are used for the Monte Carlo simulation. Thus, the resulting VaR is able to consider option contracts [Best, 1998].

Best describes the VaR resulting from asset price changes, the diversity of the portfolio (the number of assets with correlated price changes), and the holding position of the portfolio (the amount of money invested in a particular asset). This evaluation process is sufficient in a financial institution where the risk is primarily a result of price changes. To an EMCO, however, evaluating the VaR of the price changes is not sufficient. In addition to the risk of price fluctuation, there are two additional risks not described by Best. First, the customer demand and the deliverability of energy are uncertain, as there is a risk associated with the EMCO not being able to serve the customer with sufficient energy. For example, energy delivery can be prevented by transmission system failure, generation failure, etc. Thus, an EMCO suffers the risk of contract violation by its supplier. The next figure shows the three components of VaR affecting a particular decision (such as amount of load management energy, number of contracts, purchased ancillary services, etc.) for an EMCO.

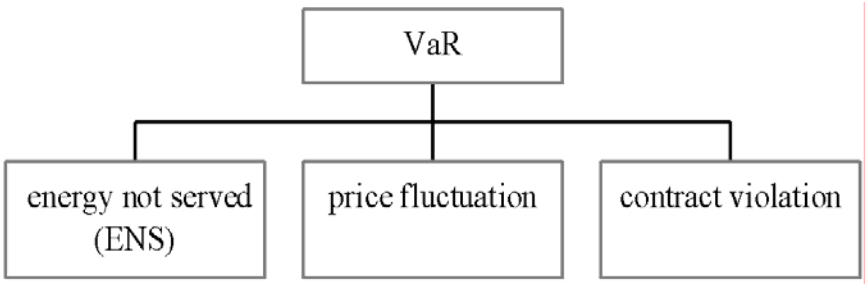

Figure 5-2 Factors in Determining VaR of an EMCO.

5.2.3 Price Fluctuation

To evaluate the VaR of market price fluctuation, the covariance matrix of the market price fluctuation is assumed available. Historical data may be used in determining the covariance matrix. Then, the VaR of market price fluctuation is evaluated using the following equation.

$$VaR = \lambda\sqrt{\mathbf{PCP}^t}$$

where **P** is the proportion or position of the assets in monetary value, λ represents the degree of volatility and determines the confidence level. For instance, when $\lambda = 1$, the confidence level is 95% [Best, 1998]. The covariance matrix, **C**, is determined as described by Sheblé and Berleant [2002]. Ng [1999] gives the steps in evaluating VaR due to market price fluctuation.

5.2.4 Sample Experiments of Computations

Our experiments focus on checking three issues: the accuracy of results, the effect of correlation, and speed. Changing the accuracy of operands will affect the accuracy of results. Different correlations will change the shapes of result envelopes. Increasing the number of intervals will take more time to compute. All our experiments were conducted on the compiled version of Statool and DLLs using Visual Basic 6.0 and Visual C++ 6.0. The running platform was Windows 2000 professional. The machine had 256M memory and the CPU ran at 1,000Mhz.

The operand *X*, a random variable, was given a uniform distribution from 1 to 9. The operand *Y*, another random variable, was given a tail-trimmed normal distribution from 2 to 10, whose mean was 6 and variance 1. This range almost covers all the probability for *Y*. A small amount of the tail was omitted and its probability distributed evenly over the accounted-for range. We discretized the supports of *X* and *Y* using three conditions, 16, 32, and 64 intervals, then used the discretized *X* and *Y* as the inputs to operations. Results of operations showed the accuracy changing for different discretizations. At the same time, correlation was set to different values to check the effects. Four operations were executed in these experiments. They were plus, minus, multiply, and divide.

The following figures show the results for different number of intervals in the operand discretizations when adding *X* and *Y* and with a correlation of zero.

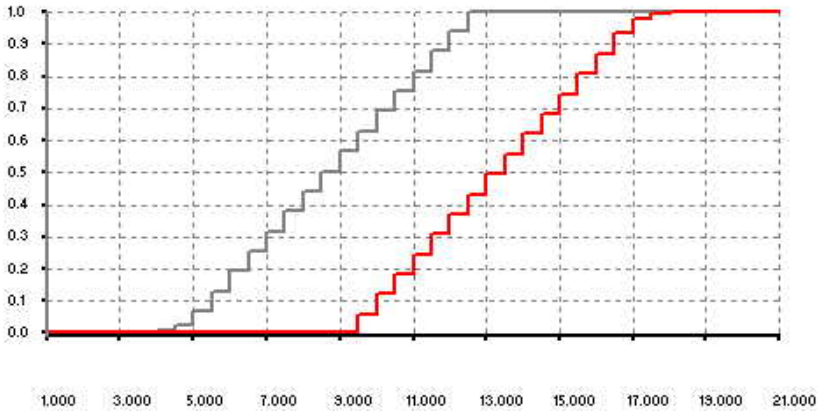

Figure 5-3 Envelopes around the distribution of $X+Y$ when X and Y are each discretized into 16 Intervals.

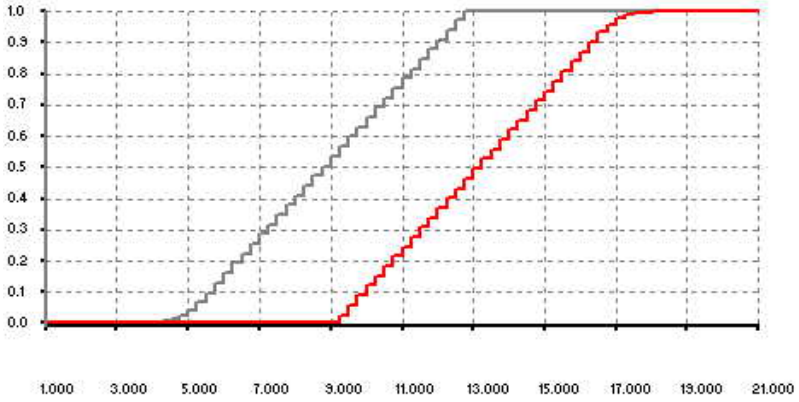

Figure 5-4 $X+Y$ when X and Y are 32 intervals.

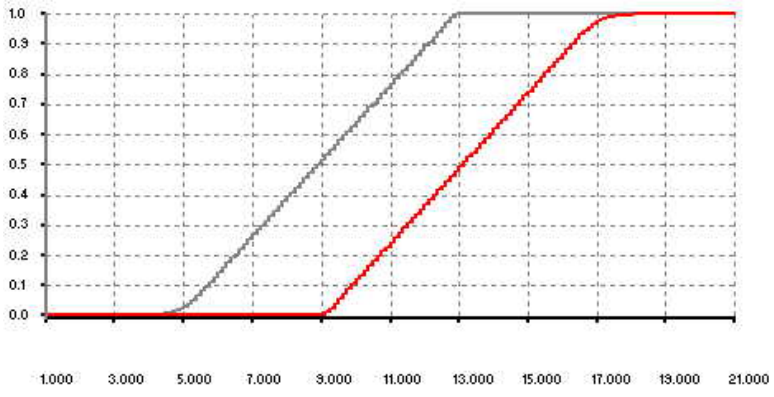

Figure 5-5 $X+Y$ when X and Y have 64 intervals.

From these three figures, it is clear the results become much smoother when the discretization of the operands is increased.

Next, we show figures illustrating the effect of correlation. For this case, we let *X* and *Y* have 64 intervals, and set correlation information to four different settings: unknown, 0.98, 0, and -0.98.

From these figures, it is clear that bounds of curves can be affected by the correlation. For unknown correlation, the widest bound curves will be obtained. Compared with the envelopes for correlation 0, the high bound curve for correlation 0.98 is changed and the low bound curve for correlation -0.98 is changed.

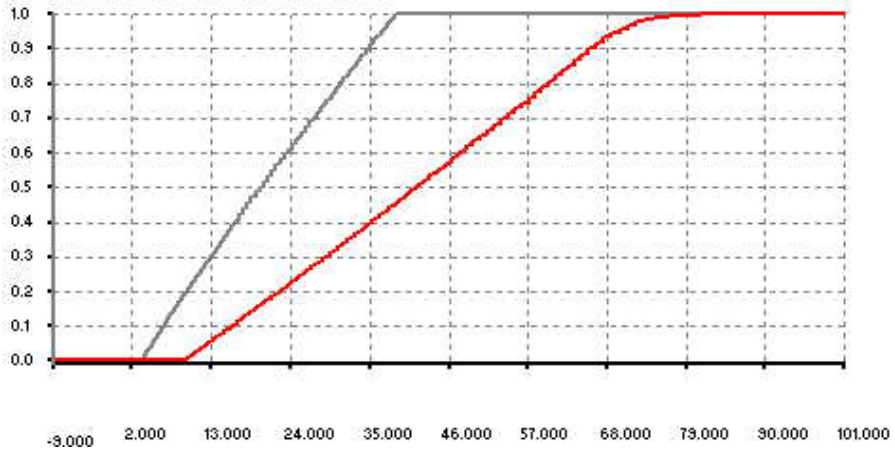

Figure 5-6 Envelopes around the Distribution of *X*Y* for an Unknown Dependency Relationship Between *X* and *Y*.

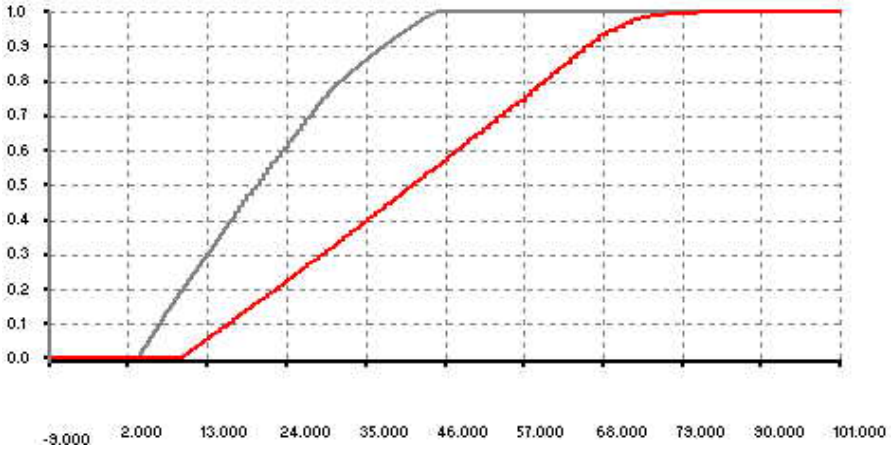

Figure 5-7 *X*Y* for Correlation 0.98.

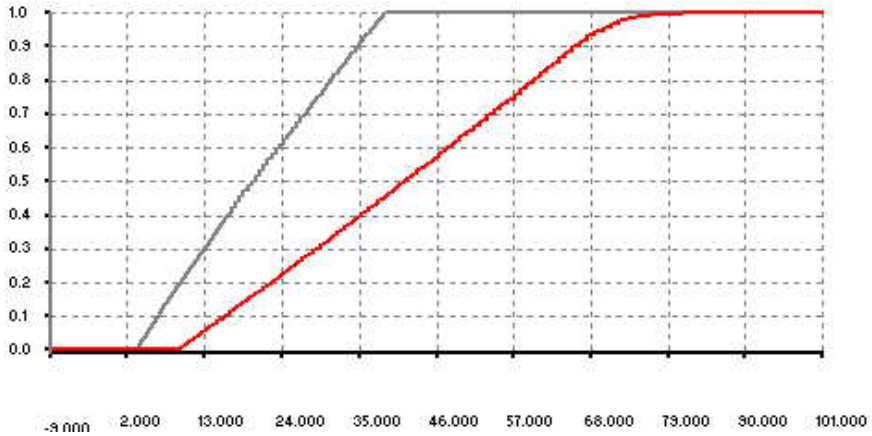

Figure 5-8 *X*Y* for Correlation 0.

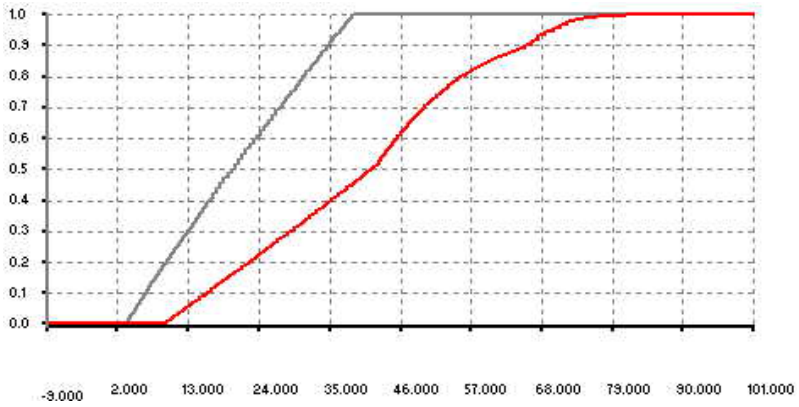

Figure 5-9 *X*Y* for Correlation -0.98.

The computing speed is also a factor to be considered. The four different operations were done for each of the different discretizations, and results were tabulated and compared.

Table 5.1 Operation evaluation time (seconds) for correlation 0.

Intervals in discretization (X x Y)	addition	Subtraction	multiplication	division	max	Min
16x16	1	1	3	5	1	1
32x32	22	26	154	328	13	11
64x64	3636	3297	52317	148173	1083	866

As this table suggests, for finer discretizations the times for plus, subtraction, max and min form a cluster. Operations for multiplication and division form a more computationally costly cluster. The following figure shows the times for operations: addition, subtraction, max and min.

The next figure shows the times for multiplication and division. The VaR analysis was not complete at the end of this project period.

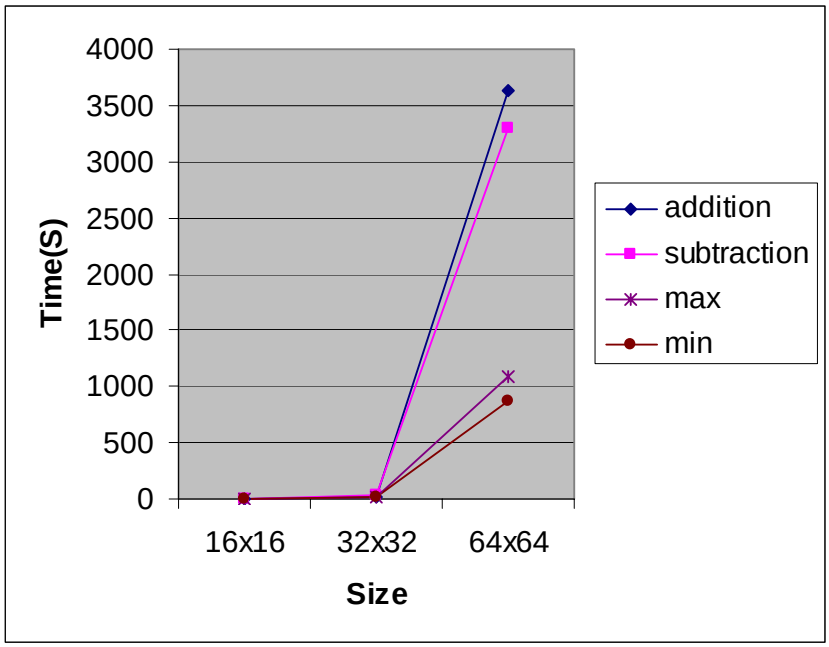

Figure 5-10 Times for Operations.

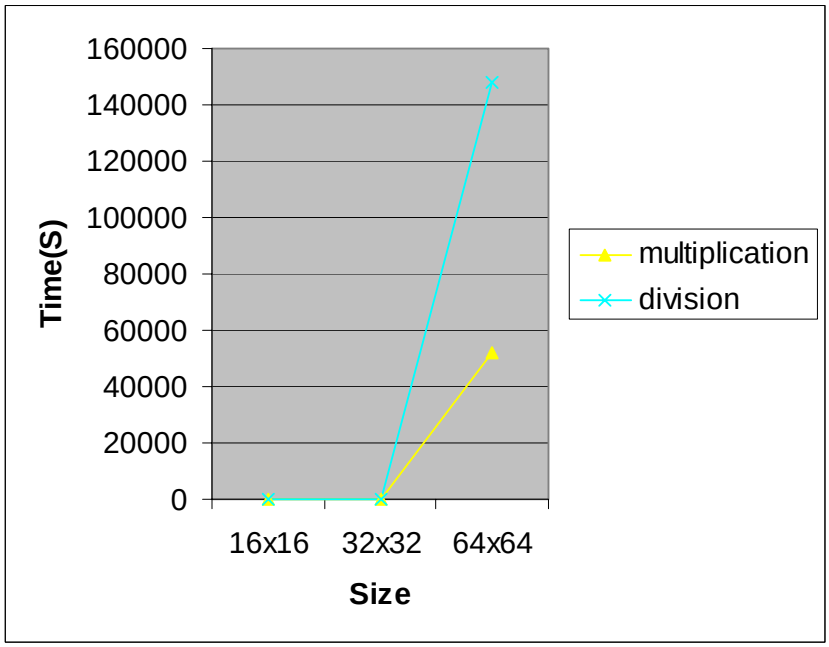

Figure 5-11 Times for Multiplication and Division.

6 Genetic Algorithms for Bidding

There are two generic approaches to the simulation of market dynamics: experimental economics, simulation. The cost and the level of experimental economics is

6.1 Application

Part of the transition from regulation to competition involves setting up systems to treat electricity as a commodity. This includes determination of distribution of electricity from producers to consumers and determination of price paid for the electricity. Currently these factors are partly determined through regional energy exchanges. In these exchanges, auctions determine sellers and price. Such exchanges should be similar to exchanges that trade other commodities such as wheat or gold. However, the physical constraints placed on power plants by network operation give rise to unique considerations. For example, minimum and maximum flow limits are fixed for each branch in the network based on the operating condition of the complete system. Production, transportation, and consumption of electricity all occur in non-trivially different ways than other commodities.

The work of developing electric power market bidding strategies by genetic algorithms is a continuation of the previously described decision-analysis work [Kumar 1996, Richter 1998, Richter 1999]. The bidding strategies were based on two different modifications of a classical data processing structure known as a finite state automaton. The genetic algorithms were varied as well, some incorporating a semi-fixed fitness function, and others using a co-evolutionary (population-specific) fitness function.

Both types of fitness functions maximize profit in a competitive bidding situation. The only feature changed for this project was the choice of competitive bidders against which to play. The auctions that determine profits or losses were played in an iterative fashion. In other words, the same bidder would play the same opponents multiple times in order to allow the bidding agents to learn their opponents' behavior and adjust their own behavior accordingly to maximize profit.

Both representations were tested on variations of experimental parameters. The co-evolutionary setup was a third, very simple, representation run to establish a comparison baseline for the other two representations. The co-evolution setup is the base solution, or the “standard” solution.

Many possible auction scenarios could determine the selection of sellers. This work uses a single-sided auction, but the work can be modified to accommodate a double-sided auction or auctions that are even more complex. The auction research varies in the specific auction design employed. This is due to the disparity in auction rules used in different regions throughout the world as well as to different simplifying assumptions made on the part of this work.

6.2 Market Design

Contreras et al. [Contreras, 2001] compare different implementations of electric market designs. Specifically, they explain the differences between single-round and multi-round auctions in terms of maximization of social welfare, computational cost, resultant market prices, etc. They reach the conclusion that iterative bidding (multi-round auctions) is not advisable for use in day-ahead markets.

Sheblé [Sheblé, 1996] proposes details for the rules governing deregulated electric markets. He defines and describes rules for the interaction of the electric commodity market and its various derivatives markets, such as the futures market, options market, and swap market. He also proposes breaking the trading into periods (e.g., hours, weeks, months) and bidding on production of electricity during these periods. Allocation already determined for a larger period (e.g., monthly) would constrain the possible allocation for smaller periods (e.g., weekly).

6.3 Market Simulation

Otero-Novas et al. [Otero, 2000] discuss the simulation of a wholesale electricity market, called COMSEE, based on Wilson's rules proposed for the power exchange in the Californian market [Wilson, 1997]. There are three basic rules.

- The price cannot be increased.
- The price can be decreased only if the new price is less than the clearing price in the previous iteration by at least a specified price decrement (e.g., $1.00 or $0.10/MWh). This new price is said to “improve” the previous price.
- The price cannot improve any previous clearing price not improved at the first opportunity.

They use these rules to simulate a perfectly competitive market by considering each generator to be an independent entity seeking to maximize personal profit. They also simulate an oligopolistic market in which each firm coordinates the bids of its own units to maximize total profit.

6.4 Optimization

Weber and Overbye [Weber, 1999] modeled the problem of bidding in an electric power market as a two-level optimization problem. The two levels of optimization consist of the problem of determining an optimal bid (first level) under the constraint that the price and dispatch quantity are determined by an optimal power flow (OPF) optimization (second level). This report assumes that the OPF problem is solved outside of the market simulation and embeds the solutions to the OPF into the generators' cost curves.

Song et al. [Song, 1999] optimizes bidding strategies by use of a Markov decision process. The Markov decision process used is similar to the finite state machine representation in use

in this report, but with transitions between states determined stochastically instead of deterministically.

6.5 Genetic Algorithms and Learning

Richter and Sheblé [Richter, 1998] used a genetic algorithm to develop bidding strategies for an electric power double auction. This used a representation based on evolving bid multipliers or a number multiplied by the producer's marginal cost (or the buyer's marginal utility) to get the bid to make.

Richter et al. [Richter, 1999] used genetic algorithms to evolve bidding strategies in a double auction for electric power. They conducted two experiments. One used genetic programming, or the evolution of parse trees, to evolve the bidding strategies. The other approach was to use GP-Automata to represent the bidding strategies.

Wu et al. [Wu, 2002] introduced a machine-learning algorithm to learn to bid in electric power markets. It is general enough to apply to any market, but they tested it specifically on a single-sided auction with discriminatory pricing.

Petrov and Sheblé [Petrov, 2001] used the generic Roth-Erev learning algorithm [Roth, 1995] and [Erev, 1995] to learn effective bidding strategies for a double auction for electric power. They pointed out that the algorithm in its original form was unsuited to learning when profits were at or near zero. They modified the algorithm to account for this deficiency and thereby allowed it to learn much more quickly and efficiently.

Much of economic game theory is concerned with finding Nash Equilibria [Nash, 1950]. A Nash Equilibrium is a set of strategies, one strategy played by each player, with the property that no player can increase its payoff by unilaterally changing its strategy. Given certain assumptions, all players' decisions should converge to that equilibrium. Some of these assumptions, however, are not met in practice: (1) the game must have a unique Nash Equilibrium; (2) all players must have perfect information; and (3) all players behave rationally (i.e., choose the best strategy once they know what it is).

The standard prisoner's dilemma game is a two-player non-zero-sum game. Each player makes a choice to "cooperate" or "defect." If both cooperate, both receive a payoff of three. If both defect, both receive a payoff of one. If one cooperates and the other defects, the cooperator receives a payoff of zero, and the defector receives a payoff of five.

Standard economic theory shows that the standard prisoner's dilemma game has a Nash equilibrium in which both players defect. In other words, if both players have chosen to defect, neither can increase its payoff by changing its strategy. However, in any other state, in which one or both of the players cooperate, a higher payoff can be achieved by changing one's strategy to defect instead. In this respect, the choice "defect" is said to "dominate" the choice "cooperate." Therefore, one would expect any agent playing the prisoner's dilemma to

learn quickly that defecting is the optimal action despite the higher payoff possible if both cooperate.

A variant of the prisoner's dilemma is called the iterated prisoner's dilemma. In this game, two players play multiple rounds of the prisoner's dilemma against each other. Axelrod evolved finite state machines to play the iterated prisoner's dilemma [Axelrod, 1987]. He found that when given the opportunity to play multiple rounds, agents would often learn to cooperate with one another in order to get the higher payoff that results when both players cooperate. However, they require a more complex strategy than simply “always cooperate” lest they be taken advantage of by more malicious players. Thus, the choice to encode strategies as finite state automata rather than simply as a single action to take was made.

This experiment has relevance to the current work in two respects. First, the market being simulated is a complex, multi-round economic game. Second, finite state automata are being used to encode the strategies for playing this game. It is important that subsequent rounds of play be against the same opponents in order for the strategies to have any meaning. Otherwise, there is no such concept as reaction to an opponent's actions because the opponent will not be the same the next time one meets him.

It is of interest to determine which prisoner's dilemma strategies are evolutionarily stable. An evolutionarily stable strategy is one that, given that it is in use across the entire population, cannot be “invaded” by another strategy. To be invaded is to be beaten consistently to be driven out of the population by virtue having a lower fitness than the invading strategy. It has been shown that no pure (deterministic) strategy is completely evolutionarily stable [Boyd, 1987]. However, many strategies, such as Tit-For-Tat (TFT), are stable against almost all invading strategies, and the strategies that successfully invade TFT are themselves very unstable.

Wagner et al. [Wagner, 2000] has also studied the iterated prisoner’s dilemma. They analyzed the effect of strategy representation on the evolution of cooperation in playing iterated prisoner's dilemma, investigating finite state machines, plain logical formulas, logical formulas with a time delay operation, If-Skip-Action (ISAc) lists, Markov chains, and neural networks.

Mayfield and Ashlock [Mayfield 0998] discovered a non-trivial effect of evolution. FSM's were evolved for 1,000 generations, more than long enough for fitness to converge, and the population was saved. The FSM's were then evolved for an additional 9,000 generations, with no apparent change in fitness. However, when the FSMs from generation 10,000 were played against the FSM's from generation 1,000, the former achieved much higher fitness than the latter. This effect, in which a population evolves for a very long time with no apparent change in fitness but is able to build up general skills in achieving high fitness, has casually been called the Mayfield Effect.

Ashlock [Ashlock, 1997] evolved GP-Automata to play a simple game known as “Divide the Dollar.” In this game, two players bid a monetary amount. If the sum is less than or equal to a

dollar, they each receive their bid amount; otherwise, they each receive zero. This is an interesting game to study theoretically because it has a continuum of Nash Equilibria corresponding to real number solutions to the equation a + b = 1, where a is the first player's bid and b is the second player's bid. This means there are an infinite number of Nash Equilibria, so predicting how the game will end up being played is difficult theoretically. It turns out that GP-Automata converges to a population in which every player bid just under $0.50.

Leahy and Ashlock [Leahy, 2000] conducted a study similar to Wagner [Wagner, 2000]. In this work, they investigated the effects of representation choice on evolving agents to play the Divide the Dollar game. They used artificial neural nets, lookup tables, real valued mathematical formulae, integer valued formulae modulo 101, and GP-Automata to represent strategies.

Ashlock [Ashlock, 2001] studied a simple evolutionary algorithm that played a population against itself in the “Public Investment Game.” In this game, multiple bidders submit a sealed bid between $0 and $100. The sum total that they invested is then doubled and distributed evenly among the bidders. This game attempts to model publicly-funded utilities, such as roads requiring maintenance, in which everyone gets nothing if no one contributes, but the total effect on the group is negligible if only a single bidder lowers its contribution. This game's Nash Equilibrium consists of every player making a bid of zero, and that is exactly the result that Ashlock observed after a few generations of evolution. He was able to achieve higher payoffs by introducing “laws” (minimum required investment levels) and “fines” (penalties subtracted from payoffs) for evading the laws.

7 Methods and Procedures

7.1 Electric Power Markets

Electric power prices in the marketplace are determined for the most part by single-sided auctions. We make the following assumptions about the structure of the market. Generation companies (GENCOs) produce power and consumers buy power for consumption. Typically, energy service companies (EMCOs) consolidate demand from a group of consumers so that GENCOs do not deal with individual electricity consumers. Transmission companies (TRANSCOs) own the power lines and are paid to transport electricity from one physical location to another. Ancillary companies (ANCILCOs) provide ancillary services. An Energy Mercantile Association (EMA) serves as a buffer between producers and consumers, or a market maker.

The value chain is different from the physical chain. The physical chain refers the actual transportation of power from the producer to the final consumer. The value chain refers to the flow of money that is paid for this power, which involves extra entities that do not themselves necessarily consume, transport or produce power.

The EMA holds a single-sided auction on either side. It will estimate demand from consumers and hold an auction taking bids from generators to sell their electricity. It will then take the electricity it bought, turn around, and hold an auction taking bids from consumers who buy the electricity. This report models the former auction in which sellers, assumed to consist of competing GENCOs, make bids to sell power and must then produce the amount they agreed to sell. The simulations could just as easily be applied to a buyer's auction.

The purpose of an auction is to expose information about buyers' and sellers' willingness to pay or sell. Commodities like electricity often have no explicit fixed worth; their worth is a function of the current market conditions. An auction attempts to find this worth [McAfee, 1987]. Essentially, an auction allows discovery of the equilibrium price defined as the intersection of the demand and supply curves of the buyers and sellers, respectively. In this case, the demand curve will be a vertical line as the EMA will estimate demand as a single fixed quantity.

According to economic theory for competitive markets, in the long-run all profits should go to zero as sellers underbid each other in the marketplace. Any number of factors that appear in a real marketplace disrupts this prediction, however, such as cost curves that do not pass through the origin, capacity limits on generation quantity due to physical limitations of generators, and irrational behavior on the part of market participants. The most disruptive factor that leads to violation of theoretical predictions is information uncertainty on the part of market participants.

7.2 Evolutionary and Genetic Algorithms

A genetic algorithm is a type of evolutionary algorithm. An evolutionary algorithm is an algorithm that uses the biological paradigm of evolution to solve mathematical problems. Many researchers in the field will disagree on definitions and terms. Part of this is due to confusion in the use of borrowed biological terms whose original definitions are somewhat obscured by their use in evolutionary algorithm descriptions.

An evolutionary algorithm is any algorithm that implements the following pseudo-code.

Basic Evolutionary Algorithm		
1		Create an initial population of potential solutions
2		Evaluate the fitness of the population
3		Repeat … until done
	A	Select pairs from the population to be parents, with a fitness bias
	B	Copy the parents to make children
	C	Perform crossover on the children (optional)
	D	Mutate the resulting children
	E	Place the new structures in the population
	F	Evaluate the fitness of the new structures

Figure 7-1 Basic Evolutionary Algorithm Loop.

Each iteration of this loop is called a generation. A solution (also called a creature or agent) in Figure 7-1 refers to some reasonable ("reasonable" being defined by the problem at hand) encoding of potential solutions to a problem. A subset of evolutionary algorithms, known as genetic algorithms, is the most prevalent and is the type used in the simulations described in this report.

These algorithms always execute the basic evolutionary algorithm loop, which performs crossover. Crossover is the process of exchanging subsets of representations between two solutions. This mimics the process of sexual reproduction, as two parent solutions are copied and their children are crossed over. Often genetic algorithms are defined to operate on fixed data structures. This is primarily to distinguish them from another type of independently developed evolutionary algorithm known as genetic programming [Koza, 1992]. Genetic programming uses the loop shown in the above figure to evolve parse trees (a variable-sized representation) to solve problems. However, it is not possible to evolve data structures which are parse trees nor have a fixed size. Such an algorithm will also be termed a genetic algorithm in this report.

In addition to crossover, the solution space is explored using mutation, the process of randomly perturbing solutions. This mimics the biological process of random genetic mutation in creatures of a living population, which occasionally give the creature an advantage over others.

A fitness function that numerically evaluates the optimality of a solution must also be defined. Whatever scheme is used to select the parents, it always biases selection toward those creatures with higher fitness. It may also bias replacement of creatures toward those that are less fit. This mimics the biological process of natural selection in which more fit creatures are more likely to survive and pass on their genes to offspring, while less fit creatures die off.

7.3 Representation

To develop solutions to a problem using a genetic algorithm, the potential solutions must be encodable in such a way that allows mutation, crossover, and fitness evaluation. Two different, but related, encodings were used to represent bidders in an auction. Each one is a generalization of an information processing structure known as a finite state machine (FSM) or finite state automaton (FSA). The first extension is known as a GP-Automaton and the second is known as a Neural-Automaton. GP-Automata were first introduced by Ashlock [Ashlock, 1997]. Neural-Automata are introduced here for reasons explained in the description of Neural-Automata.

A finite state machine is a theoretical data processing structure. It takes input from some external source, changes its internal state in response to the input, and may or may not produce an output, known as a response. Mathematically, it is a directed graph, with each node called a state. Each state has associated with it a transition table. The transition table enumerates all possible inputs and states, for each input, what response to give and to what state to transition. Each directed edge in the graph, therefore, represents a transition from one state to another, and each edge's satellite data consists of an input that triggered the transition and a response to output. The FSM receives inputs sequentially, each time transitioning to a new state and outputting a response. Each transition has only one response in the model used in this report, though in theory there could be more than one transition between two states, each with a different input and response. An FSM also has an initial state and an initial response, since it must start somewhere before it receives its first input. All computer programs may be thought of as finite state machines with memory.

Finite state machines are good structures to evolve because the division of functionality into discrete states and transitions allow for natural choices of crossover operators. Finite state machines have been evolved to play simple economic games such as the Iterated Prisoner's Dilemma [Axelrod, 1987] and as control structures for virtual robots [Ashlock, 2000].

Because all possible inputs must be enumerated, FSM's are impractical to use on problems that have a large number of inputs. Even a single unbounded input renders the number of possible inputs infinite. Since the actual amount of information carried in the input tends actually to be much smaller, bandwidth compression is performed on the data. Bandwidth compression is a term borrowed from communications. It refers to reducing the resources needed to represent some data. In this case, the bandwidth of the input data needs to be compressed to the set of integers in the range [0, number of states - 1], since this is the maximum number of possible transitions possible from any state.

Some method must be used to map all possible inputs into an integer in the range [0, number of states - 1]. GP-Automata and Neural-Automata provide two different methods of accomplishing this bandwidth compression.

7.3.1 GP-Automata

GP-Automata (GPA) have been described as a combination of genetic algorithms and genetic programming (hence the "GP" for genetic programming). Each state in a GPA replaces the transition table with a parse tree whose input nodes are the external inputs to the FSM. When a GPA iterates (i.e., takes an input and decides what to do), it runs the input through the parse tree at the current state. The parse tree outputs an integer and the parity of this integer is used to determine the next state transition. Therefore, only two distinct transitions may be made from any state.

During evolution, the parse trees are modified according to the standard mutation and crossover operators used on parse trees in genetic programming. The exact details are given in Doty [Doty, 2003].

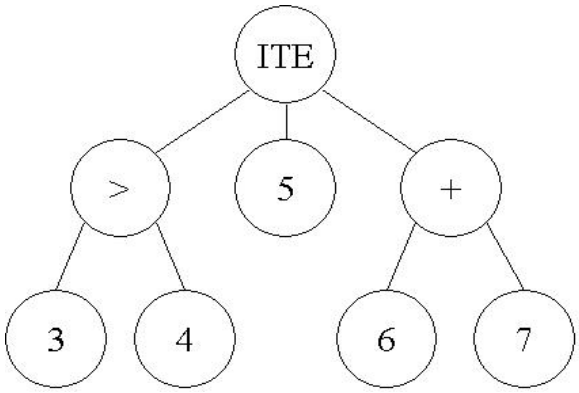

Figure 7-2 Example Parse Tree.

The parse trees are shown in a LISP-like notation. Figure 7-2 shows a parse tree in a graphical notation. The tree in this figure would output the value 13. ITE is if-then-else and would evaluate the subtree on the left to false because three is not greater than four. If it were true, it would output the middle child node, 5. However, since it is false, ITE outputs the left child node, 6 + 7 = 13.

7.3.2 Neural-Automata

While the GP-Automata scheme has the advantage of being adaptable to an arbitrary number of states, it has the disadvantage that each state may contain at most two next-state transitions. Such an FSM is strictly less expressive than an FSM that allows arbitrary transitions from any state to any other state. A proof of this follows.

Suppose we wish to design an FSM that takes nickels, dimes or quarters as input and outputs a “1” if at least 15 cents have been entered cumulatively and outputs a “0” otherwise. The initial state would correspond to 0 cents being entered so far. After the first coin, 5, 10, or 25 cents will be the total. Since 25>15, but 5<15 and 10<15, the next state transition corresponding to a quarter being entered must be different from the transition resulting from the other two coins, since a quarter should cause a response of “1,” and the other two should cause a response of “0.” Since we allow only two transitions per state, if either a nickel or a dime is entered, the FSM must end up in the same state regardless of which was entered. If a dime was entered, then no matter what coin is input the next time, the cumulative total will be equal to or exceed 15 cents, and the FSM must therefore output a “1.” However, if a nickel was entered the first time, then the total may or may not exceed 15 cents, depending on the second coin entered. Since the first coin sends the FSM to the same state whether the input was a nickel or a dime, there is a contradiction. To handle the second coin, three possible states must exist to go to from either the initial state or the second state. Limiting the number of next state transitions to two renders the FSM incapable of executing this function. However, an FSM with no limit on transitions could easily execute this function. Therefore, an FSM with only two next state transitions per state is strictly less expressive than a general FSM.

Neural-automata filter the input data while allowing an arbitrary number of next-state transitions. Instead of a parse tree, a neural-automaton uses a feed-forward neural net at each state to decide the next state transition. The output node of the neural net uses a sigmoid transfer function $\frac{1}{1+e^{-x}}$ where x is the weighted sum of the inputs. The output of this function is bounded in the range [0, 1]. Therefore, any number in the range [0, number of states - 1] can be acquired by multiplying the output of the neural net by the number of states and truncating the result. Furthermore, since a feed-forward neural net with two hidden layers is capable of approximating any mathematical function [Cybenko, 1988], it is capable of approximating the expressive power of a parse tree.

7.4 Market Setup

Varieties of experiments were performed in the project. In each case, where applicable, three different representations were evolved. GP-Automata and Neural-Automata were each evolved with a genetic algorithm. The third representation was simply an ordered pair of real numbers (p, q) which represented a constant bid to make. The reason for introducing this representation is discussed shortly.

The market consisted of an auctioneer, representing the EMA, and any number of bidders, representing the competing generation companies. The next section describes the auction proceeds.

7.4.1 Auction Process

The auction process is shown in the following figure. The auctioneer announces a demand to be met. Bidders each submit a bid, which is an ordered pair (p, q) representing a price, p, and a quantity, q. Market clearing then follows. These bids are sorted in ascending order by price. They are then accepted sequentially, adding the quantity of each bid to a running total until this total meets the demand. At this point, no more bids are accepted and the last one is accepted for only the amount of demand that remained, not for the total quantity given in the bid.

After market clearing, price discovery is tested. The condition used for price discovery in these experiments was whether at least 50 % of the bids were accepted. If price discovery did not occur, new bids are taken and the market is cleared again. The results of the previous bids and market clearing are forgotten and do not affect profits. One round of bid submission and market clearing is termed a cycle. If after ten cycles, price discovery still has not occurred, typically the results of the last market clearing would be accepted. In these experiments, however, the profits of all bidders were simply zeroed to introduce selective pressure to make bids conducive to price discovery.

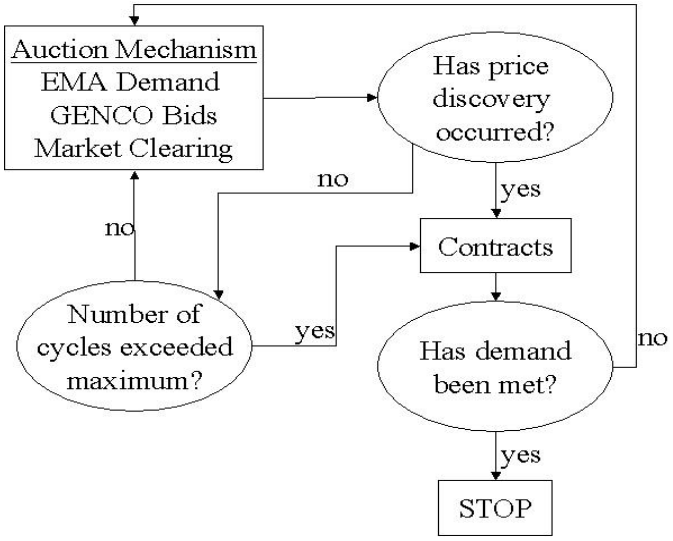

Figure 7-3 Auction Process.

Once price discovery occurs, the bids are committed and contracts are written. Now any bidder whose bid was accepted is obligated to provide the quantity of electricity accepted. If the total quantity accepted did not meet demand, then another auction is held with demand revised to be the previous demand minus the quantity accepted in the last auction. After ten auctions, if demand has not been met, then no more auctions are held.

7.4.2 Price and Cost Determination

The actual revenue to each bidder is the price the bidder is paid times the bid quantity accepted. There are two methods implemented in determining the price each bidder is paid. The first is simply to pay the bidder the price it gave on its bid, known as discriminatory pricing. The second is to give it the market-clearing price, defined as the price submitted by the last bidder whose bid was accepted. This is known as uniform pricing. The reasoning behind this is that the uniform market-clearing price is the equilibrium price or the price at which the supply curve crosses the demand curve.

Cost determination is modeled by a quadratic function of quantity, a common scheme for approximating the cost of producing electricity [Wood, 1996]. Each generation company may own more than one generator. However, once a generation company has a quantity it has committed to deliver, there are constrained optimization techniques, such as the use of LaGrangian multipliers that can be used to find the optimal power production from each generator. That optimization is not explicitly performed in this simulation, and the total cost to a generation company to produce a given quantity is modeled as a single quadratic function of quantity. This is a simplifying assumption. In a real situation, any number of optimization techniques such as LaGrangian Relaxation, could be used to find the actual optimal power production for each unit. These optimizations would take into account the fact that a power plant may operate more than one unit, and that these units not only produce electricity at different costs, but also are connected to transmission system that has losses dependent on the location of the generating unit and that has flow limits.

In this work, each power producer was treated as if it had a single unit that had a single quadratic cost curve. This differs from previous work done in this area in which the optimization was performed explicitly.

The total profit made by a seller after an auction is {(price paid) * (quantity delivered) - (cost to deliver quantity)}. If more than one auction is held, total profit is the sum of the profits from each auction. All the auctions needed to meet demand are termed one round of auctions. In the genetic algorithms, 24 rounds of auctions would be held against the same opponents for fitness evaluation, summing the total profit from each auction round. The number 24 was chosen to correspond with the 24 hourly auctions held in a single day. In the next generation of agents determined by the algorithm, different opponents would be played, but the same opponents would always be played over the course of the 24 rounds of auctions.

The auctions being simulated in this work are representing day-ahead (forward) markets for electricity. This is why the auctioneer must estimate demand. The auction is held to determine generation allocations for each hour of the following day. In reality, an hourly auction would be held the next day in order to cover the difference between the estimated demand (which this forward market is designed to cover) and the actual demand (which may end up being different from the predicted value).

7.5 Experiments

7.5.1 Variable Parameters

There were a number of choices to be made concerning the implementation of a genetic algorithm simulating this market. Varying these choices led to the different experiments reported here.

7.5.1.1 Representations

Two different representations were evolved using a genetic algorithm, GP-Automata and Neural-Automata. In the previous discussion of FSMs, it was stated that they produce a response, but exactly what this response consists of was not covered. In these experiments, the response was an ordered pair (p, q), representing a bid price and bid quantity to submit. At the beginning of the 24 rounds of bidding, the FSMs would be reset (i.e., internal state set to initial state and initial response taken as first bid). After that, each bidder would update its internal state and output the appropriate response in each subsequent auction. Note that because more than one cycle per auction may occur to achieve price discovery, and more than one auction may occur to meet demand for one round, there may be more than 24 bids taken from each bidder in one fitness evaluation.

The inputs fed to each FSM, after the first bid, were:

- Previous high bid
- Previous low bid
- Previous average bid
- Bidder's own previous bid
- Previous number of bids accepted
- Demand remaining to be met
- Quantity that the bidder has agreed to deliver so far.

For each argument that is a bid, the FSM was actually fed two real numbers: the price and the quantity of the bid. “High” bid and “low” bid refer to the bid with the highest and lowest price, respectively, not the highest and lowest quantity. The “demand remaining to be met” refers to residual demand left over if an auction is held, all bids were accepted, and demand was not met. Since another auction will be held, agents need to know the new demand.

Additionally, a third representation was “evolved.” This was simply an ordered pair, (p, q), which represented a constant bid. However, crossover was not performed, and so this was not really a genetic algorithm. The bid was simply mutated every generation by adding Gaussian noise to each of the numbers in the bid.

The algorithm to develop these constant bids was therefore more of a population-based stochastic search algorithm. This is an algorithm that starts with a population of initial random solutions and perturbs each slightly with random noise, keeping and copying more

optimal solutions over those who were less successful. The population was initialized with random bids, and at each “generation” the most-fit half of the population replaced the least fit half of the population. Each newly copied bid was then perturbed by Gaussian noise.

The reason for using this representation was two-fold: to test the dynamics of the market and to reject the null hypothesis that evolution of more complex FSM-based strategies is no better than random guessing. Initially, when debugging the simulated marketplace, we needed to see if it would match the predictions of economic theory under perfect conditions and how exactly the market would be affected by imperfect conditions. This representation should conform to theoretical predictions that say which single bid is optimal given bidders who attempt to maximize profit. Deviance from theoretical predictions would then point to a flaw in the implementation of the simulated marketplace or a theoretical assumption not being upheld. This aids in understanding the behavior of the more complex strategies in the same simulated environment.

7.5.1.2 Co-Evolution vs. Fixed Fitness vs. Immortality

In general, fitness functions in evolutionary algorithms can be divided into two classes: co-evolutionary and fixed. A fixed fitness function evaluating one member of a population is independent of the other members of the population. A co-evolutionary fitness function evaluating the same member will give different results depending on the other members of the population. For example, a genetic algorithm evolving strategies to play chess might evaluate fitness by having the agents play an expert alpha-beta strategy such as the one programmed into Deep Blue, IBM's grand champion chess-playing computer. This would be a fixed fitness function. The algorithm might alternately evaluate fitness by simply playing the strategies against each other in a round-robin tournament and averaging the scores. This would be a co-evolutionary fitness function.

The marketplace simulated has agents that are not identical to one another, the difference being that they have different production cost curves. Therefore, one is not as concerned with how an agent might perform against another agent like it, but how it would perform in a marketplace with agents different from it. The co-evolutionary strategy used therefore requires a bit of a re-definition (which actually brings it closer to the original definition in biology). Instead of evolving one population of agents, one population for each of the number of bidders in the market is evolved. When breeding is done each generation, members from a population are bred only with members from the same population. When fitness evaluation happens, however, the fitness of a member of a population depends only on members from the other populations because the agent bids only against one member from each of the other populations. This is actually closer to the biological notion of co-evolution in which two species are said to co-evolve with one another if “a change in one species acts as a new selective force on another species, and counter adaptation by the second species, in turn, affects selection on individuals in the first” [Campbell, 1987].

A method of introducing a fixed fitness function when no external one exists is possible in cases where the agents being evolved play against each other.

- First, evolve using co-evolution for a certain number of generations.
- Then pick the best members of the population and "immortalize" (save) them.

The fitness function now becomes the profits attained by playing against the immortal strategies, instead of playing against each other. A ratcheted immortalizing fitness function can be used as well. This involves immortalizing the best members of the population every n generations. Fitness is determined by playing all the sets of agents that have been immortalized so far. To account for space considerations, a shortcut may be used, such as "play the last 5 immortalized sets of agents." For example, if n=1,000, and we are on generation 8,500, the fitness would be determined by playing the best agents from generations 4000, 5000, 6000, 7000, and 8000.

This sounds similar to elitism, but this is a distinct concept. An elite genetic algorithm is one in which at least one member of the current population is guaranteed to be a part of the population in the next generation. In other words, there is no way for the children of breeding to replace all the members of the population. However, this elite group differs from the immortal group introduced here in a number of ways. First, the immortal group is not considered when breeding. Only non-immortal members of the population can breed to create children. Secondly, the immortal group is the fitness function. An elite group in a normal genetic algorithm has no special role in the fitness function. Finally, the immortal group is the same from one generation to the next, unless explicitly replaced. The elite "group" in the normal genetic algorithm, however, is defined by the current population. Usually an elite member is one with the highest fitness. However, it may not have the highest fitness during subsequent generations.

7.5.1.3 Uniform vs. Discriminatory Pricing

Under uniform pricing, all "winners" (i.e., bidders whose bids were accepted) receive the same price. In the simplest case, this amount is the market-clearing price, the highest price bid that was accepted. The theoretical justification for this is that the highest accepted price in an auction is the price at which the supply curve would intersect the demand curve. This is the equilibrium price, which is the price that all trades should be made in a perfectly competitive market.

Under discriminatory pricing, winners receive different prices. In the simplest case, this amount is the price they bid. In any auction where there is more than one winner, this would allow the auctioneer to retain more money at the expense of the sellers. The sellers who are aware that their bids are lower than the highest accepted bid (market-clearing price) would raise their bids close to that expected value. Thus, it would tend to inflate the bids above the bidders' marginal costs, which may actually lead to less profit for the auctioneer in the long run.

7.5.1.4 Number of Bidders

In a market with few bidders, each bidder has more market power. Bidders with market power have enough influence to affect the outcome of the auction. In a uniform pricing

scheme, one would say they have the power to change the price. In the context of an auction, the immediate effect of having few bidders is that it becomes easy for bidders to cooperate with each other and raise prices by raising their bids. Alternately, a bidder could withhold capacity to drive the price up and undercut it later.

7.5.1.5 Cost Curves and Capacity Limits

Different producers will have different generation cost curves. This models the real world fact that some electricity producers are coal-driven, some are oil-driven, some are nuclear, some have many generators, and some may only have one. They also have different capacity limits, or minimum and maximum quantities that they are able to produce. This models the physical limitations of power plants. The fact that different bidders have different cost curves and capacity limits is the motivation behind separating the genetic algorithm into separate populations. Each population represents a different type of producer and is evolving to get better at bidding under the constraint of its own cost curve and capacity limits. Previous studies in this area assumed that all power producers had similar cost curves and capacity limits.

7.5.2 Genetic Algorithm Parameters

There are many generic parameters of genetic algorithms. Some, such as mutation and crossover operators, significantly affect the outcome of the algorithm. Some, such as population size and number of generations, are modified according to the problem at hand, usually by making them as big as possible while allowing for the program to finish running in a reasonable amount of time.

Unless noted otherwise, these experiments used a population size of 32 and ran for 1,000 generations. The model of evolution (i.e., method of breeding agents with high fitness), used for the GP-Automata and Neural-Automata, is known as single tournament selection. Figure 7-5 shows how a single tournament selection works.

Creatures before Breeding

Creature	Fitness
Creature 1	6
Creature 2	5
Creature 3	7
Creature 4	2
Creature 5	7
Creature 6	1
Creature 7	2
Creature 8	3

Creatures after Breeding

Creature	Fitness
Creature 1	6
Child of 1 and 3	?
Creature 3	7
Child of 1 and 3	?
Creature 5	7
Child of 5 and 8	?
Child of 5 and 8	?
Creature 8	3

Figure 7-4 Single Tournament Selection of Size Four.

In this model of evolution, the population is divided into sets of four agents. Within each set of four, the two most fit are picked as parents. They are then bred (i.e., copied, crossed over, and then mutated) and their children replace the two least fit. Since the fixed-bid representation did not use a genetic algorithm, in each "generation," the most fit half of the population would replace the least fit half, and then the newly copied agents would be "mutated."

The mutation and crossover operators depended on the representation. For both of the FSM-based representations, there were a number of possible mutation and crossover operators that could be chosen. Each was assigned a probability and picked to be the mutation or crossover operator with that probability. The mutation rate was a number between 0 and 1 that correlated with the severity of mutation. If the number of possible mutation operators was given by m, and the mutation rate given by r, then m * r mutations would be performed on each child in each generation. Each mutation, the probability of picking any of the individual mutation operators remained the same.

7.5.2.1 Fixed Bid Mutation

Since the fixed bid representation did not use a genetic algorithm, there is no significant amount of crossover. However, it randomly perturbed the bid between each fitness evaluation, like a mutation. Both the price and quantity were perturbed by Gaussian noise with standard deviation equal to {maximum – minimum) / 10}. As described previously, the minimum and maximum prices were 0 and 120, respectively, and the minimum and maximum quantity depended on the capacity limits of the producer.

7.5.2.2 GP-Automata Mutation

The GP-Automata mutation operator selected one of the mutation operators listed in Table 7.1. A state (i.e., initial state or state transition) was changed by randomly selecting a new state uniformly from the set of states. A response was perturbed by adding Gaussian noise with standard deviation {maximum – minimum) / 10}. Since a response consisted of both a price and a quantity, both of these were perturbed. In the case that a state transition or response was changed, the particular transition edge to mutate was selected uniformly from all the state transitions. A new parse tree was created by randomly generating nodes until the tree was of size six (had six nodes). Crossover between parse trees involved randomly selecting a node in each tree and exchanging the subtrees rooted at those nodes. The trees on which to perform crossover were randomly selected uniformly from all the decider trees.

Table 7.1 GP-Automata Mutation Operators

Probability	Mutation
0.1	Change initial state
0.1	Perturb initial response
0.2	Change a state transition
0.2	Change a response
0.1	New decider parse tree
0.1	Crossover two decider parse trees
0.1	Exchange two decider parse trees
0.1	Copy one decider parse tree over another

7.5.2.3 Neural-Automata Mutation

The Neural-Automata mutation operator selected one of the mutation operators listed in the following table. These mutation operators follow the same rules as those of GP-Automata. Crossover between neural nets involved randomly selecting the indices of two edges within the neural nets and exchanging the values of the edges in between them.

7.5.2.4 GP-Automata Crossover

The GP-Automata mutation operator selected one of the crossover operators listed in the following table. If the "exchange states" operator was selected, two random indices were selected, and all states in between these indices were exchanged.

7.5.2.5 Neural-Automata Crossover

The Neural-Automata mutation operator selected one of the crossover operators listed in Table 7.2.

Table 7.2 Neural-Automata Crossover Operators

Probability	Mutation
0.2	Exchange initial states
0.2	Exchange initial responses
0.6	Exchange states

If the "exchange states" operator was selected, two random indices were selected, and all states in between these indices were exchanged.

7.5.2.6 Fixed-Bid Initialization

The two numbers constituting the fixed bids were initialized to a random number distributed uniformly over the possible range of each number. Price varied from 0 to 120, and quantity depended on the capacity limits of the generator.

7.5.2.7 GP-Automata Initialization

The FSM was initialized with six states. The initial response was initialized in the same manner as the fixed-bid representation. Each transition next state and response was initialized in the same manner as the initial state and transition. The decider parse tree was initialized to a random parse tree with six nodes. A sample parse tree is shown in Doty [Doty, 2003]. The nodes could be any of those shown in Table 7.3. These nodes were inserted at random until a tree of size six was obtained.

7.5.2.8 Neural-Automata Initialization

The FSM was initialized in the same manner as the GP-Automata. The decider neural net was initialized to a random feed-forward neural net with two hidden layers. Each hidden layer had 3 nodes. The weights were initialized to random values distributed uniformly in the range [-1, 1].

Table 7.3 Parse Tree Nodes

Node	Name	Return Type	Args	Returns
ITE	if-then-else	args 2 and 3	3	arg 2 if arg 1 is true; arg3 3 otherwise
Odd	Odd	Boolean	1	true if arg1 is odd, false otherwise
Max	Maximum	Real	2	maximum of args 1 and 2
Min	Minimum	Real	2	minimum of args 1 and 2
~	Negation	Real	2	negation of arg 1
Com	Complement	Real	2	1 - arg 1
>	Greater than	Boolean	2	true if arg 1 > arg2; false otherwise
>=	Greater than or equal to	Boolean	2	true if arg 1 >= arg2; false otherwise
<	Less than	Boolean	2	true if arg 1 < arg2; false otherwise
<=	Less than or equal to	Boolean	2	true if arg 1 <= arg2; false otherwise
+	Add	Real	2	arg 1 + arg 2
-	Subtract	Real	2	arg 1 - arg 2

8 Genetic Algorithm and Market Experiments Results

The experiments performed were based on changing the variable parameters discussed in the previous chapter. All three of the representations were run and compared in the cases of co-evolution. Only the finite state automata were compared in the cases of evolution by periodic immortalization.

Each experiment also shows the average bid and average fitness of the whole population versus generation of evolution. In most cases, a single run the algorithm is shown in addition to an average of many runs of the algorithm to demonstrate general behavior.

The experiments were divided into two general cases: evolution by co-evolution and evolution by periodic immortalization. Co-evolution has the potential to develop good strategies, but since fitness will not necessarily prove to be an effective measure of improvement of the strategies. Evolution by periodic immortalization, however, evolves the bidders against a set of fixed strategies after the first 1,000 generations. Therefore, if the bidders are learning better bidding strategies, we should see the average fitness of a population increase with this evolutionary scheme.

Unless stated otherwise, all experiments used a mutation rate of 0.5. Each figure shows the average fitness ("Fitness"), average bid price ("Bid"), average committed bid price ("Committed Bid"), and equilibrium price ("Equilibrium Price"), each graphed against generation of evolution. "Committed Bid" does not count bids that were in cycles that failed to meet the condition of price discovery. "Equilibrium Price" is the price or the last bid that was accepted. Only a sampling of the figures is in this report. All of the figures are in Doty [Doty, 2003].

8.1 Co-Evolution

The experiments described in this section developed the bidders through co-evolution. These experiments tested all three representations: GP-Automata, Neural-Automata and Fixed-Bid. The Fixed-Bid equilibrium behavior in each case should give a reasonable approximation to expected theoretical behavior. In many cases, it gives an equilibrium price higher than expected until one takes into account the effect of market power on price.

8.1.1 Experiment Set 1: Co-Evolutionary Fitness Function, Discriminatory Price

This experiment used discriminatory pricing. As noted previously, with discriminatory pricing, each bidder, if it has a bid accepted, is paid the price it listed in the bid. Variations on the cost curves, capacity limits and number of bidders were explored. The different trials are shown in Table 8.1. In Figure 8.1, there are the three figures for each of the graphed results of a particular representation (GP-Automata, Neural-Automata, Fixed-Bid). Additional data and all figures are in Doty [Doty, 2000]. Such data includes "Cost Curve/Min/Max" that refers to the cost curves, and upper and lower capacity limits, respectively. In all cases, either a single cost curve and capacity limits were used for all bidders, or the population was split

into two groups, each of which had its own set of cost curves and capacity limits. In these cases, if there were ten bidders, seven of them would have Cost Curve 1 and Min/Max 1 capacity limits, and the other three would have Cost Curve 2 and Min/Max 2 capacity limits. If there were four bidders, two of them would have Cost Curve 1 and Min/Max 1 capacity limits, and the other two would have Cost Curve 2 and Min/Max 2 capacity limits. In the cases that every bidder had the same cost curve, these quantities are equal.

Table 8-1 Variations on Experiment Set 1

Variation	Bidders	Demand
A	10	2000
B	10	2000
C	10	2000
D	10	2000
E	4	800
F	4	800
G	4	800
H	4	800

Demand was set to be proportional to the number of bidders and to the average capacity limits of the bidders. The demand listed is the average demand, d, generated for each auction. The actual demand for each auction was drawn from a uniform probability distribution in the range [d – d/10, d + d/10].

Figure 8-1 show four different cost curve and capacity limit choices for all three representations using the co-evolutionary fitness function and discriminatory pricing, with an auction size of ten bidders. Examining the fixed-bid representation as a guide to the expected behavior of a market, a few general trends are evident. These trends are in place in the other two representations as well, although they appear amid the more complex behavior that the FSM's display.

Not surprisingly, when the bidders have different cost curves, those with a higher cost curve get a lower fitness. Higher capacity limits lead to a lower equilibrium price. This is due to the demand (which is independent of the capacity limits of the bidders) being more easily met when the capacity limits are higher. Therefore, more competition is present, driving the price down.

The GP-Automata and Neural-Automata consistently achieved a higher equilibrium price than the fixed-bid representation. All experiments achieved equilibrium just as quickly as the first experiment.

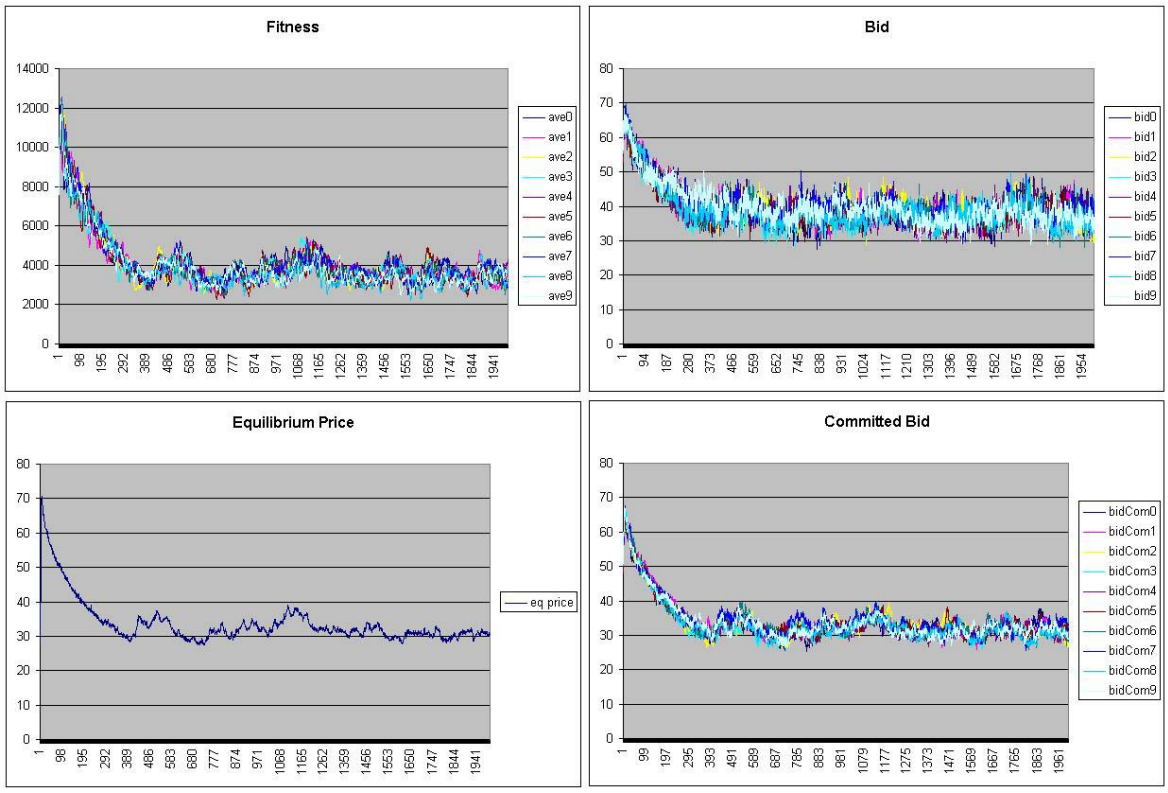

Figure 8-2 Co-Evolutionary Fitness Function and Associated Data.

The effects noted above are present when there are four bidders as well. The average equilibrium prices achieved are higher than those achieved with ten bidders. This illustrates the effect of the number of market participants on the market power of each individual market participant; less participants gives each participant more market power.

8.1.2 Experiment Set 2: Co-Evolutionary Fitness Function, Uniform Price

This experiment used uniform pricing. As noted previously, with uniform pricing, each bidder, if it has a bid accepted, is paid the equilibrium price, or the price of the highest accepted bid. The rest of the details are identical to those described in the previous section.

Figure 8-2 shows one of the four different cost curve and capacity limit choices for all three representations using the co-evolutionary fitness function and uniform pricing, with an auction size of ten bidders.

Comparing these results to those obtained with the same conditions except for using discriminatory pricing, we see that on average, the equilibrium price and bids are lower under a uniform pricing scheme. This makes intuitive sense economically. When one's own bid price determines whether the bidder wins or not, and the bidder's payoff varies with the size of the bid, then there is a tradeoff. A lower bid, for instance, has a better chance of winning, but results in a lower payoff. However, under uniform pricing, one's own bid determines

whether one wins or not, but has no effect on the bidder's payoff (unless it was the last accepted bid). Therefore, the pressure to bid higher for a higher payoff is removed, and only the pressure to bid lower for a better chance of winning remains.

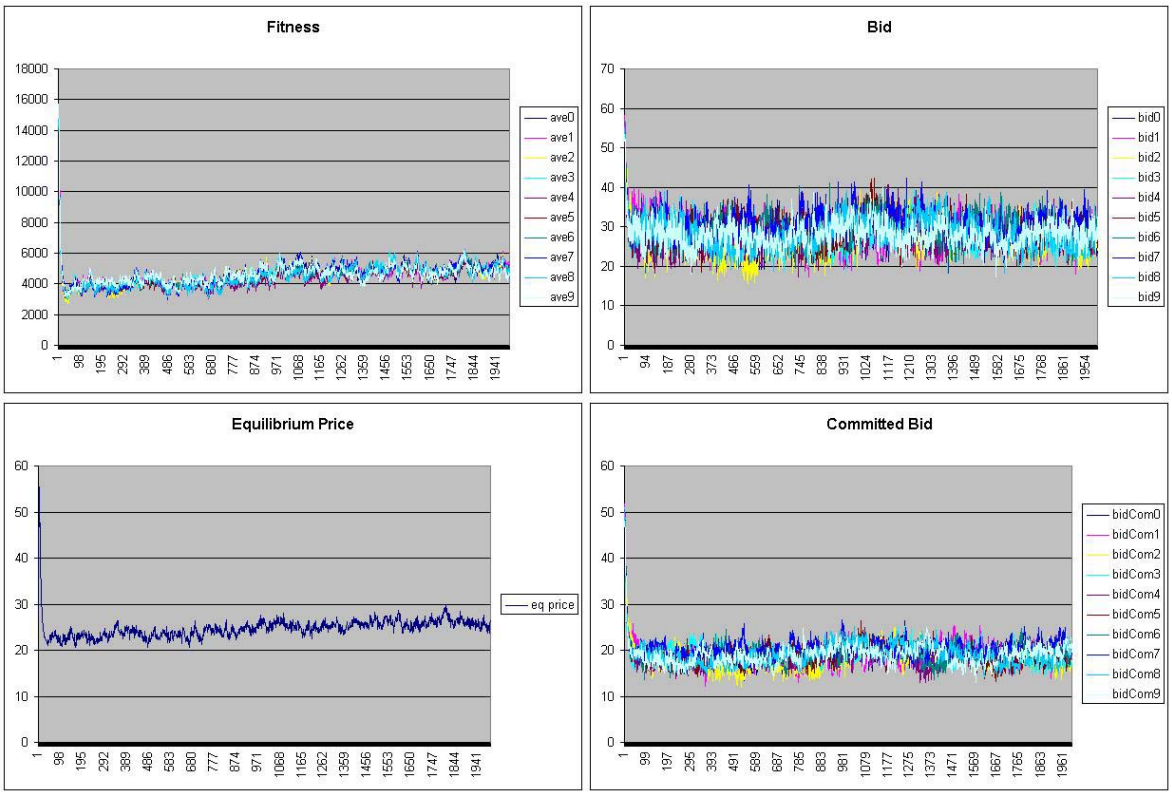

Figure 8-3 Co-Evolutionary Fitness Function, Uniform Price, GP-Automata.

Figure 8-3 shows one of the four different cost curve and capacity limit choices for all three representations using the co-evolutionary fitness function and uniform pricing, with an auction size of four bidders.

The same effects that occurred moving from ten to four bidders under the discriminatory pricing model (i.e., higher equilibrium price and bids) occur in the uniform pricing model. In some cases, these effects are more pronounced.

Curiously, comparing these results with those obtained under the same conditions under the discriminatory pricing scheme, we see that the uniform pricing scheme led to the same or higher equilibrium prices and bids than the discriminatory pricing. It seems that uniform pricing with ten bidders lowered the price, but with only four bidders, the price-raising effect of market power is much more pronounced in uniform pricing than in discriminatory pricing. The price-lowering effect of uniform pricing is overcome by this compound effect.

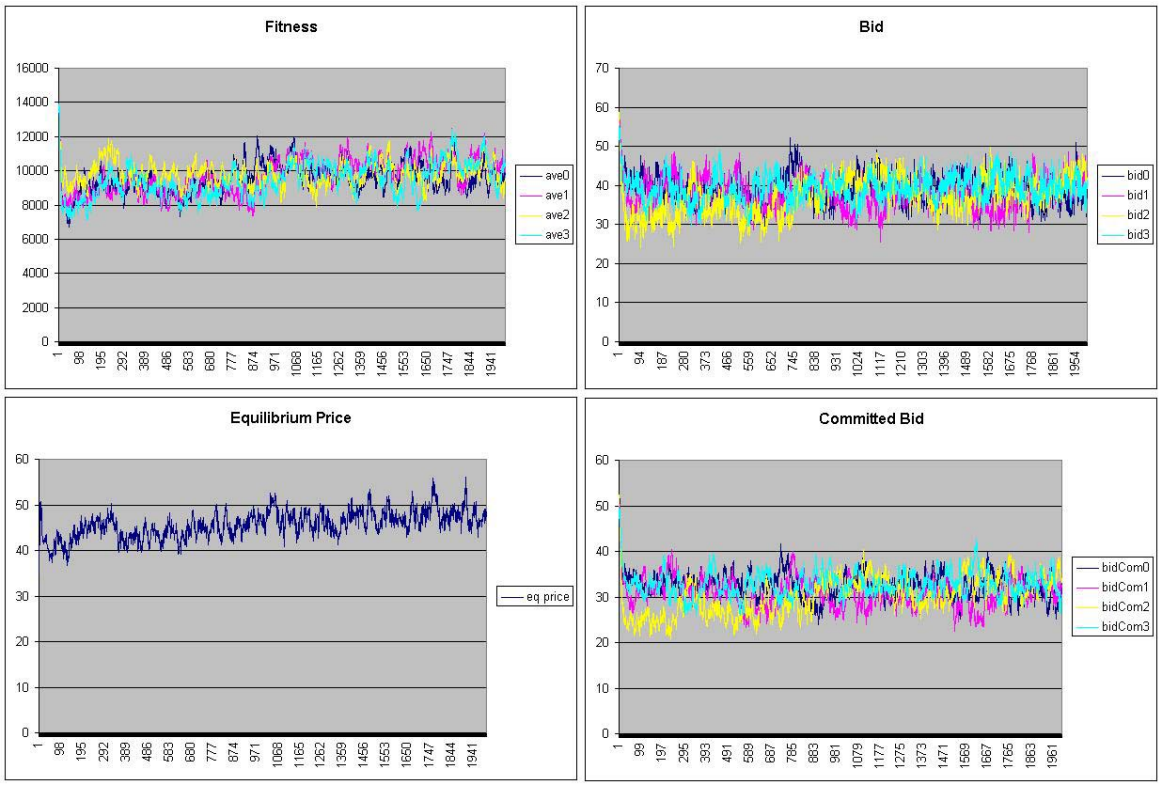

Figure 8-4 Co-Evolutionary Fitness Function, Uniform Price, GP-Automata.

8.2 Evolution by Periodic Immortalization

The experiments described in this section developed the bidders through co-evolution for 1,000 generations. At that point, the best half of the population was immortalized, and the fitness function was thereafter measured by performance against the immortals. A set of the last five immortalized populations was maintained, and the fitness was determined by playing against all of these populations. Before five populations had been immortalized, the bidders simply played all the populations that had thus far been immortalized. Each of the following experiments continued evolution for 10,000 generations, immortalizing the best half of the population every 1,000 generations.

These experiments used only the complex representations, GP-Automata and Neural-Automata. The fixed-bid representation was not evolved.

8.2.1 Experiment Set 3: Immortalized Population Fitness Function, Discriminatory Price

This experiment used discriminatory pricing. Recall that with discriminatory pricing, each bidder, if it has a bid accepted, gets paid the price it listed in the bid. The rest of the details are identical to those described in the first section. Figure 8-4 shows one of the different cost curve and capacity limit choices for the two finite-state automata representations using the immortalizing fitness function and discriminatory pricing, with an auction size of ten bidders.

The difference between these graphs, as in the co-evolutionary case, stems from the difference between the cost curves and capacity limits of the bidders. In each case, the population gradually improved in fitness (and the equilibrium price increased along with it) until about the 5,000th generation, after which it leveled off. At this point, the equilibrium price actually began decreasing slightly.

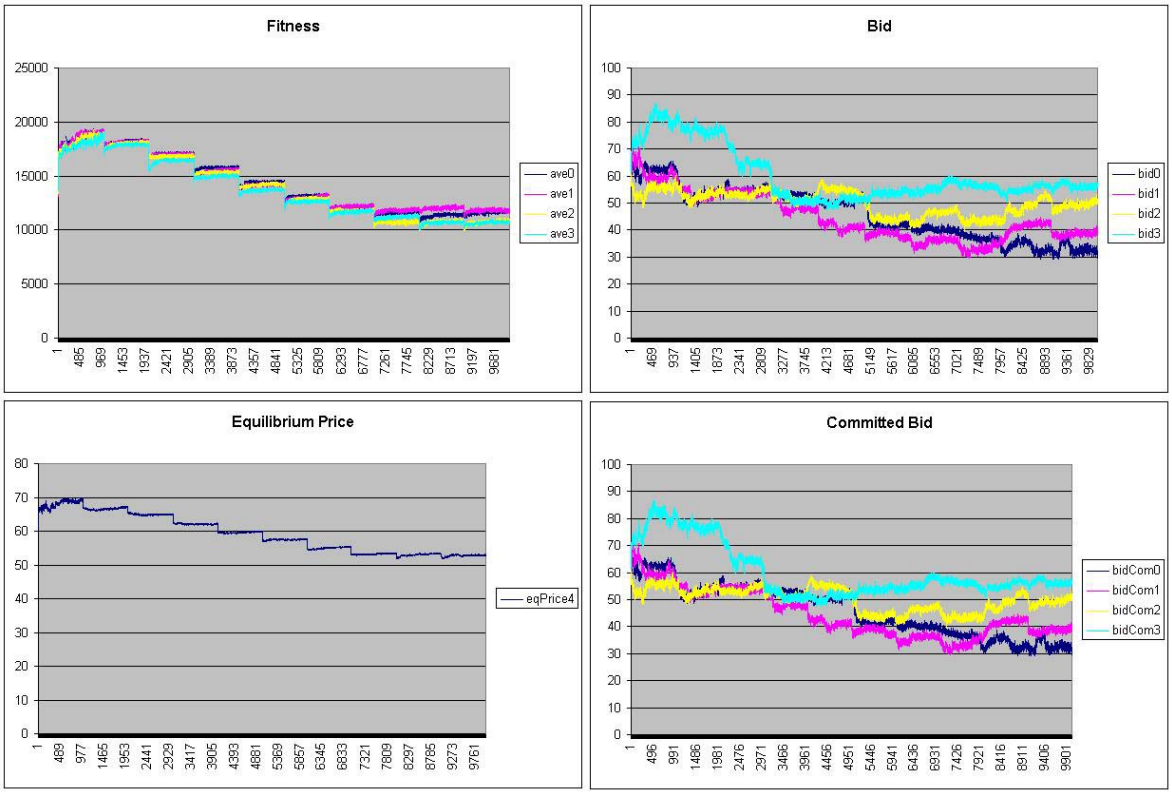

Figure 8-5 Immortalized Population Fitness Function, Discriminatory Price, GP-Automata.

Figure 8-5 shows one of three variations of two different cost curve and capacity limit choices for the two finite-state automata representations using the immortalizing fitness function and discriminatory pricing, with an auction size of four bidders.

It is interesting to note that the effect of number of bidders on average equilibrium price (i.e., fewer bidders results in higher price) is not nearly as pronounced in the immortalizing fitness function case as in the co-evolutionary case.

8.2.2 Experiment Set 4: Immortalized Population Fitness Function, Uniform Price

This experiment used uniform pricing. Recall that with uniform pricing, each bidder, if it has a bid accepted, is paid the equilibrium price, or the price of the highest accepted bid. The rest of the details are identical to those described in the first section.

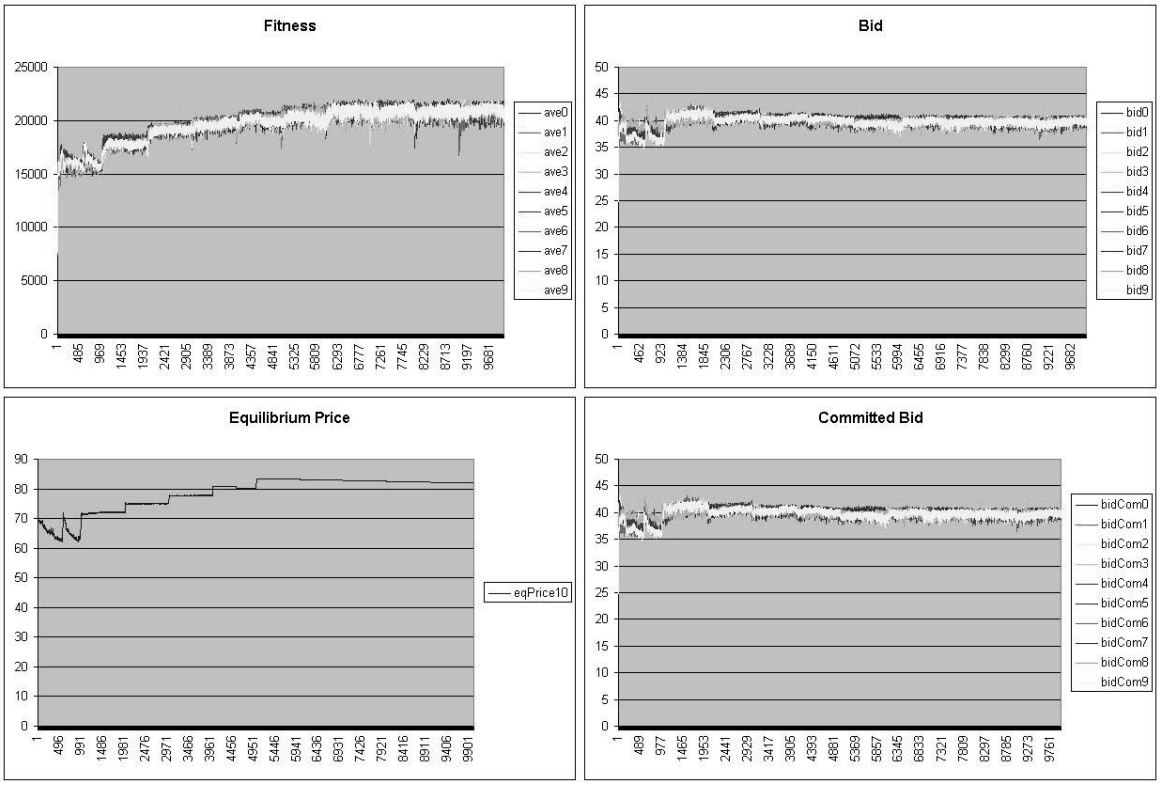

Figure 8-6 Immortalized Population Fitness Function, Discriminatory Price, GP-Automata.

Figure 8-6 shows one of four variations of two different cost curve and capacity limit choices for the two finite-state automata representations using the immortalizing fitness function and uniform pricing, with an auction size of ten bidders.

These results differ remarkably from those of the discriminatory case. Whereas the discriminatory pricing model led to gradual fitness and price increase, uniform pricing led to decrease in both fitness and price. The fitness increases for the first 1,000 generations, when the fitness function is co-evolutionary, and afterwards it decreases sharply at each replacement of the immortal bidders.

Of course, the effect of differences in cost curves and capacity limits also occurred in this experiment; those with higher costs bid higher and got lower fitness.

Figure 8-7 shows one of four variations of two different cost curve and capacity limit choices for the two finite-state automata representations using the immortalizing fitness function and uniform pricing, with an auction size of four bidders.

Comparing these results to those in the case with ten bidders, we see the usual effect that fewer bidders mean higher average equilibrium price and higher fitness.

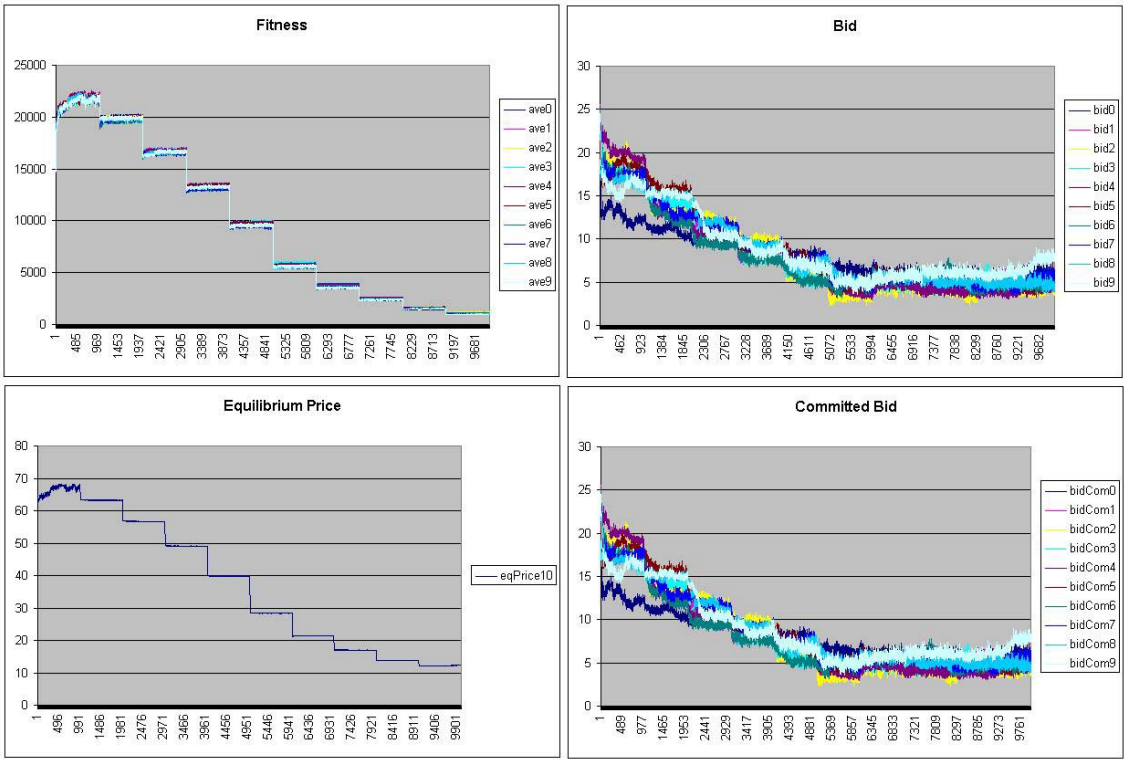

Figure 8-7 Immortalized Population Fitness Function, Discriminatory Price, GP-Automata.

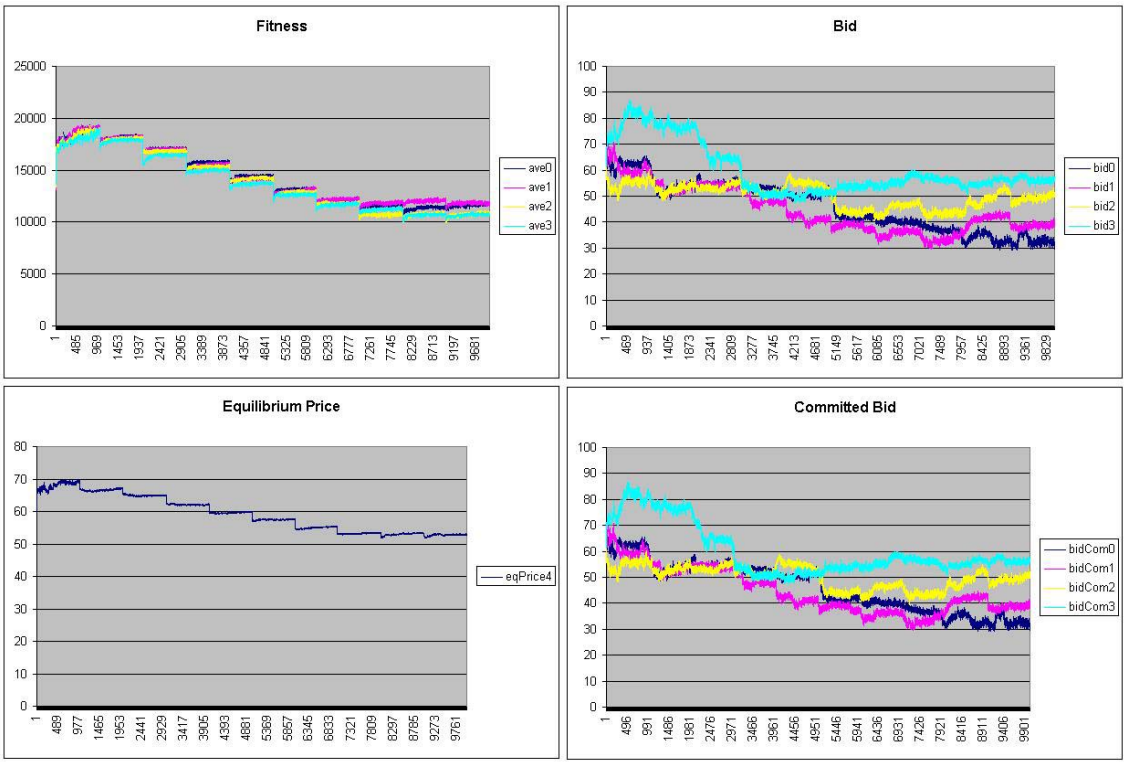

Figure 8-8 Immortalized Population Fitness Function, Discriminatory Price, GP-Automata.

9 Summary and Discussion

The first part of this research has been developed and tested, but events blocked the completion of this work. The first part was to direct the decision tree structures of the genetic algorithm to provide agents with information not previously provided. The conversion of the decision trees including uncertain dependencies is presently under development.

The research to explore the adaptability of finite state automata to a simulated electric power market was thoroughly tested. The finite state automata were developed using a genetic algorithm. Two different types of finite state automata, GP-Automata and Neural-Automata, were tried. Their performance was compared to each other and to a third simplified representation that served as a baseline of comparison. Extensive test sets evaluated the program and established some differences between market rules.

9.1 Conclusions

9.1.1 Interval Analysis

The interval analysis is proving to be a dramatic tool for estimating the spread of the distribution function. This spread is useful when the economic driving factors are of known but uncertain dependency. The ability to estimate the upper and the lower cost of production should prove most useful in a competitive environment. As the tool is extended to the application of Value at Risk or Profit at Risk, the results should be as equally dramatic.

9.1.2 Genetic Algorithms

The difference between GP-Automata and Neural-Automata lies in the data processing structure used to compress the bandwidth of the input to the finite state machine. GP-Automata used parse trees, and Neural-Automata used neural nets. The first conclusion to draw from the data in the previous chapter is that GP-Automata and Neural-Automata performed nearly the same under all conditions. From this, one can conclude that both a parse tree and a neural net are equally capable of processing the data received during rounds of bidding in an auction.

The second conclusion to draw is that the finite state automata behaved, within reasonable limits, in ways similar to the baseline representation. In those cases in which the market outcome was dissimilar, the finite state automata achieved higher fitness than the baseline. Since the average fitness was higher for all populations, each of which represented one bidder in the auction, this indicates cooperation taking place between bidders in the auctions.

Finally, though the fitness often changed during the course of evolution, this does not indicate exactly what is happening. With co-evolution especially, the average fitness going down does not necessarily mean the population is getting worse, and the average fitness going up does not necessarily mean the population is getting better. It only indicates

performance of the agents relative to each other for the co-evolutionary case and relative to the immortal agents in the immortality fitness function case.

9.2 Improvements

9.2.1 Interval Analysis Improvements

Two improvements will be evaluated in the future: obtaining narrower bounds from information about correlation and decreasing the computation time for finer discretizations. Linear programming problems have more than 4,000 variables for a 64x64 discretization, so computation time becomes significant. It may be possible to decrease computation time by using another linear programming method with improved speed or by using a parallel algorithm.

9.2.2 Genetic Algorithm Improvements

The markets rules were not chosen arbitrarily, but neither were they unique. A double-sided auction should be implemented next.

This research assumed that the cost of producing electricity by a multi-unit GENCO could be simplified as a single quadratic cost curve. This should be extended to perform a more complex economic dispatch optimization.

This research considered only the day-ahead market used to determine allocation for the next day. It ignored the existence of forward markets on larger time-scales (e.g., weekly, monthly, etc.) and the hourly spot market held the hour before delivery. Future research should explore these types of markets or develop strategies that make decisions for all of these timescales.

The effectiveness of evolving finite state automata as learning agents was measured only relative to other evolved finite state automata. Future research could compare strategies developed through other learning mechanisms to these agents. One could also evolve the finite state automata in parallel with another learning algorithm to determine which is more adaptable.

The Neural-Automata representation has not been tried for any other genetic algorithm problems. Future research into this representation could test this representation on other problems that accommodate the use of finite state machines. Obvious test problems are those that have already been tried with GP-Automata, such as simple economic games as Divide the Dollar [Ashlock, 2000] or as control structures in simulated robots [Ashlock,, 2001].

The parse trees used in the GP-Automata took real numbers as input and hence, used real numbers as intermediate values during evaluation of the tree. Since limits were not placed on the possible connections between parse tree nodes, this means that a real number could potentially be fed into a boolean input. Since booleans are defined in terms of integers by the convention (e.g., 0 = false, everything else = true), this could lead to every real being fed as a

boolean argument being evaluated as true, since the odds of getting a real exactly equal to zero are infinitesimal. However, this does not preclude other data types, such as booleans and integers, still being fed as boolean inputs, and so it will not prevent the boolean functions from operating normally. Randomly generated trees will simply have a bias toward true values being fed to boolean inputs. One potential way to correct this would be to implement a grammar to build the parse trees in which feeding real numbers as boolean inputs would be disallowed. This work assumed that evolution would simply weed out those parse trees that use reals as booleans if they cause problems.

Appendix: Review of Interval Mathematics

The interval computations field is often taken as beginning with the work of Moore [1966], although earlier relevant works exist. An interval value is describable using two real numbers, which are called the low bound and high bound. For example, given interval X= [a, b], a and b are real numbers, a is the low bound and b is the high bound. If a=b, this interval value is the real number a. Set theory can also be used to describe the interval X=[a,b]. We can define it as a set X={x: a<=x<=b}.

If we say [a,b]=[c,d], then a=c and b=d. If [a,b]<[c,d], then b<c. Other relationships may also be defined. Interval arithmetic includes addition, subtraction, multiplication and division. Let X=[a,b] and Y=[c,d] be two intervals. The following gives the definition for arithmetic operations based on the set definition for intervals.

$$X \otimes Y = \{x \otimes y : x \in X, y \in Y\}$$

where $\otimes$ is in +,-,*,/.

Therefore, X+Y = [a+c,b+d] and X-Y = [a-d,b-c]. Multiplication is a little more complex.

XY=[min(ac,ad,bc,bd), max(ac,ad,bc,bd)].

Division is a bit more complex still. Note that Y must not include zero.

$1/Y = [1/d, 1/c]$ if $0 \notin Y$

$X/Y = X(1/Y)$ if $0 \notin Y$

If Y includes zero, X/Y should be [-∞, ∞] if the interval system includes infinites as allowable endpoints.

Interval arithmetic also includes the following characteristics:

- Set Rule
  - $(V \cup W) \pm Z = (V \pm Z) \cup (W \pm Z)$
- Rule for the addition and subtraction of infinite or semi-infinite intervals
  - [a,b]+[-∞,d] = [-∞,b+d]
  - [a,b]+[c, ∞] = [a+c, ∞]
  - [a,b] ±[-∞,∞] = [-∞,∞]
  - [a,b]-[-∞,d] = [a-d,∞]
  - [a,b]-[c,∞] = [-∞,b-c]

- Associativity and Commutativity
  - $X+(Y+Z) = (X+Y)+Z$
  - $X*(Y*Z) = (X*Y)*Z$
  - $X+Y = Y+X$
  - $X*Y = Y*X$

Unlike in real arithmetic, operations are not invertible, which means there is no inverse operation for a given operation. Although for the real domain, + and – are inverse operations, in interval mathematics this is not true.

In interval analysis, interval-valued functions form a major topic. An interval function F is interval-valued and has one or more interval arguments. For a real-valued function f of real variables $x1,\ldots,xn$, if we have an interval function F of interval variables $X1,\ldots Xn$, and if $F(x1,\ldots,xn) = f(x1,\ldots,xn)$ for all $xi(i=1,\ldots,n)$ then F is an interval extension of f.

Interval functions have the following characteristics:

- Inclusion monotonicity

If $Xi \subseteq Yi$ ($i=1,\ldots,n$) then $F(X1,\ldots,Xn) \subseteq F(Y1,\ldots,Yn)$.

- Arithmetic inclusion monotonicity

If op denotes +,-,*, or /, then $Xi \subset Yi$ ($i=1,2$) implies ($X1$ op $X2$) $\subset$($Y1$ op $Y2$).

Excess width is an issue in interval mathematics. Let us use a simple example to explain this problem. For interval value $X=[a,c]$, what is the result for $X-X$? A naïve calculation gives a result that is not zero, but $[a-c,c-a]$. Zero is just one real number included in this result. Obviously, it is not the desired result, which is simply zero. For functions, an interval function extension need not be unique, but can depend on the form of the real function. For example, here are two expressions corresponding to the same real function:

$f1(x) = x*x - x + 1$ and $f2(x) = x*(x-1) + 1$.

The corresponding interval extensions are:

$F1(X) = X*X - X + 1$, $F2(X) = X*(X-1) + 1$.

These do not represent the same interval function, as:

$F1([0,2]) = [-1,5]$, and $F2([0,2]) = [-1,3]$.

The true range of $F([0,2])$ is actually [3/4,3]. It can be determined by setting the derivative of $x*x - x + 1$ to 0 to get the value of x giving the low bound and substituting that real value into $f1$ or $f2$.

This is referred to as the dependency problem or excess width. It enlarges intervals in the result collection. The reason why excess width occurs is that a variable occurs more than one time in an expression. Various methods have been developed to address this issue. Some of these methods follow.

- Various centered forms:
  - Computing the range of values [Asaithambi, Zuhe, and Moore, 1982]
  - Enclosure methods [Alefeld, 1990]
  - Artificial intelligence work [Hyvonen, 1992]

Computation time tends to be a problem with these excess width removal techniques. To apply interval analysis, the following guiding principles should be considered. [Walster, 1998]:

- "Interval algorithms should bound error."
- "Interval input/output conventions should be consistent with people's normal interpretation of numerical accuracy."
- "The application of interval algorithms should be universal."
- "Where interval algorithms currently do not exist, we should get to work developing them rather than abandoning the principle of universal applicability."

Bibliography

[1] Alefeld, G., and J. Herzberger, *Introduction to Interval Computations*, Academic Press, 1983.

[2] Alefeld, G., Enclosure Methods, in C. Ullrich, ed., *Computer Arithmetic and Self-Validating Numerical Methods*, Academic Press, 1990.

[3] Aliprantis, Charalambos D., Subir K. Chakrabarti, *Games and Decision Making*, Oxford University Press, 2000.

[4] Arons, Henk de Swaan, and Philip Waalewinjn, A Knowledge Base Representing Porter's Five Forces Model, Proceedings CIMCA'99, Vienna 1999.

[5] Asaithambi, N. S., S. Zuhe and R. E. Moore, On Computing the Range of Values, Computing, Vol.28, 1982.

[6] Bather, John, *Decision Theory, An Introduction To Dynamic Programming And Sequential Decisions*, John Wiley & Sons, West Sussex, England, 2000.

[7] Bayes, Reverend Thomas, An Essay Toward Solving a Problem in the Doctrine of Chance, Philosophical Transactions of the Royal Society, 1763.

[8] Beightler, C. S., D.T. Phillips, and D.J. Wilde, *Foundations of Optimization,* Prentice-Hall Inc., 1979.

[9] Berleant, D. and J. Zhang, Representation and Problem Solving with the Distribution Envelope Determination (DEnv) Method, *Reliability Engineering and System Safety*, in press.

[10] Berleant, D., and C. Goodman-Strauss, Bounding the results of arithmetic operations on random variables of unknown dependency using intervals, Reliable Computing 4 (2) (1998), pp. 147-165.

[11] Berleant, D., and J. Zhang, Using correlation to improve envelopes around derived distributions, Reliable Computing, in press.

[12] Berleant, D., Automatically Verified Arithmetic on Probability Distributions and Intervals, in B. Kearfott and V. Kreinovich, eds., *Applications of Interval Computations*, Kluwer Academic Publishers, 1996, pp. 227-244.

[13] Berleant, D., J. Zhang, R. Hu, and G. Sheblé, Economic Dispatch: Applying the Interval-based Distribution Envelope Algorithm to an Electric Power Problem, SIAM Workshop on Validated Computing 2002 Extended Abstracts, Toronto, 2002, pp. 32-35.

[14] Berleant, D., L. Xie, and J. Zhang, Statool: A Tool for Distribution Envelope Determination (DEnv), an interval-based algorithm for arithmetic on random variables, *Reliable Computing* 9 (2) (2003), pp. 91-108.

[15] Bernoulli, Daniel, Exposition of a New Theory of the Measurement of Risk, Econometrica (1954), pp. 23-36, Translation of a report *Specimen Theoriae Novae de Mensura Sortis*, Reports of the Imperial Academy of Sciences in Petersburg, V, 1738.

[16] Best, P., *Implementing Value at Risk*, Chichester, New York, 1998.

[17] Binger, Brian R. and Elizabeth Hoffman, *Microeconomics with Calculus*, Harper Collins Publishers, 1998.

[18] Box, M. J., D. Davies and W. H. Swann, *Non-linear Optimization Techniques*, Oliver and Boyd, 1969.

[19] Bussey, Lynn E., *The Economic Analysis of Industrial Projects*, Prentice-Hall, Englewood Cliffs, New Jersey, 1981.
[20] Byrd, R. H., and P. Lu, A Limited Memory Algorithm for Bound Constrained Optimization, Technical report NAM-08, Northwestern University.
[21] Berleant, D., Automatically Verified Reasoning with Both Intervals and Probability Density Functions, Interval Computations (1993 No. 2), pp. 48-70.
[22] Doftman, Robert, Paul A. Samuelson and Robert M. Solow, Linear Programming and Economic Analysis.
[23] Fabrycky, W. J., G. J. Thuesen, D. Verma, *Economic Decision Analysis*, Prentice Hall, Upper Saddle River, New Jersey, 1998.
[24] Fang, K.-T., & Y. Wang, *Number-theoretic Methods in Statistics*, Chapman & Hall, 1994.
[25] Ferson, S., W. T. Root, and R. Kuhn, RAMAS Risk Calc: Risk Assessment with Uncertain Numbers. Applied Biomathematics, Setauket, New York, 1998.
[26] Ferson, S., What Monte Carlo Methods Cannot Do, Human and Ecological Risk Assessment 2 (1996), pp. 990-1007.
[27] Hillier, F. S., and G.J. Lieberman. *Introduction to Operations Research*, McGraw-Hill, 2001.
[28] Hyvonen, E., Constraint Reasoning Based on Interval Arithmetic: The Tolerance Propagation Approach, Artificial Intelligence, Vol.58, 1992.
[29] Kaplan, Barish, *Economic Analysis: for Engineering and Managerial Decision Making*, McGraw-Hill, New York, 1978.
[30] Kaufmann, Arnold, *The Science of Decision-making*, McGraw-Hill Book Co., New York, New York, 1968.
[31] Kotler, Philip, *Marketing Management*, 10th Edition, Prentice Hall, 1999.
[32] Kurz, Heinz D., and Neri Salvadori, The Dynamic Leontief Model and the Theory of Endogenous Growth, Twelfth International Conference on Input-Output Techniques, New York, 18-22 May 1998.
[33] Luenberger, David G., *Investment Science*, Oxford University Press, New York, New York, 1998.
[34] Lindgren, Bernard W., *Statistical Theory*, MacMillan Publishing Co. Inc., New York, New York, 1976.
[35] MATLAB Handbook, Online help version 6.0, 1984-2000.
[36] Mckelvy, M., R. Martinsen and J. Webb, Using Visual Basic 5, Que, 1997.
[37] Microsoft Developer Network, MSDN Library Visual Studio 6.0.
[38] Moore, R. E., *Interval Analysis*, Prentice-Hall, Inc., 1966.
[39] Moore, R. E., Methods and Applications of Interval Analysis, SIAM, 1979.
[40] More, J. J., and G. Toraldo, On the Solution of Large Quadratic Programming Problems with Bound Constraints, SIAM J. Optimization, Feb. 1991.
[41] Neumaier, A., *Interval Methods for System of Equations*, Cambridge University Press, 1990.
[42] Ng, K .H., Operational Planning for Energy Service Company, Ph.D. Preliminary Report, Iowa State University, Ames, July, 1999.
[43] Pindyck, Robert S., The Dynamics of Commodity Spot and Futures Markets: A Primer, Massachusetts Institute of Technology, Cambridge, May 16, 2001.

[44] Qian, Y. Y., Operations Research, Tsinghua University Press, Beijing.
[45] Quatrani, T., *Visual Modeling with Rational Rose 2000 and UML*, Addison Wesley, 2000.
[46] Raiffa, Howard, *Decision Analysis*, Addison Wesley Publishing Co., Reading Massachusetts, 1968.
[47] Regan, H., S. Ferson and D. Berleant, Equivalence of Five Methods for Bounding Uncertainty, accepted pending revision.
[48] Rose, Rational, Rational Unified Process 5.1 Online Help, www.rational.com.
[49] Schach, S. R., *Software Engineering with Java*, McGraw-Hill, 1997.
[50] Shaalan, H. and R. Broadwater, Using Interval Mathematics in Cost-Benefit Analysis of Distribution Automation, Electric Power Systems Research Journal, Vol. 27, No. 2, pp. 145-152, 1993.
[51] Sheblé, G. B., and D. Berleant, Bounding the Composite Value at Risk for Energy Service Company Operation with DEnv, and Interval-based Algorithm, SIAM Workshop on Validated Computing 2002 Extended Abstracts, Toronto, 2002, pp. 166-171.
[52] Sheblé, G. B., Computational Auction Mechanisms for Restructured Power Industry Operation, Kluwer Academic Publishers, 1999.
[53] Sheblé, Gerald B., Decision Analysis Tools for GENCO Dispatchers, Transaction on Power Systems, Vol. 14, No 2, May 1999. pp. 745-749.
[54] Shoup, T. E., and F. Mistree, *Optimization Methods with Applications for Personal Computers*, Prentice-Hall Inc., 1987.
[55] Shrestha, G. B., Song Kai and L. Goel, Strategic Bidding for Minimum Power Output in the Competitive Power Market, Tans. On Power Systems, Vol. 16, No. 4, November 2001. pp. 813-818.
[56] Silberberg, Eugene, and Wing Suen, *The Structure of Economics: A Mathematical Analysis*, Irwin McGraw Hill.
[57] Smith, Gerald W., *Engineering Economy*, ISU Press, Ames (IA), 1987.
[58] Sposito, Vincent A., *Linear Programming with Statistical Applications*, Iowa State University Press, 1989.
[59] Springer, M. D., *The Algebra of Random Variables*, Wiley, 1979.
[60] Thompson, Gerald L., and Sten Thore, *Computational Economics: Economic Modeling with Optimization Software*, the Scientific Press Series, 1992.
[61] Vazquez, C., M. Rivier, and I.J. Perez-Arriaga, Production Cost Minimization versus Consumer Payment Minimization in Electricity Pools, IEEE Trans on Power Systems, Vol. 17, No. 1, Feb. 2002, pp. 119 –127.
[62] Walsh, G, R., *Methods of Optimization*, John Wiley & Sons, 1975.
[63] Wang, Z. and F. L. Alvarado, Interval Arithmetic in Power Flow Analysis, IEEE Trans and Power Systems, Vol. 7, No. 3, Aug. 1992, pp. 1341-1349.
[64] White, Agee, Case, *Principles of Engineering Economic Analysis*, Wiley, New York, 1989.
[65] Whittington, H. W., G. M. Bellhouse, Coal-Fired Generation in a Privatized Electricity Supply Industry, Electric Power & Energy Systems 22 (2000), pp. 205-212.
[66] Williams, Jeffrey C., and Brian D. Wright, *Storage and Commodity Markets*, Cambridge University Press, 1991.

[67] Williamson, R., and T. Downs, Probabilities Arithmetic I: Numerical Methods for Calculating Convolutions and Dependency Bounds, International Journal of Approximate Reasoning 4 (1990).
[68] Wood, A. J. and B. F. Wollenberg, Power Generation, Operation and Control, 2nd ed., Wiley, 1996.
[69] Ashlock, D, GP-Automata for Dividing the Dollar, Genetic Programming 1997: Proceedings of the Second Annual Conference, July 13-16, 1997, pg. 18-26.
[70] Ashlock, D, J Freeman, A Pure Finite State Baseline for Tartarus, Proceedings of the 2000 Congress on Evolutionary Computation, 2000.
[71] Ashlock, D, Math 378: Artificial Life, Class Notes, Department of Mathematics, Iowa State University, 2001.
[72] Axelrod, R, The Evolution of Strategies in the Iterated Prisoner's Dilemma, Genetic Algorithms and Simulated Annealing, chapter 3, pages 32 41. Morgan Kaufmann, Los Altos, Calif., 1987.
[73] Boyd, R, J Lorberbaum, No Pure Strategy is Evolutionarily Stable in the repeated Prisoner's Dilemma Game, Nature, 327 (7 May 1987): 58-59.
[74] Campbell, N A, Biology, Third Edition. pg. 1108. The Benjamin/Cummings Publishing Company, Inc., 1987.
[75] Contreras, J, O Candiles, J de la Fuente, T Gomez, Auction Design in Day-Ahead Electricity Markets, IEEE Transactions on Power Systems, Vol 16, No 1, February 2001.
[76] Copeland, T V, Antikarov, *Real Options: A Practitioner's Guide*, New York: Texere, LLC.
[77] Cybenko, G, Continuous valued neural networks with two hidden layers are sufficient (Technical Report), Department of Computer Science, Tufts University, Medford, MA.
[78] Roth, A E, I Erev, Learning in Extensive-Form Games: Experimental Data and Simple Dynamic Models in the Intermediate Term, Games and Economic Behavior, Special Issue: Nobel Symposium, vol. 8, January 1995, 164-212
[79] Koza, J, Genetic Programming: On the Programming of Computers by Means of Natural Selection. Cambridge, MA: The MIT Press.
[80] Kumar, J, G Sheblé, Auction Market Simulation for Price-Based Operation, Department of Electrical and Computer Engineering, Iowa State University, 1996.
[81] Leahy, N, D Ashlock, Representational Sensitivity in A Simple Agent-Based Computational Economics Experiment, IEEE Transactions on Evolutionary Computation.
[82] Mayfield, J, D Ashlock, Acquisition of General Adaptive Features by Evolution, Lecture Notes in Computer Science, v1447, 1998, p 75.
[83] McAfee, R, J McMillan, Auctions and bidding. Journal of Economic Literature, 25(2):699–738.
[84] Nash, J, Equilibrium points in N-Person Games, 1950, Proceedings of the National Academy of Science.
[85] Otero-Novas, I, C Meseguer, C Batlle, J Alba, A Simulation Model for a Competitive Generation Market, IEEE Transactions on Power Systems, Vol 15, No 1, February 2000.

[86] Petrov, V, G Sheblé, Building Electric Power Auctions with Improved Roth-Erev Reinforced Learning, Department of Electrical and Computer Engineering, Iowa State University, 2001.
[87] Richter, C, G Sheblé, Genetic Algorithm Evolution of Utility Bidding Strategies for the Competitive Marketplace, IEEE Transactions on Power Systems, Vol. 13, No. 1, February 1998.
[88] Richter, C, G Sheblé, Ashlock, D. (1999). Comprehensive Bidding Strategies with Genetic Programming/Finite State Automata, IEEE Transactions on Power Systems, Vol. 14, No. 4, November 1999.
[89] Erev, I and A E Roth, On the Need for Low Rationality, Cognitive Game Theory: Reinforcement Learning in Experimental Games with Unique, Mixed-Strategy Equilibria, mimeo, July 1995, University of Pittsburgh.
[90] Sheblé, G B, Price Based Operation in an Auction Market Structure, IEEE Transactions on Power Systems, Vol 11, No 4, November 1996.
[91] Song, H, C Liu, J Lawarrée, R Dahlgren, Optimal Electricity Supply Bidding by Markov Decision Process, IEEE Transactions on Power Systems, Vol 15, No 2, May 2000.
[92] Wagner, B, N Leahy, D Ashlock, A Representational Sensitivity Study of Game Theoretic Simulations, submitted to 2000 Congress on Evolutionary Computation, 2000.
[93] Weber, J D, T J Overbye, A Two-Level Optimization Problem for Analysis of Market Bidding Strategies, Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign
[94] Wilson, R, Activity Rules for the Power Exchange, Report to the California Trust for Power Industry Restructuring, March 14, 1997.
[95] Wu, Q H, J Guo, D R Turner, Z X Wu, X X Zhou, Optimal Bidding Strategies in Electricity Markets Using Reinforcement Learning, Department of Electrical Engineering and Electronics, The University of Liverpool, Liverpool, L69 3Gj, U.K. and Electric Power Research Institute Qinghe, Beijing 100085, P.R. China
[96] Doty, David, Genetic Algorithm-Based Simulation of Electric Power Markets, Master's Thesis, Iowa State University, 2002.