

Spring 5-31-2019

Estimating Auction Equilibria using Individual Evolutionary Learning

Kevin James

Chapman University, kejames@chapman.edu

Follow this and additional works at: https://digitalcommons.chapman.edu/cads_dissertations



Part of the [Behavioral Economics Commons](#)

Recommended Citation

K. James, K, "Estimating auction equilibria using individual evolutionary learning," Ph.D. dissertation, Chapman University, Orange, CA, 2019. <https://doi.org/10.36837/chapman.000053>

This Dissertation is brought to you for free and open access by the Dissertations and Theses at Chapman University Digital Commons. It has been accepted for inclusion in Computational and Data Sciences (PhD) Dissertations by an authorized administrator of Chapman University Digital Commons. For more information, please contact laughtin@chapman.edu.

Estimating Auction Equilibria using Individual Evolutionary Learning

A Dissertation by

Kevin James

Chapman University

Orange, CA

Schmid College of Science and Technology

Submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computational and Data Sciences

May 2019

Committee in charge

David Porter, Ph.D., Committee Chair

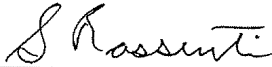
Stephen Rassenti, Ph.D.

Ryan French, Ph.D.

The Dissertation of Kevin James is approved.



David Porter, Ph.D., Committee Chair



Stephen Rassenti, Ph.D.



Ryan French, Ph.D.

May 2019

Estimating Auction Equilibria using Individual Evolutionary Learning
Copyright© 2019
by Kevin James

ABSTRACT

Estimating Auction Equilibria using Individual Evolutionary Learning

by Kevin James

I develop the Generalized Evolutionary Nash Equilibrium Estimator (GENEE) library¹. The tool is designed to provide a generic computational library for running genetic algorithms and individual evolutionary learning in economic decision-making environments. Most importantly, I have adapted the library to estimate equilibria bidding functions in auctions. I show it produces highly accurate estimates across a large class of auction environments with known solutions. I then apply GENEE to estimate the equilibria of two additional auctions with no known solutions: first-price sealed-bid common value auctions with multiple signals, and simultaneous first-price auctions with subadditive values.

¹The library is available for download at <https://github.com/kajames2/biddingga>

TABLE OF CONTENTS

1	Introduction	1
2	Individual Evolutionary Learning (IEL) Library	5
2.1	Genetic Algorithms	5
2.1.1	Representation	6
2.1.2	Parent Selection	6
2.1.3	Crossover	6
2.1.4	Mutation	9
2.1.5	Survival	9
2.1.6	Selection Operators	9
2.1.7	Survival vs. Election operator vs. Elitism Only	14
2.1.8	Choosing Selection Operators	14
2.1.9	Example Library Usage	15
2.2	Individual Evolutionary Learning	16
2.2.1	Strategy Play and Foregone Utility	17
2.2.2	Additions & Optimizations	17
3	Auction Individual Evolutionary Learning	22
3.1	Previous Work	23
3.2	Calculating Expected Profit	24
3.3	Representation	25
3.3.1	Sample Code	27
3.3.2	Accuracy Metrics	28
3.3.3	Initial Results	28
3.3.4	Independent Segment Representation	32
3.4	All Pay & Second Price Auction	36
3.5	Results	38
3.6	Conclusions	38
4	Simultaneous First Price Auctions	41
4.1	Extending GENE	42
4.1.1	Strategy Representation	42
4.1.2	Expected Profit Calculations	43

4.1.3	Accuracy Measurement	44
4.2	Results	45
4.2.1	Additive Values	45
4.2.2	Dating Game	47
4.2.3	Perfect Substitutes	49
4.2.4	Estimated Auction Attributes	49
4.3	Conclusions	51
5	Common Value Auction with Multiple Signals	52
5.1	Single-Signal Scenario	52
5.2	Multiple Signal Scenario	55
5.3	Model Results	56
5.4	Experiment	59
5.4.1	Procedure	59
5.4.2	Results	61
5.5	Conclusions	62
6	Conclusions	64
	Bibliography	66

LIST OF FIGURES

2.1	Genetic Algorithm Flow Chart	7
2.2	Sample Binary Population	8
2.3	Illustration of One-Point Crossover	8
2.4	Illustration of Bit Mutation	9
2.5	Illustration of Survival	10
2.6	Example GA Library Usage	15
2.7	Sample Evolution	16
2.8	Individual Evolutionary Learning Flow Chart	18
2.9	Individual Evolutionary Learning Hierarchy	19
3.1	Example Piecewise Representation	26
3.2	Example GENE Library Usage	27
3.3	Raw Representation 2-Player	29
3.4	Sorted Representation 2-Player	30
3.5	Asymmetric Auction Convergence	32
3.6	Example Composite Bid Function	33
3.7	Individual Evolutionary Learning Hierarchy	34

3.8	Composite GA Effects on Asymmetric Auction Convergence	35
3.9	Composite-Strategy Accuracy Matrix	35
3.10	Convergence for Common Auctions	39
4.1	Component GAs in Simultaneous Auctions	42
4.2	Relevant Areas of Bid PDF	43
4.3	Accuracy for Additive Values	45
4.4	Convergence for Additive Values	46
4.5	4-Bidder Dating Game	48
4.6	4-Bidder with Perfect Substitutes	49
5.1	Convergence for Single-Signal Scenario	54
5.2	Sample Signals	55
5.3	Precision-Midpoint Distributions	57
5.4	Model Bidding	58
5.5	Profitability of Signal Purchasing	60
5.6	Subject Bidding	61
5.7	Signal Purchasing	62

LIST OF TABLES

3.1	Accuracy by Submission Method	31
3.2	Estimated Attributes for Common Auctions	38
4.1	Estimated Auction Characteristics	50
5.1	Realized profit of Winners	61

Chapter 1

Introduction

Auctions are heavily studied, by both economists, and more recently, computer scientists. This is because they relied upon to aid the allocation of resources in the most efficient manner, whether it be a knick-knack on eBay, Ad space on Google, or electromagnetic spectrum from the FCC. However, there are still many open questions without any theoretical solutions.

Consider this simple auction: the first-price sealed-bid. In it, every bidder privately submits a bid for the item, and the one who bids the highest gets it for that price. How should someone bid in such an auction? If we assume all the bidders have the same properties (risk aversion, valuation of the item, etc), the solution is well-known [33]. However, once we allow for some eminently reasonable heterogeneity among the bidders, much of auction theory falls apart, and only special cases are known with certainty. The rest have no known solution, only having some proofs of general properties and constraints on how one should bid.

Once we expand our search to other auctions, things are even bleaker. There is very little known beyond the simplest case for a common value auction, where the item has an identical value to all bidders, but each bidder has a different estimate for that underlying value. And whenever there are multiple items, there are no exact theoretical predictions except for very special conditions. This matters, because often, poorly understood auctions are still viewed as the best mechanism for selling a good, and so we observe scenarios where auctions with significant financial stakes have had unexpected results.

New Zealand fell into precisely such a trap in 1990 [28]. The government was looking to sell mostly identical licenses for spectrum, and not knowing how much

to charge for them, decided to hold an auction. The consulting firm suggested a simultaneous second-price sealed-bid auction. They placed all the spectrum licenses up for sale simultaneously, and had everyone submit a private bid on as many licenses as they wanted. Afterwards, the highest bidder wins the license and pays the second-highest bidder’s price. While this auction type is well understood in the case of a single item, this simultaneous design is not. The government projected they would raise \$250 million in revenue, but ultimately only received \$36 million. For one item, the highest bid was \$100k, and the second highest—the price for which the highest bidder had to pay—was \$6. For another license, the highest bid \$7 million, and the second highest was just \$5000. Because the bids were posted to the public after the auction, the government was publicly humiliated for only collecting about 15% of the projected revenue. They then switched to a first-price sealed-bid auction, which worked better, but had other challenges.

So understandably, if the real-world is demanding these more complicated auctions, we would prefer to gain as much insight into how they work and how bidders will operate them prior to actually putting them into practice. Outside the limited theory available, laboratory experiments are the primary tool to analyze these complex auctions. Indeed, they have proven themselves invaluable in aiding development of the FCC’s 1994 spectrum auction [25]. However, while much cheaper than the real auctions, experiments still require significant time and resources, and given the sheer number of possible permutations of both auction designs and environmental parameters, prior information that can guide the research agenda is highly sought after. Computational modeling can provide this, but no generalized tool exists.

There are several routes that show promise towards providing insights into nash equilibrium auction bidding behavior¹, and I ultimately decided to use Individual Evolutionary Learning (IEL). IEL was first introduced by Arifovic and Ledyard in 2003 [3]. It is an multi-agent extension of the more common Genetic Algorithm (GA). Genetic Algorithms have been shown to have good convergence properties in economics, even in environments where naive best-response algorithms diverge and the equilibrium is unstable [2]. They have also been shown to be effective with finding global optima even in large decision spaces [32]. In 2017, French [16] created a proof-of-concept program that used an IEL to successfully find the equilibrium bidding strategy in a specific auction domain. Although it was limited to linear profit

¹I compare some of these techniques in more detail in chapter 3

functions and not designed around performance, it had several critical innovations that can serve as the foundation of a more generalized tool.

The goal then, is to create a computation library that applies Individual Evolutionary Learning to estimate the equilibrium bid function in a wider variety of auctions and settings that have previously been achievable. Within an auction, the library should support asymmetries between players in the form of risk aversion, subjective probabilities, value draw distributions, and budgetary constraints. For auction types, the program should match the theoretical solutions to the symmetric forms for the first-price, second-price, all-pay, and single-signal common value auctions. Furthermore, it should provide clear and consistent results for unsolved auctions, including multiple-signal common value auctions and simultaneous sealed-bid auctions. The GENEE library satisfies these requirements.

The library has multiple potential uses. The primary use—and focus of the dissertation—is to make predictions of human behavior in complex auctions. A direct byproduct of this is that the library will also yield estimates for the efficiency of an auction, the profits of each bidder, and the revenue raised for the auctioneer. This can aid auction designers in evaluating mechanisms in different environments and in the creation of experimental testbeds that stress the design.

A second use of the library is in the evaluation of human learning in laboratory experiments. Arifovic and Ledyard have successfully leveraged Individual Evolutionary Learning to estimate convergence to equilibria in complex laboratory experiments [4]. In addition, they have found that for large decision spaces with many variables, IEL outperforms other learning models [5]. There have been recent experiments that have subjects submit full bidding functions, rather than just a single bid given a randomly drawn value [11], [23]. These align with the inputs of the GENEE library, allowing it to provide both predictions and post-experiment analysis on how subjects learn relative to the IEL.

A third use of the library is to estimate econometric structural models from data. Because the model is tolerant to significant asymmetries, one can use observed bidding data in repeated auctions, such as Google AdSense auctions, and estimate underlying features of participants—like risk aversion—by determining which parameters result in equilibrium behavior nearest to the observed data.

A final usage for the GENEE library is to help bidders taking part in an auction. A bidder can encode their assumptions for competing bidders and their own preferences,

and the program will output the equilibrium behavior. Bidders and even ascribe off-equilibrium behavior to competitors and calculate their optimal bidding strategy in response to such play. This level of flexibility gives the program value that cannot be found in any other tool.

The dissertation is organized in the following manner. Chapter 2 describes the evolutionary model library. Individual Evolutionary Learning models have a niche in economics research, and no generalized library exists. While a basic version is relatively simple, there are numerous optimizations and extensions that can be added to increase performance and convergence. I compile these features into a high-performance, flexible library. This not only provides a base for auction bidding model, but also contributes to the field by lowering the programming burden of future models.

Chapter 3 details the core of the GENE auction model. It covers the numerical library that performs the necessary statistics, as well as how bidding strategies are represented and evaluated. Beginning with the representation used in French, I create a new design leveraging domain knowledge to improve convergence speed and precision. The library also takes a more general approach than prior attempts when evaluating auctions. This allows it to work on a much larger set of auctions than previously achievable. Taken together, my model increases performance—and thus practically achievable precision—by several orders of magnitude and turns a proof of concept into a practical application. After detailing the core model, I then analyze its output for auctions with known solutions.

Chapter 4 extends the library to work with 2-item auctions. It is then used to evaluate simultaneous first-price auctions with subadditive values. This environment lacks a theoretical solution, as New Zealand learned after switching to it when the simultaneous second-price auction failed.

Chapter 5 applies the model to a common value auction with signals. The common value auction is most frequently used to model the purchase of mineral rights. In it, it is assumed that all participants have the same value for an object up for auction, but each participant only has an independently drawn, noisy estimate for this shared value. While this specific model is well understood, there is no known theoretical solution for the extension where participants can purchase additional noisy signal draws for the common value. By adding some additional numerical algorithms, my model can estimate the core part of the bidding strategy in this auction. The results of the model are compared against experimentally observed data.

Chapter 2

Individual Evolutionary Learning (IEL) Library

Most learning models are relatively domain-agnostic, capable of being used across a variety of problems, albeit with varying performance characteristics. Individual Evolutionary Learning (IEL) and Genetic Algorithms (GAs)—upon which IEL is built—are no different. But, while there are commercial and open-source GA libraries, they are not very amenable to extending to IEL, and all previous implementations of IEL have been one-off algorithms, specific to the environment being modeled. Because of this, I implemented a high-performance, generalized library for a GA and the IEL extension. The purpose of this section is to detail the library as it is the core learning algorithm upon which GENEE is built.

2.1 Genetic Algorithms

GAs are a learning technique originally developed to mimic biological evolution [18]. In a GA, there is a set of strategies, called a population. Each strategy is a potential solution to the problem being solved. For example, if the problem is to find the x that maximizes some function $f(x)$, then a strategy could be a single real number, or anything that could be converted into a real number, such as a binary array. Each iteration, the strategies in the population are evaluated, modified, and culled, akin to population dynamics in nature where random mutations among members of a population are removed or promoted in future generations based upon their fitness (performance) in the environment. In a static environment, the best strategy in the

population will eventually converge to the global optimum¹ [31].

Both genetic algorithms and the related evolutionary algorithms have mostly found use as a general optimization technique. Although not considered the most efficient, they rely on very little domain knowledge and can find good results in large strategy spaces [32]. This makes them a common choice for non-linear global optimization when more specialized techniques cannot be used. Figure 2.1 shows the general flow for my implementation of the algorithm. My implementation is based on Eiben and Smith's [13]. It conforms with what is found in Arifovic [2], but with one stage being formalized and generalized. The next sections detail the implementation.

2.1.1 Representation

Strategies in genetic algorithms can take any form, but some are better suited to certain domains than others. Genetic Algorithms only require that the strategy can be mutated and evaluated. The most common representations are binary arrays, real-values, and orderings (e.g. [1,4,2,3], signifying an order of events). While my implementation has hooks for any type of representation, only the binary representation has been implemented, as that is the only one needed for this research. Figure 2.2

2.1.2 Parent Selection

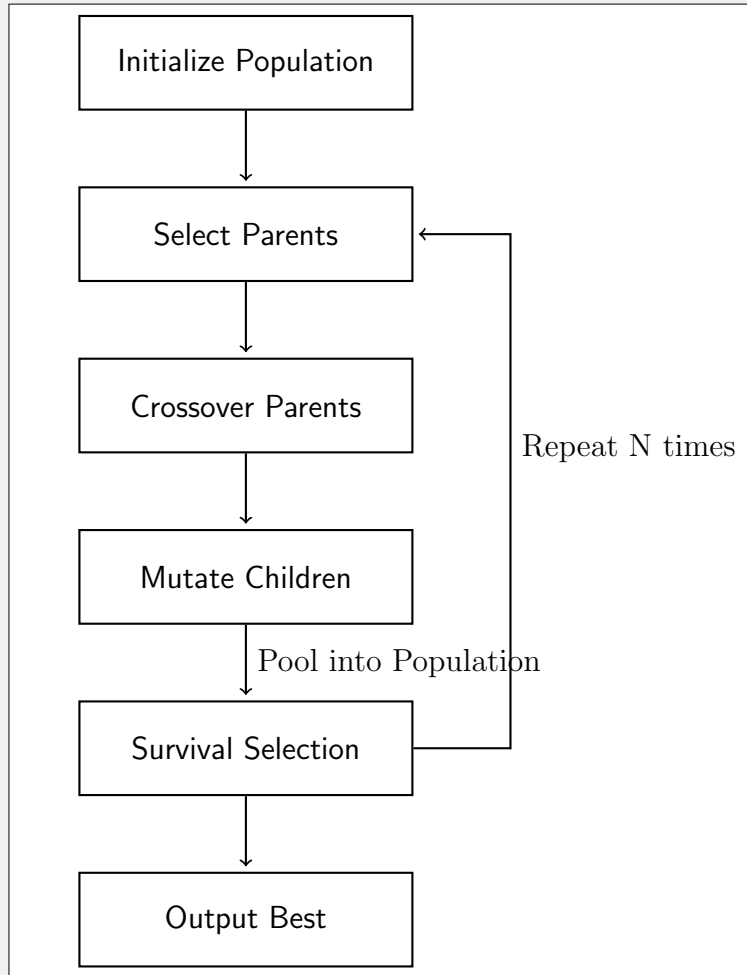
Each iteration of the algorithm requires parents to be selected. Parents are paired up, and a pair of children are created based off the parents. The number of parents selected is therefore equal to the number of children creates, which is a hyperparameter, often set to the same size as the initial population. Strategies are selected to be parents stochastically, with probability dependent on the fitness of each strategy in relation to the fitnesses the other strategies in the population. There are many popular selection techniques. They are covered in detail in a few sections.

2.1.3 Crossover

After a pair of parents have been selected, the child creation algorithm is run. The first step is to create two new strategies that contain features from each parent. For binary array strategies, I keep to the most common technique, one-point crossover.

¹In practice, reaching the optimal may require an impractical number of iterations. Like other stochastic algorithms, GAs are often ran multiple times, and the best result is used.

Figure 2.1: Genetic Algorithm Flow Chart



First, the population is initialized. Each iteration, parents are selected, paired up, and an algorithm creates child strategies with characteristics from both parents (Crossover) as well as some random component (Mutation). Then, these created children are pooled back in with the population, and the Survival stage culls the pooled population so that only there are only as many strategies as there were at the beginning of the iteration. Either one or both of the selection stages (Parent Selection and Survival Selection) will favor strategies with higher fitness in the environment. This iterative process is repeated for some number of iterations.

Figure 2.2: Sample Binary Population

	Genotypes (Encoded Strategies)		Phenotypes (Realized Strategies)
S1	1100000...01001111		[-10.48, 2.99, -7.19, -4.06]
S2	1110010...00101100		[11.30, -4.78, -7.50, -12.99]
S3	1111100...01010011	→	[-14.97, -16.78, 14.5, 18.00]
S4	1001111...11111010		[-15.39, 14.14, -16.68, -8.82]
S5	0001110...00101001		[13.30, 18.83, -19.75, -13.39]

First, the population is initialized. Each iteration, parents are selected, paired up, and an algorithm creates child strategies with characteristics from both parents as well as some random component. Then, these created children are pooled together with the population, and the Survival stage culls the pooled population so that only there are only as many strategies as there were at the beginning of the iteration. This iterative process is repeated for some number of periods.

Figure 2.3: Illustration of One-Point Crossover

110000	0...01001111	→	110000	0...00101100
111001	0...00101100	→	111001	0...01001111

First, an index is selected at random. Two children are created by taking the bits of the first (second) parent from beginning to the selected index, and then the bits of the second (first) parent for the remainder.

Let each parent strategy, p_1 and p_2 be n bits in length, and let p_i^j be the j th bit in the strategy. In one-point crossover, an index k is chosen at random between 1 and n , and the two children, c_1 and c_2 are defined by:

$$\begin{aligned}
 c_1^i &= \begin{cases} p_1^i & \text{if } i \leq k \\ p_2^i & \text{if } i > k \end{cases} \\
 c_2^i &= \begin{cases} p_2^i & \text{if } i \leq k \\ p_1^i & \text{if } i > k \end{cases}
 \end{aligned} \tag{2.1}$$

This cuts each parent into two pieces and swaps the second part, as seen in Figure 2.3.

Figure 2.4: Illustration of Bit Mutation

```
1100000...00101100 → 1100100...00101101
1110010...01001111 → 0110010...01001111
```

Indices are selected at random, each with a probability p_{mut} of being selected. Those bits are flipped to create altered children.

2.1.4 Mutation

After the children are created via crossover, they are each mutated. For binary representation, mutation occurs by iterating through each bit, and flipping the bit ($0 \rightarrow 1$, or $1 \rightarrow 0$) with probability equal to the mutation rate, p_{mut} . The mutation rate must be tuned so that small changes of only a single bit can occur, as well as larger mutations requiring multiple bit flips. This also must be changed based upon the number of bits in the array. To simplify the parameter, I implement mutation by drawing from a Poisson distribution with mean, μ_{mut} , and then flipping that many bits at random in the strategy. This is equivalent to the iterative approach for large bit lengths, but requires fewer computations and is an easier parameter to interpret and set.

2.1.5 Survival

After the children have been generated, the original population and the children are pooled together. A selection process similar to Parent Selection occurs to bring the population back down in size to that of the original population, as seen in Figure 2.5. Just like with Parent Selection, some strategies may be chosen more than once. The next section covers the different selection operators implemented during the Parent Selection and Survival stages.

2.1.6 Selection Operators

Roulette Selection

Roulette selection assigns a weight, w to each strategy s . The weight is determined by the weight function,

$$w_s = W(f_s, \mathbf{f}) \tag{2.2}$$

Figure 2.5: Illustration of Survival

	Pooled Strategies	Fitnesses
S1	[-10.48, 2.99, -7.19, -4.06]	100
S2	[11.30, -4.78, -7.50, -12.99]	57
S3	[-14.97, -16.78, 14.5, 18.00]	123
C1	[-15.39, 14.14, -16.68, -8.82]	145
C2	[13.30, 18.83, -19.75, -13.39]	63
Surviving Strategies: C1, C1, S1		

During Survival, the children are added to the rest of the population. In this case, the 2 generated children are added in with the original 3 strategies. Of these, a number equal to the original population size are chosen. While this selection typically favors higher fitness strategies, it does not just pick the highest. Also note that a strategy can be selected multiple times.

where f_s is the fitness of the strategy, and \mathbf{f} is the vector of all fitnesses in the population. Given the weights, a virtual roulette wheel is constructed with each strategy occupying an arc in proportion to its weight. The strategies are then selected by spinning the wheel N times and taking the strategies landed on. This is the same as selecting weighted random numbers with weights:

$$P(s) = \frac{w_s}{\sum_s w_s} \quad (2.3)$$

There are several commonly used weight functions. I will describe them in turn.

- Simple Roulette

Here, the weight is simply equal to the fitness of the strategy:

$$W(f_s, \vec{f}) = F_s \quad (2.4)$$

which reduces the probability of selection to:

$$P(s) = \frac{F_s}{\sum_s F_s} \quad (2.5)$$

While this is straightforward, it suffers two deficiencies. The first is that it breaks with negative fitnesses, as they would be assigned an unfeasible negative probability of being chosen. This is easily handled by shifting fitness values as described in the next method. The other problem is harder to fix. The selective pressure is weak near the optimal unless the gradient around the optimal is

large. The fitness of nearby solutions are often within a percent of the optimal, and as such are only weakly disfavored in the selection process. This limits exploitation and slows convergence to the optimal.

- Zeroed Roulette

This is a simple variant of the simple roulette method, where:

$$W(f_s, \mathbf{f}) = f_s - \min(\mathbf{f}) \tag{2.6}$$

Shifting by the minimum fitness observed immediately resolves the issue of negative fitnesses in the simple roulette scheme. Additionally, it can help increase the selective pressure by increasing the relative differences in weights (e.g. transforming fitnesses of [100,101,102] to [0,1,2]). Note, this method always eliminates the lowest performing strategy from future pools.

- Ranked Exponential Roulette

Both this and Ranked Weighted Roulette make use of ranking. Ranking entails assigning the strategy with the lowest fitness a rank of 0, 2nd lowest a rank of 1, and so forth, with the highest fitness strategy receiving a rank of $n_s - 1$. Ties are handled by assigning the average rank to those that tied. Using ranks creates separation in selection probability even when the fitness differences are small.

In exponential roulette, the weights are then assigned as follows:

$$W(f_s, \mathbf{f}) = 1 - e^{-r_s} \tag{2.7}$$

where r_s is the rank of the strategy.

- Ranked Weighted Roulette²

This has a hyper-parameter, α , that tunes how selective the mechanism is. If the number of strategies is given by n , weights are defined as:

$$W(f_s, \mathbf{f}) = (1 - \alpha)\frac{1}{n} + \alpha\frac{2r_s}{n(n - 1)}, 0 \leq \alpha \leq 1 \tag{2.8}$$

When $\alpha = 0$, the weights are equal for all strategies. Meanwhile, at $\alpha = 1$, the weights are linear with rank (normalized by the sum of all ranks). Being

²This is algebraically equivalent to Linear Ranked Selection from Baker 1985[6]

able to tune how selective the mechanism is makes this very versatile and used frequently.

Tournament Selection

Tournament selection operates by choosing a tournament size, m , and then running a tournament for each strategy selection. In each tournament, m strategies are chosen, with equal probability, and the highest fitness strategy from the the subset is selected in Thunderdome fashion.

If the tournament size is one, the tournament reduces to just random selection. Increasing the tournament size exerts stronger selective pressure, and in the limit, will only select the strongest strategy in the population.

Usually, choosing the tournament subset is done with replacement, but it is instructive to consider the case without replacement and $m = 2$. Here, the probability of a strategy being ultimately chosen is given by:

$$\begin{aligned}
 P(\text{selected}) &= P(\text{chosen})P(\text{winning}|\text{chosen}) \\
 &= \left(1 - \frac{n-1}{n} \frac{n-2}{n-1}\right) \frac{r_s}{n-1} \\
 &= \frac{2r_s}{n(n-1)}
 \end{aligned} \tag{2.9}$$

This is the same as the Ranked Weighted Roulette with $\alpha = 1$. And so we can see that Ranked Weighted Roulette can be interpreted as running a Tournament with $m = 2$ without replacement with probability α , and selecting a strategy at random (a tournament with $m = 1$) with probability $1 - \alpha$. And so, for large population sizes where the distinction between drawing with or without replacement can be safely ignored, we see that Ranked Weighted Roulette is just Tournament selection where the tournament size hyper-parameter is generalized to be a continuous variable and constrained to be no greater than two.

This is a result not found anywhere in the literature and hints at a unifying selection mechanism that allows m to be any continuous number greater than one. I do not know of a specific need for continuous values when $m > 2$, but it is a curious oversight in the literature, so I propose two possible generalizations:

- Mixing Tournament Selection

This is the straightforward way to generalize tournament sizes into continuous

space. For a given m , let m_i be the integer part of m (the floor), and m_f be the fractional component. Then, for each iteration of Tournament selection, run a tournament of size m_i+1 with probability³ m_f , and a tournament of size m_i with probability $1 - m_f$. For large population sizes, this reduces to Ranked Weighted Roulette for $0 \leq m \leq 1$, and normal tournament selection whenever $m_f = 0$.

- **Poisson Tournament Selection**

This borrows from binary mutation. Consider the simple transform $m_o = m - 1$. Now, we can consider a selection operator where a strategy is chosen at random with equal probability, and then an average of m_o other strategies are chosen to compete in the tournament. One way to do this would be to iterate through all strategies and choose them to compete with probability m_o/n , with n being the size of the population. However, it would be more efficient (and equivalent in the limit of large n) to instead draw the number of competing strategies from a Poisson distribution.

This selection operator exerts the same pressure on average as the Mixing Tournament Selection, it is not the same for any m . It will have more instances of strategies being selected off smaller tournaments and also more strategies being selected off larger tournaments. So in some sense, it may allow for a bit more exploration and exploitation than typically occurs in standard Tournament selection.

Age Selection

With Age Selection, the age of each strategy is incremented at the end of iteration of the GA. Newly created children strategies are assigned an initial age of zero. Selection is as simple as selecting the N newest strategies, regardless of fitness. I do not utilize this, but it is important to mention because many in the literature implicitly do.

Best Selection

This applies the strongest selective pressure of any method. The strategies are sorted by fitness, and the N with the highest fitness are selected. In practice, this tips the

³To reduce computation and prevent adding stochasticity on top of an already stochastic process, it seems reasonable to use a deterministic process for the number of draws in each tournament, by having $\lfloor n * m_f \rfloor$ number of tournaments of size $m + 1$ and $n - \lfloor n * m_f \rfloor$ of size m .

scale too far towards exploitation. It will find an optimum in few iterations, but it is more likely to be a local optimum.

Elitism

Elitism can be used in conjunction with another selection method. When N strategies are to be selected, elitism says to pick some number $N_e < N$ of them using Best Selection, and then pick the remaining $N - N_e$ using another selection scheme. This guarantees that the strategies with the highest fitness make it into the next population. It turns out that this feature is required to prove a genetic algorithm will converge to the optimum [31].

2.1.7 Survival vs. Election operator vs. Elitism Only

The survival stage is a relatively new stage in GAs and not universally used. Originally, the entire population would be replaced by the new children strategies. Then, Elitism was shown to be necessary for convergence, and so the children replaced all but the select elites from the original population.

Arifovic introduced the concept of an Election operator. In it, after two children were generated from the two parents, a Best Selection operator was used to pick the best two of the four (two children + two parents), thereby only keeping children that were superior to their parents. This is a fairly strong selection operator, whose strength depends on the parent selection operator chosen.

Survival encompasses these previous extensions and formalizes it into a new stage. By just using Age Selection in survival and setting the number of children equal to the population size, the stage has no effect and the algorithm reduces to the canonical form. Adding Elitism to the Age Selection gives the most common algorithm, and using some other selection criteria like Tournament or Roulette will give something capturing the essence of the Election operator by not keeping a child strategy unless it is shown to be better.

2.1.8 Choosing Selection Operators

There is no clear consensus on which selection operators to use when. There is always a trade-off between convergence speed and not getting stuck in a local optimum. In practice, Weighted Ranked Roulette and Tournament selection have tended to be

Figure 2.6: Example GA Library Usage

```
// Calculate the squared error from sin(x).
// Returns negative of error since the GA maximizes the fitness function.
float SineErrorFitness(vector<float> coeffs) {
    float sqr_err = 0;
    for (float x = -3; x < 3; x += 0.01) {
        float est = 0;
        for (int i = 0; i < coeffs.size(); ++i) {
            est += coeffs[i] * pow(x, i);
        }
        sqr_err += pow(est - sin(x), 2);
    }
    return -sqr_err;
}

int main() {
    // Create a binary GA whose phenotype is 4 floats between -20 and 20.
    auto ga = BinaryGA({4, FloatEncoding{-20, 20}}, SineErrorFitness);
    ga.RunRound(100);
    // Extract the phenotype of the best strategy and print out the values.
    auto coeffs = ga.GetBestStrategy().phenotype;
    for (float c : coeffs) {
        cout << c << ", ";
    }
    cout << endl;
    return 0;
}
```

Sample C++ code implementing a GA for finding a cubic fit of the sine function between $x=-3$ and 3 .

the most popular methods, and elitism is almost always employed during survival. Based on guidelines from the literature, I set the default Parent selector to be a Weighted Ranked Roulette with $\alpha = 0.4$, and the Survival selector to be the same with 5-strategy elitism. For binary strategies, the default mutation rate, μ_{mut} , is set to 2.

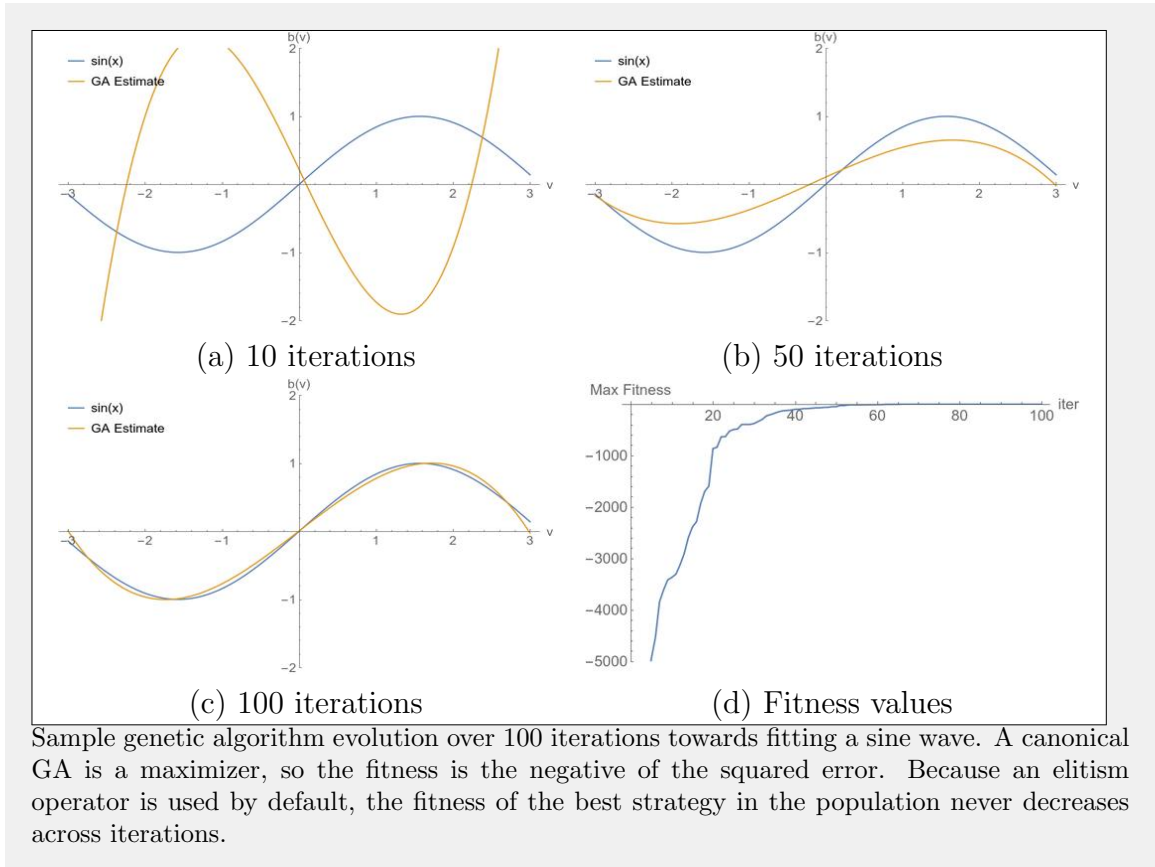
2.1.9 Example Library Usage

Figure 2.6 provides sample code for using the genetic algorithm library on the contrived example of finding the cubic polynomial that best fits a sine wave between $x=-3$ and $x=3$. The most important line of code is the creation of the GA:

```
// Create a binary GA whose phenotype is 4 floats between -20 and 20.
auto ga = BinaryGA({4, FloatEncoding{-20, 20}}, SineErrorFitness);
```

Here the GA is created with a binary genotype representation, and a phenotype representation of a vector of 4 floats, each in the set $[-20, 20]$. The population size, bit precision, genotype-to-phenotype conversion method, and all selection, mutation,

Figure 2.7: Sample Evolution



crossover methods are provided default values. The fitness function is set via an input parameter, and must be able to accept a vector of four floats and output a fitness value. Figure 2.7 shows a sample set of results for the problem and how it successfully finds a good fit by the end of the horizon.

2.2 Individual Evolutionary Learning

Individual Evolutionary Learning is an extension to genetic algorithms to enable multi-agent competition. It has been argued that GAs were not originally designed for function optimization [20], and are in many ways weak at doing so. Instead, they are better-suited to adapting to a shifting environment or fitness function. IEL leverages this fact, by having several GAs compete with one another and react to the actions of their competitors. After introducing IELs in 2003, Arifovic and Ledyard showed the algorithm to be more accurate and likely to converge [5] in large state spaces than other learning algorithms like Reinforcement Learning [14] or Experience

Weighted average [8].

2.2.1 Strategy Play and Foregone Utility

Figure 2.8 shows the process of IEL compared to a standard GA. Upon initialization, each player (population) selects a strategy at random to play for the first round (iteration). Once everyone has submitted a strategy, they can each evolve as normal, by using a their foregone utility as a fitness function. The foregone utility is calculated by holding all other players' chosen strategies constant, and replacing their own played strategy with the strategy in the population being evaluated. One way to think of the algorithm is that each GA represents an actual player in a game. Each iteration, they each must play a strategy. The population within each GA represents other possible strategies in the player's mind. Each iteration, the players update their beliefs based on how well the strategies in their mind would have performed had they chosen to play any one of them instead.

Figure 2.9 shows the structure of the algorithm, with several GAs that are coupled by the IEL environment. Computationally, standard GAs are created, and then wrapped in a data structure that allows them to submit strategies to the IEL environment and have a unique player ID. An IEL object runs the algorithm. Figure 3.2 in chapter 3 provides example code on how to set up these relationships.

2.2.2 Additions & Optimizations

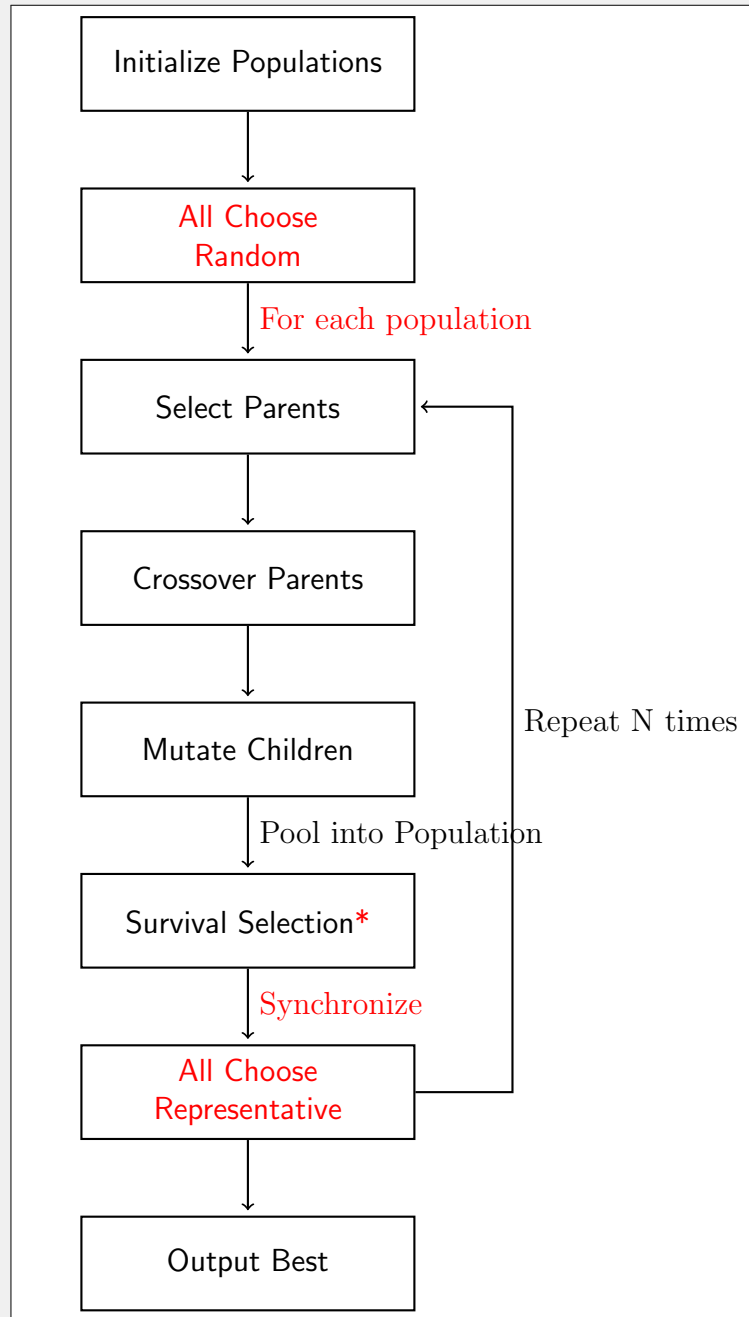
The library has some additions not found in other IEL models. Some are speed optimizations, and some give it extra flexibility for certain problem domains.

Fitness caching

In both GAs and IEL, the fitness function is traditionally the most computationally expensive step—often by multiple orders of magnitude. In a single GA, the fitness function is often constant. When it is, the GA automatically saves the fitness with each strategy, and reuses the value in subsequent periods.

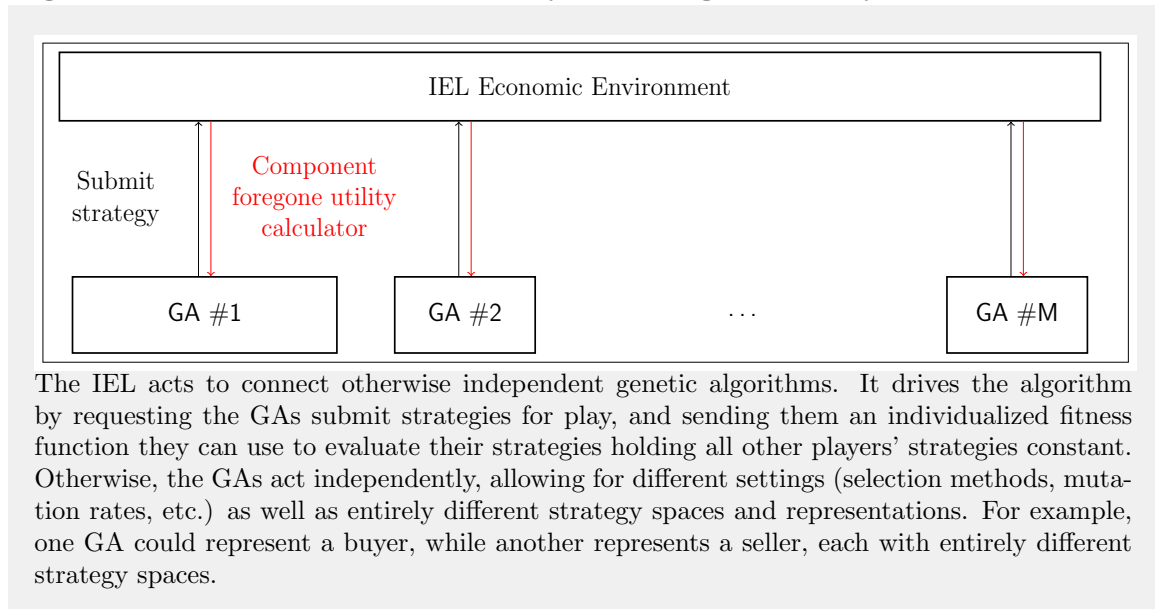
Each iteration of the GA has the original population which survived the last iteration, and newly created children. With this fitness caching, fitness only has to be calculated for the children. The most common case is for the number of created

Figure 2.8: Individual Evolutionary Learning Flow Chart



The process is largely the same as for the single GA in Figure 2.1. The two main differences are that there is a stage for each population to choose a strategy for play each iteration, and that now multiple populations are updating rather than only one. Note that in the base version, all populations update, then all populations choose a new strategy for play. The selection method for choosing a strategy for play is typically very strong, with the default set to a size-4 tournament.

Figure 2.9: Individual Evolutionary Learning Hierarchy



children to equal the size of the population, in which this case fitness caching reduces the number of fitness calculations in half.

Phenotype caching

Converting from a binary genotype to the representation can also require some computational time and can be cached with the strategy once it has been calculated. The benefit to this is that it can be used in both static fitness environments and ones where the fitness function changes each iteration. This means in IELs, the phenotypes of the previous population can be reused.

Because of phenotype caching, it is important that in an IEL, the phenotype contains all calculations that can be performed independently of what other GAs have chosen as strategies. In some cases with only light coupling between GAs, the vast majority of the calculations can be pulled out of the fitness calculation and into the genotype-to-phenotype conversion, yielding similar savings to fitness caching.

Count storage

Another important optimization is to realize that many strategies end up being copies of one another. By the end of a GA or IEL, on the order of half of the strategies in a population are not unique. To exploit this duplication, rather than storing an

array of the strategies, the unique strategies are paired with the frequency in the population. Selection strategies are tasked with returning a new frequency list. This allows for phenotype calculations and fitness calculations to be only performed once per unique strategy.

While it is possible to do an additional optimization where once children are created, they are each compared to the existing population to see if there already exists an identical strategy, in practice this can be computationally expensive, and is frequently not worth it. Because of this, there is some inefficiency where there are multiple instances of the same strategy in the population due to these serendipitous copies.

Batching

The last two features are additional features, rather than optimizations. One potential problem in an IEL is that the updating is reliant on the strategies chosen by each sub-GA. If a GA submits a poor strategy, the other GAs will adapt accordingly, leading to setbacks in the convergence. In practice, having GAs only submit their best strategies largely eliminates this. However, as a precaution, the algorithm allows for batching. With batching, each GA submits N strategies. These are paired up into N groups of strategy pairings, and the fitness of a strategy becomes the mean across the N . This sampling allows strategies to be evaluated on how well it performs against the pool of strategies the other GAs have, rather than just their best performing.

This process of batching is completely equivalent to only selecting one strategy per iteration, keeping a running total of the fitness, and only updating each population once every N iterations.

Sequential Updating

In some environments players take turns, rather than moving simultaneously. The IEL library allows for each GA to be assigned a priority. All GAs with the same priority update, and then submit new strategies.

Consider a scenario where there are buyers and sellers. Sellers each set a price, and buyers then choose how much purchase. In this, all buyers can be assigned an identical higher priority, and buyers assigned identical lower priority. Each IEL iteration, the sellers will update based on what the sellers and buyers chose the previous iteration, and submit new prices. The buyer then update based on these newly submitted seller

strategies and the submitted buyer strategies the previous iteration.

Taken as a whole, the GA and IEL base of the library provides a relatively simple coding interface, high performance, and a wide array of features, many not found in any other implementation. This lets it serve as a strong core to the rest of the GENE library, and also allows others to use it when implementing their own GAs and IELs.

Chapter 3

Auction Individual Evolutionary Learning

The rest of the Generalized Evolutionary Nash Equilibrium Estimator library can now be developed using the Individual Evolutionary Learning component of the library as a base. There are two main ways of modeling auctions: Bayesian Incomplete Information, and Complete Information. While GENE can be shown to work in the Complete Information environments, the focus will be on Bayesian Incomplete Information environments.

Suppose a bidder walks into the room where an auction for an item is being held and sees all the competing bidders. In a Bayesian Incomplete Information model, he knows how much he values the item at, and estimates a distribution of the value for each competing bidder. Furthermore, even though he knows his true value for the item being auctioned, he also estimates a distribution for his own value—this is the distribution the other bidders are assumed to ascribe to him. Given this, and assuming all bidders hold the same assumed distributions, an equilibrium bid function can be found for each bidder. The bid function, $b_i(v)$ for player i prescribes how much should be bid for each possible value draw, v .

To find the optimal bid function, consider a first-price sealed-bid auction. Let X_i be the distribution of bidder i 's value. Given the bid function $b_i(v)$, let $B_i \sim b_i(X_i)$. Given this and our previous assumptions, we can write the expected profit of player

i as:

$$\begin{aligned}
E(\pi(b_i(v))) &= \int_{\underline{v}}^{\bar{v}} (v - b_i(v)) \text{Prob}(\text{win}|b(v)) f_{X_i}(v) dv \\
&= \int_{\underline{v}}^{\bar{v}} (v - b_i(v)) \prod_{j \neq i} [\text{Prob}(b_j < b_i(v))] f_{X_i}(v) dv \\
&= \int_{\underline{v}}^{\bar{v}} (v - b_i(v)) \prod_{j \neq i} [F_{B_j}(b_i(v))] f_{X_i}(v) dv
\end{aligned} \tag{3.1}$$

where it is assumed $v \in [\underline{v}, \bar{v}]$, $f_{X_i}(v)$ is the probability density function X_i , and F_{B_j} , is the cumulative density function of B_j .

Each bidder then has an objective to maximize this expected profit:

$$\max_{b_i(v)} \int_{\underline{v}}^{\bar{v}} (v - b_i(v)) \prod_{j \neq i} [F_{B_j}(b_i(v))] f(v) dv \tag{3.2}$$

Doing a simultaneous maximization leads to the equilibrium bid functions. Crucially, this is often not analytically tractable. It has been solved when value draws are symmetric between bidders [33], [24] for both risk neutral bidders as well as some risk averse bidders with some assumption. For asymmetric bidder, it has been solved when all values are drawn uniformly with the same lower support, or when the values are drawn uniformly with asymmetric value shifts [22]. Similar limitations exist in other auction types.

3.1 Previous Work

There has been some research into using computers to solve for the equilibrium bid functions. The greatest success has been by reducing the problem to a system of ODEs, and then employing various methods to solve them. Li and Riley [27] created a program BidComp to do this. And while it has high precision, it also carries significant assumptions, including: the auction must be first-price sealed-bid, value distributions must have the same support, and the value distributions must be orderable via stochastic dominance. It also has some inaccuracies at the endpoints of the function. Hubbard [19] surveys the literature of these systems of ODE solutions and describes the most modern techniques for solving them, but they all face similar constraints and assumptions that make it difficult to generalize to new auction types and more generalized utility functions and subjective probability functions.

There have also been a few attempts to solve for the bid functions using GAs and IEL. Andreoni and Miller [1], used a GA to solve symmetric first price auctions. They assumed the bid function to be linear (which is the case), and task the GA to find the slope. To evaluate the fitness, they used Monte Carlo simulations. While it was reasonably successful, it is limited to a very narrow set of problems.

Chernomaz generalized Andreoni and Miller by using an IEL, which allowed for asymmetric bidders [9]. He also generalized to allow quadratic functions. However, this is still a very strong constraint on the functional form, and he found that even allowing for quadratic functional forms led to inaccuracies when the underlying solution was actually linear.

French [16] generalized further by allowing for piecewise linear bid functions. With enough segments, the program was able to capably estimate non-linear bid functions. Additionally, French evaluated the expected profit equation analytically, rather than using Monte Carlo simulations. This was done by leveraging the linearity of the bid functions, the uniform distributions of values, and risk neutrality. While a piecewise linear function can be used to describe any continuous function form, the analytical evaluation of the expected profit equation only works in the more simple cases.

3.2 Calculating Expected Profit

French's method of directly calculating the expected profit integral can be generalized to non-linear values and utility functions, as well as to other auction types, at the cost of requiring numerical integration rather than analytically. Additionally, an array of distribution and order statistic calculations are required. This can be quite computationally intense for some auctions, although many of the calculations must only be performed once per IEL iteration since the submitted strategies are held constant. The benefit is that it is a deterministic result and has high precision. The downside is the additional programming required to implement.

Meanwhile, Chernomaz and Andreoni used Monte Carlo simulations to estimate the expected profit. The benefit to this method is that this is computationally straightforward. For example, a second price auction can be evaluated by simulation. To do so, draw a value for each bidder from their respective value distributions, lookup their corresponding bids according to their submitted bid functions. The bids are sorted and the highest bidder receives the item and pays the second highest bid.

This is much simpler than calculating the expected profit integral directly, where each bidder must calculate the distribution of the highest order statistic of the $n - 1$ other bidders based on the bid functions submitted this iteration (to determine the probability of each bid winning), as well as the conditional expected highest statistic (to determine the expected price of the item upon winning).

The downside to Monte Carlo simulations is getting the necessary precision. First, consider a first-price auction with 10 identical bidders drawing values from the standard uniform. Given that a bidder’s value is drawn to be 0.1, the odds that all other bidders have a lower value—and thus the bidder is expected to win—is 10^{-9} . And so, to get an accurate estimate of whether $b(0.1)=0.09$ or $b(0.1)=0.08$ is superior would require millions of draws.

Similarly, suppose all bidders are bidding at the equilibrium. One bidder starts to shade by ϵ over a small value range Δv . Now, every time they win within Δv , the bidder makes an extra ϵ in profit upon winning, but has some small additional probability of losing, which ultimately outweighs the added profit the rest of the time. To accurately measure this with Monte Carlo simulations also requires on average $\epsilon^{-1}v^{-(n-1)}$ samples for a set of draws where the bidder loses due to underbidding.

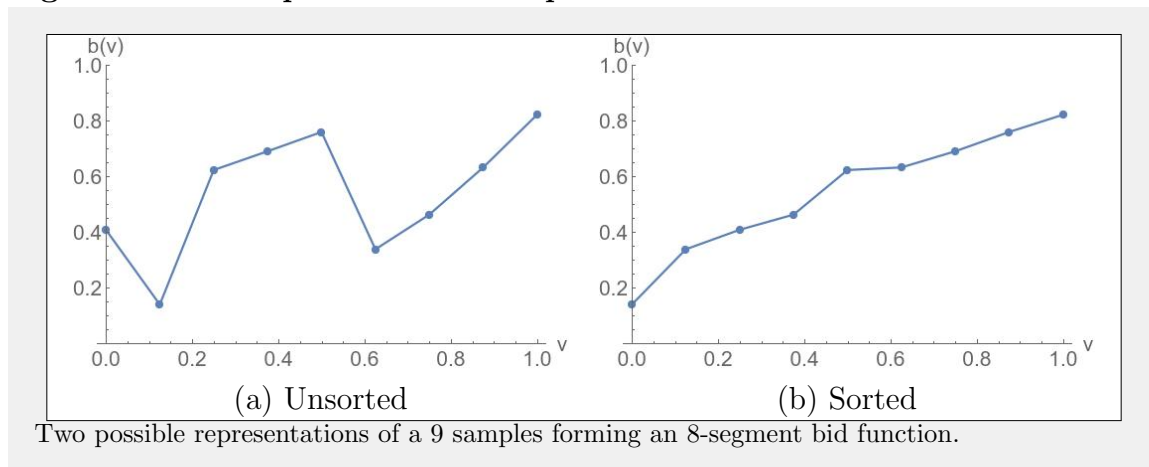
These problems with Monte Carlo simulations do not have large impacts in either Chernomaz or Andreoni and Miller in part because they both constrain the bidding functions to simple forms like linear, so a mistake in one area of the bidding function also applies to all other areas of the bidding function. A below-optimal slope means the bid function too low at all values, not just on some small subset of values. But such a luxury does not exist with more flexible representations like a piecewise linear bid function.

While it is perhaps possible to ameliorate some of this imprecision with Monte Carlo simulations by utilizing more advanced techniques such as importance sampling, I decided to just use the direct evaluation approach. However, as is detailed in section 3.3.3, the regions with low win probability are still difficult to solve for accurately without additional techniques.

3.3 Representation

A single strategy in the IEL represents a candidate bid function. Suppose we represent such a function by n floats, each being a sample from the function equally spaced

Figure 3.1: Example Piecewise Representation



across the value range. If we do a first-order interpolation, this yields a linear piecewise function with $n - 1$ segments. This can be seen in Figure 3.1. This is representation used by French, and in GENE¹.

All bid functions begin as a binary genotype, are converted from an implicitly assumed gray code, and then turned into an array representing the y-samples of the bid function.² Once converted into y-samples, there is an optional final step. Because bid functions should typically monotonically non-decreasing, the y-samples can be sorted. Figure 3.1 shows both the unsorted and sorted representation with the same underlying y-samples.

Sorting eliminates all non-monotonic functions from the search space. While it is apparent that the equilibrium strategy is a member of the reduced search space, it is not apparent a priori that limiting search to this space is beneficial. For example, without sorting, the effects of mutation are localized around the sample mutated. But if the samples are sorted, such a mutation can result in a shift of the entire of the bid function. Such effects may be deleterious to the search process.³

To test whether sorting leads to better results, as well as configure other param-

¹A simple cubic spline was investigated, but ultimately rejected because it does not enforce a monotonicity constraint even when the samples are monotonically non-decreasing, a common feature of bid functions. There exists other cubic interpolations that do enforce monotonicity, and they remain a worthwhile branch for future research.

²The y-samples as well as all the statistical calculations are stored as Eigen library [17] Arrays to simplify the syntax.

³There are other possible representations, such as a truncation representation, where any time the y-sample decreases, it is truncated to be equal to the previous value instead. In practice, this representation shows very low performance.

Figure 3.2: Example GENE Library Usage

```
int n_bidders = 2;
// Have each bidder draw values from a standard uniform.
vector<Distribution> value_dists(uniform_distribution<>(0, 1),
                               uniform_distribution<>(0, 1));

// Must configure the individual GAs x-y ranges
// Can also set custom values for parameters like population size.
vector<BidFunctionGAConfiguration> configs(n_bidders);
for (int i = 0; i < n_bidders; ++i) {
    configs[i].value_range = {lower(value_dists[i]), upper(value_dists[i])};
    configs[i].bid_range = {0, upper(value_dists[i])};
}

// Create the auction, individual GAs, and the IEL driver.
FirstPrice auction(value_dists);
auto gas = MakeGAs<FirstPrice, Scatter>(configs);
auto driver = MakeIELDriver<FirstPrice, Scatter>(gas, auction);

driver.RunRound(1000);
// Output the most common strategies and fitnesses.
for (const auto& ga : gas) {
    cout << ga.GetBestStrategy().phenotype << endl;
    cout << ga.GetBestStrategy().fitness << endl;
}
```

Sample C++ code running a first-price auction with 2 bidders drawing values from the standard uniform. There is much more boilerplate code than in the single GA, but, modifying the boilerplate for other auctions and player asymmetries is comparatively simple.

eters, the first-price sealed-bid auction is used as a baseline. The equilibrium bid functions are well known, and they are well understood to not be a strictly dominant strategy. This will lead to fluctuations from the equilibrium, allowing for a fair comparison on which model parameterizations lead to the great accuracy and precision.

3.3.1 Sample Code

Figure 3.2 shows an implementation of a First-Price auction with 2 bidders. Each bidder can be assigned a different value distribution (in the example there are each just given a standard uniform). Afterwards, the GA configurations are set for each bidder. Here, settings such as population size, survival selection method, mutation rate, etc. can be set on a per GA basis.

Next, the auction is created by passing in the value distributions. Additional optional arguments include utility functions and subjective probability functions. Finally, the individual GAs and the IEL driver are created. Once all that boilerplate

is complete, the driver can run for a set number of rounds and the each GA can output their bid functions and estimated expected profit. Additionally, the auction can be queried for common auction metrics such as the expected value realized by each bidder and the auctioneer’s expected revenue from each bidder.

3.3.2 Accuracy Metrics

French and Chernomaz both measured the accuracy of the estimated bid functions by using a continuous version of Root Mean Squared Error, D_1 , and Mean Signed Error, D_2 . They are given by the following equations:

$$\begin{aligned}
 D_1 &= \frac{1}{(\bar{v} - \underline{v})} \sqrt{\int_{\underline{v}}^{\bar{v}} (b(v) - \hat{b}(v))^2 dv} \\
 D_2 &= \frac{1}{(\bar{v} - \underline{v})} \int_{\underline{v}}^{\bar{v}} (b(v) - \hat{b}(v)) dv
 \end{aligned}
 \tag{3.3}$$

Together, D_1 will inform how close GENE’s result is to the theoretical equilibrium, and D_2 indicates whether there is any directional bias.

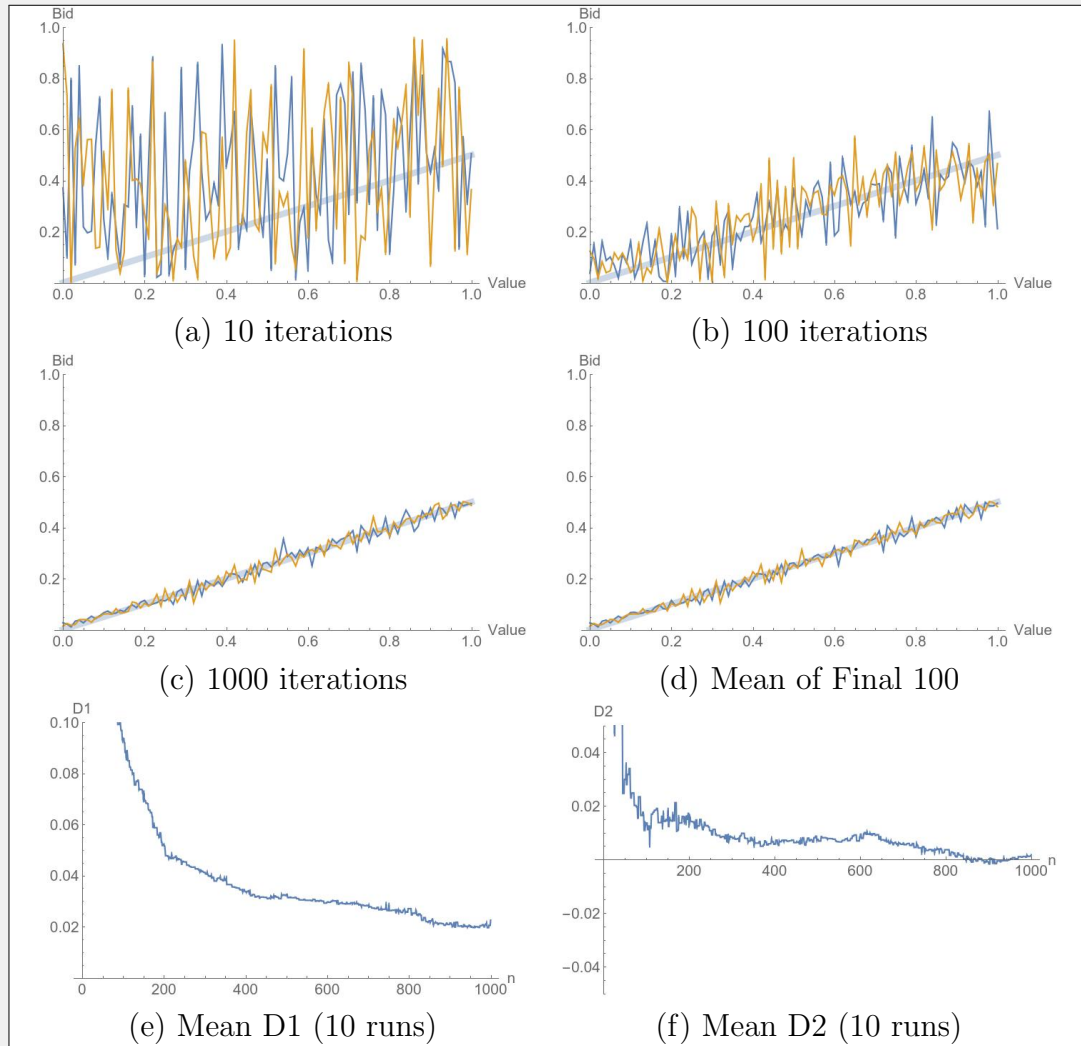
3.3.3 Initial Results

For initial testing, I ran a symmetric two-player first-price sealed-bid auction with draws from the standard uniform. For this, the equilibrium bid function is just $b(v) = 0.5v$. Each GA was configured to have 1000 strategies, and the bid functions is composed of 100 segments. Figure 3.3 shows the results for when the y-samples are not sorted during phenotype conversion, and Figure 3.4 shows the results when they are sorted.

Based on the graphs of D_1 and D_2 , the sorted representation converges within about 200 iterations, while the unsorted appears to still not be done even after the full 1000 iterations. Because the equilibrium for the first-price auction is not a dominant strategy, the bidding functions fluctuate around the theoretical equilibrium. This makes it valuable to average bidding functions submitted by each player over the last 100 iterations. When performed, D_1 drops further in the sorted representation (from about 0.07 to about 0.02). The increased accuracy can also be seen visually when comparing parts (c) and (d) in Figure 3.4.

Another open question is which selection mechanism is appropriate when sub-

Figure 3.3: Raw Representation 2-Player



Parts (a)-(c) highlight the evolution of the 2 bidders' submitted bid functions across one run of 1000 iterations using unsorted representation. The theoretical equilibrium is displayed as the thick blue line. Part (d) takes the bid functions each bidder submitted over the last 100 iterations and averages them. Because there are often fluctuations about the equilibrium, taking the mean of the submitted bid functions over the last 100 iterations provides the best fit. Parts (e) and (f) show the progression of D1 and D2 averaged over 10 runs. D1 does not appear to have fully converged by the end of the 1000 iterations.

Figure 3.4: Sorted Representation 2-Player

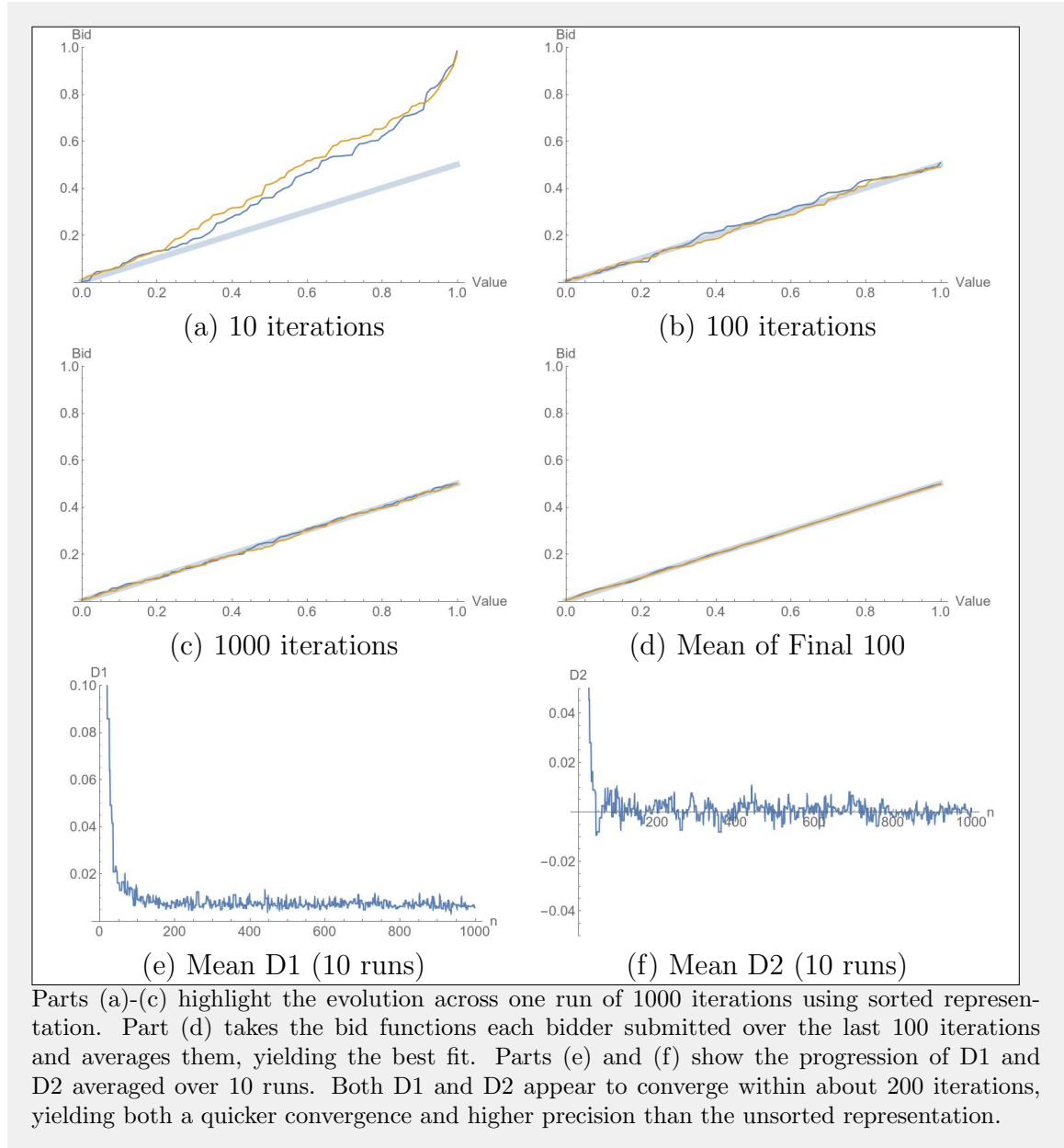


Table 3.1: Accuracy by Submission Method

	4-Tournament	Best	Mode
D1	0.00175	0.00138	0.00246
D2	0.00017	-0.00019	-0.00028

Mean across 10 runs of the mean of the last 100 iterations.

mitting a strategy into the auction each iteration. Intuitively, we can expect that it is optimal to usually submit a high-fitness strategy. Submitting a low-performing strategy results in the other bidders adapting to the anomalous submission, setting back convergence. However, one reason GAs perform well is because they are not simply best-responders [2]. In fact, best response algorithms are notorious for getting stuck in cycles and never reaching an equilibrium. To test which submission technique is best, three methods are compared: a tournament selection with size 4, simply selecting the best strategy, and finally selecting the mode in the population.

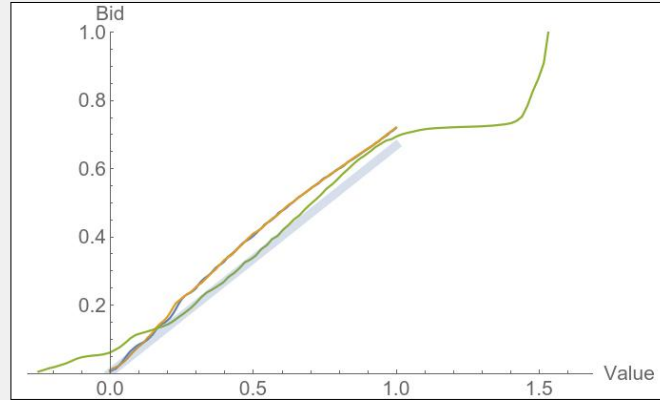
Table 3.1 provides estimates to the performance of each. Each method was averaged over 10 runs. In each run, the bid function over the last 100 iterations was averaged over, and D_1 and D_2 were calculated. Based on the results, simply submitting the best strategy yields the most accurate results, implying that the error caused by submitting poor strategies outweighs any costs of being nearer to naively best-responding. This is likely because even submitting the best strategy within the population is not the same as best responding, since the true best response is unlikely to be a member of the population.

Limitations

Now suppose a new bidder is added whose value distribution is drawn from a normal distribution $X \sim \mathcal{N}(0.7, 0.2^2)$. Similar to before, Figure 3.5 shows the end average convergence. Theory states that bids should be higher than when all three draw from the standard uniform $b_U(v) = \frac{2}{3}v$. This is because one bidder now has a higher value on average, and so the other bidders are facing more difficult competition and up their bids accordingly. This in turn makes the new bidder also bid above the base function.

Although the general behavior is reasonable, there are serious flaws in the result for the normally distributed bidder. At high values, the bid function increases well above the highest possible competing bid, which results in paying more without any

Figure 3.5: Asymmetric Auction Convergence



Averaged over the last 100 iterations. Unlike the uniform distributions, the normal distribution has an infinite support. GENE automatically truncates between the 0.000001 quantile and the 0.999999 quantile. Much of the bid function is over a range of values with only a fraction of a percent chance of being realized, leading to inaccuracies.

increase in win probability. Equally distressing, on the very low end of the value range, the bidder bids above value. What can account for this?

It turns out the expected profit is changed very little by these mistakes. There is only a 0.1% chance of the bidder drawing a value below 0.1, and less than a 1% chance of winning with such values. And so bidding too much changes the expected profit by less than 0.001%. A similar argument can be made on the upper end of the distribution. Compared to getting ϵ better results near the center of the distribution, the tails represent rounding errors. Given a set of strategies, the likelihood that both the tails and center are improved is unlikely, and so the ones with the center are favored as long as the tails ends are not too egregiously wrong.

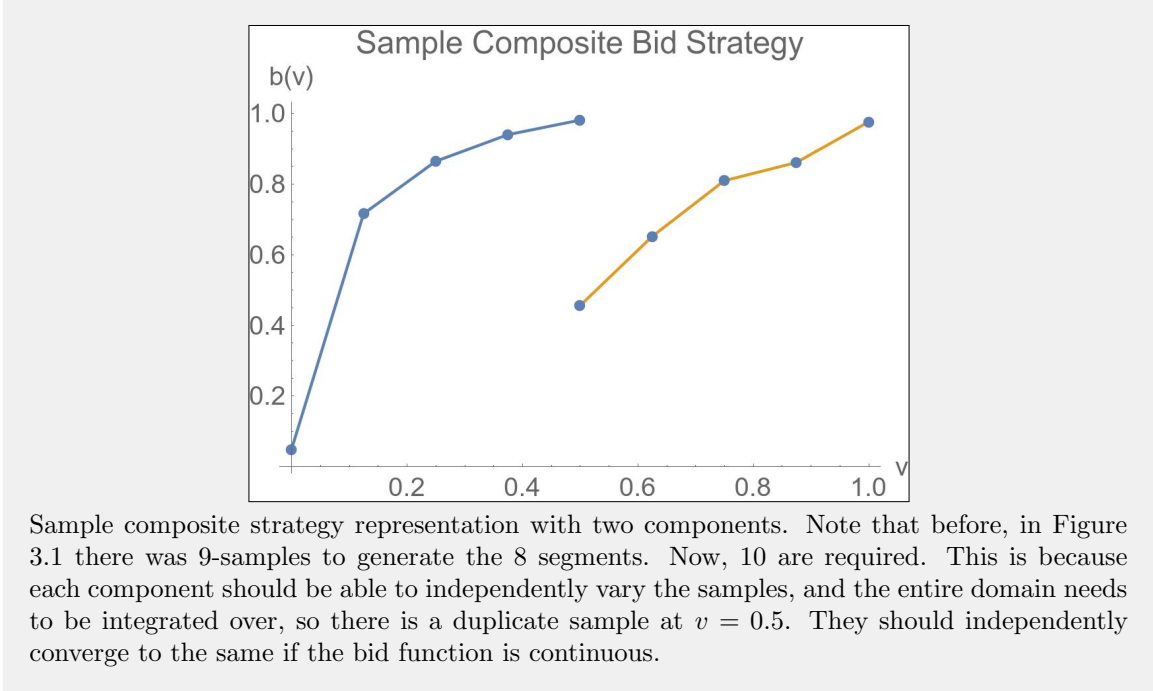
Even though it makes sense to not care about low probability events, we still expect the IEL to be able to solve such problems. It turns out that with some small modifications, it is possible to correct these tail-end values.

3.3.4 Independent Segment Representation

To resolve tail-end inaccuracies, consider the optimization problem faced by bidders again:

$$\max_{b_i} \int_{\underline{v}}^{\bar{v}} (v - b_i(v)) \prod_{j \neq i} [F_{B_j}(b_i(v))] f(v) dv$$

Figure 3.6: Example Composite Bid Function



The integral can be split over m equally spaced segments:

$$\max_{b_i} \sum_{k=1}^m \left[\int_{v+(k-1)L}^{v+kL} (v - b_i(v)) \prod_{j \neq i} [F_{B_j}(b_i(v))] f(v) dv \right] \quad (3.4)$$

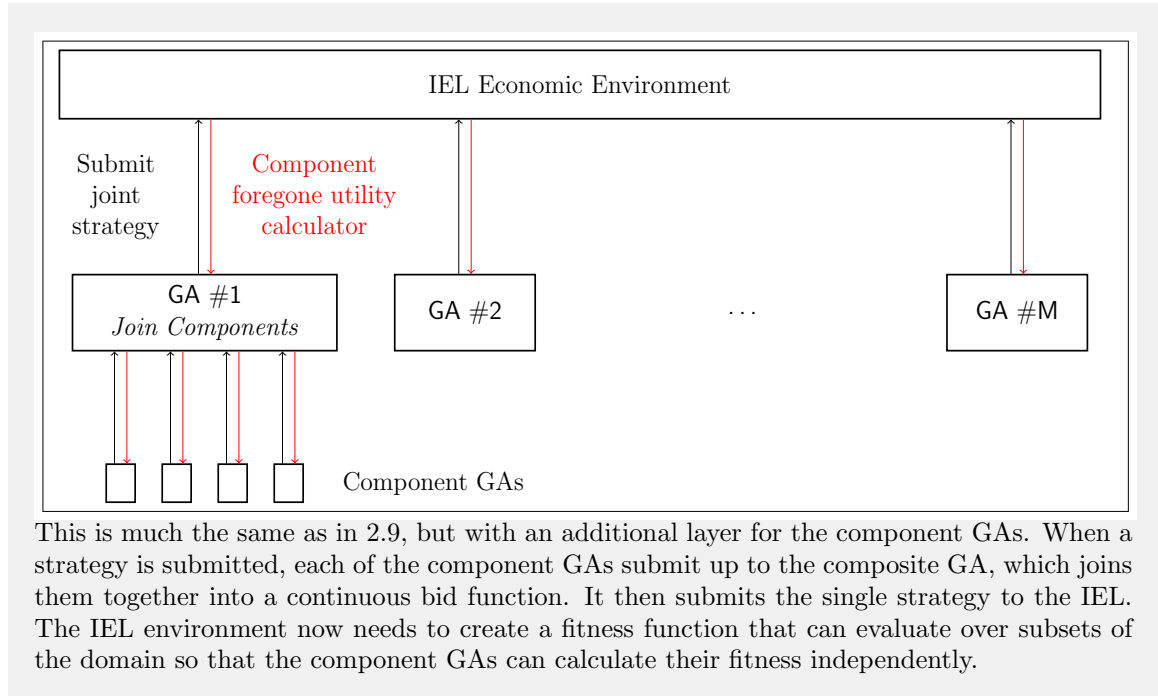
The maximization can then be swapped with the summation:

$$\sum_{k=1}^m \left[\max_{b_{i,k}} \int_{v+(k-1)L}^{v+kL} (v - b_{i,k}(v)) \prod_{j \neq i} [F_{B_j}(b_{i,k}(v))] f(v) dv \right] \quad (3.5)$$

In this form, we now have the bid function split over m segments, $b_1(v), b_2(v), \dots, b_m(v)$, each maximizing profit over a value range. This makes intuitive sense because once a value is realized, the bid should maximize profit for that value independently of the maximization over other value ranges. That means the bid function can be found via many independently optimizing GAs acting together. 3.6 shows how such a bid strategy might look. In this case, there are two GAs, one optimizing the bid function over the first half of the value range, and the other optimizing the other half. Each can sort the bids just as was done in the case with a single GA.

The benefit to doing this is that regions of the value domain with low probability

Figure 3.7: Individual Evolutionary Learning Hierarchy



can be evaluated and optimized independently from the rest of the bid function, so that the profit differentials are not swamped by those in regions with higher likelihood and expected profit.

The component GAs operate just as the individual GAs operated with only a single component. Now, there is an added compositor that joins the components together. Figure 3.7 details the process. The key is that the compositor needs to join the component bid functions together into a single one. While it can leave things discontinuous as seen in Figure 3.6, it is better to stitch them together by taking the mean y-sample when there are two y-samples at the same value ($v = 0.5$ in the figure). Additionally, just as before, the compositor may sort all of them knowing that the final bid function should be monotonically non-decreasing. Similar to the single-component case, sorting does increase convergence accuracy and speed.

Beyond splitting the bid function into regions with more comparable expected profits within each, it also dramatically reduces the search space. Rather than searching in the domain $[\underline{b}, \bar{b}]^n$, there are m components, with each searching in a reduced space of approximately $[\underline{b}, \bar{b}]^{\frac{n}{m}}$. This allows it to converge significantly faster, with smaller populations.

While this would seem to imply more component GAs are always better, in prac-

Figure 3.8: Composite GA Effects on Asymmetric Auction Convergence

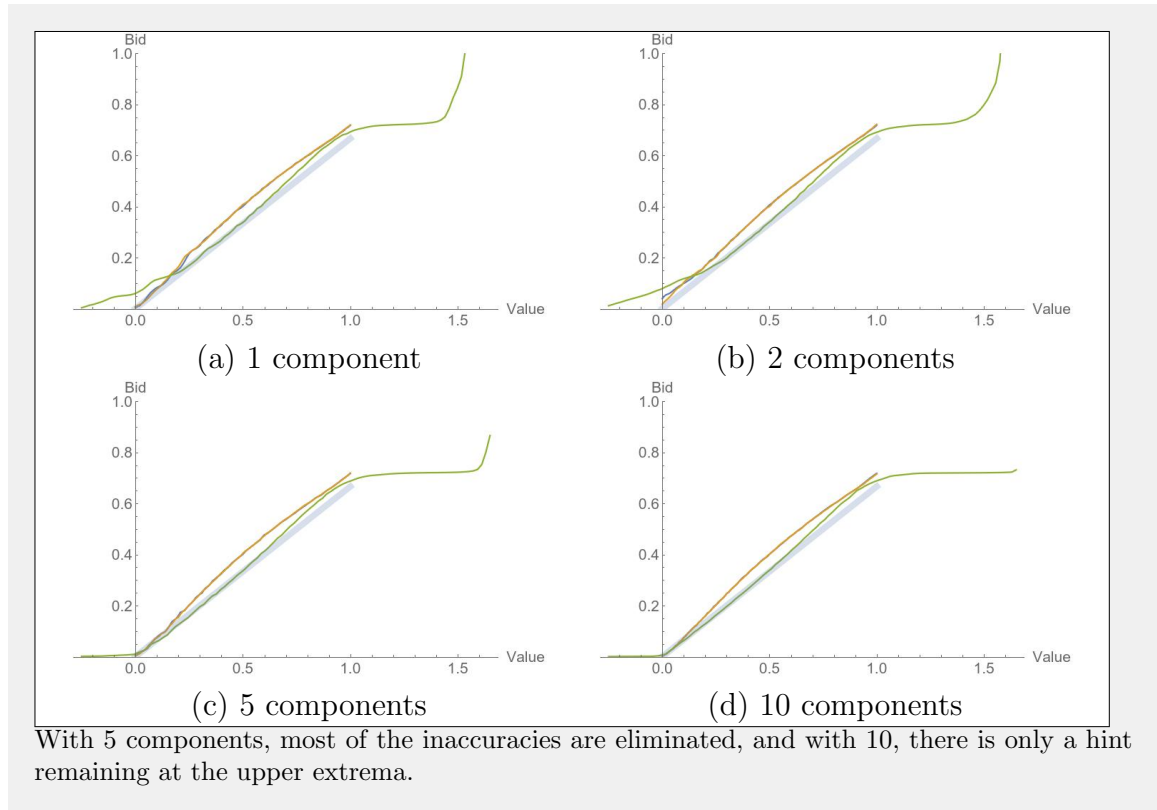
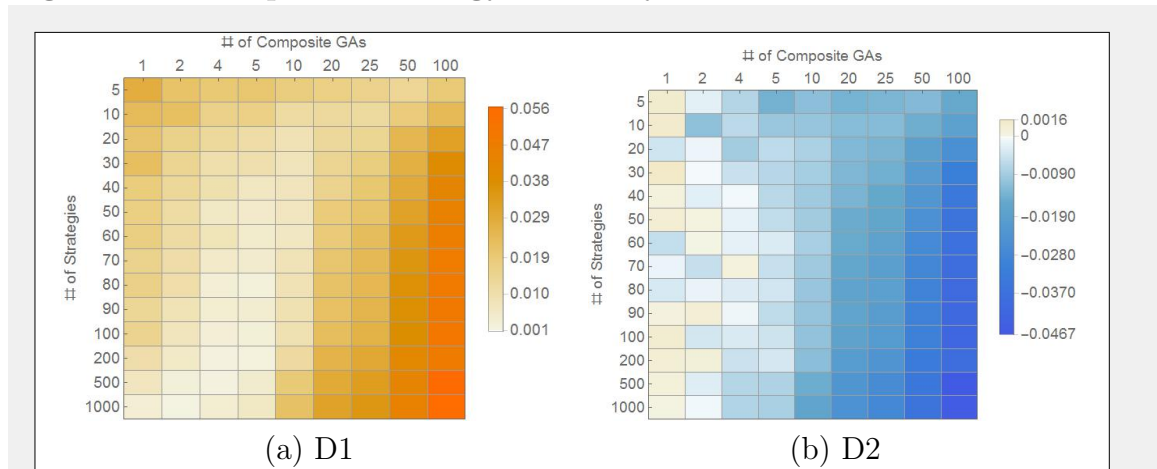


Figure 3.9: Composite-Strategy Accuracy Matrix



Each cell represent a population size-component count pairing. In each scenario, 10 runs were performed, with D1 and D2 being calculated off the average bid function submitted over the last 100 iterations. Lighter colors denote closer to 0, which is better. There is a clear detriment from having either too many components/strategies, or too few, and a basin is apparent.

tice this is not the case. As the number of component GAs increase and the search space diminishes, the component GAs get to be too good, turning into little more than best responders. This leads to cycling and inaccuracies. So, there is a “sweet spot” for the number of components and the number of strategies. Figure 3.9 shows the accuracy for the 2-bidder symmetric first-price auction over a variety of combinations. In practice, it appears having more than 10 components will result in poor convergence.

However, it is not as simple as just picking the combination that has the lowest error in Figure 3.9. Figure 3.8 displays the effects of components on the asymmetric auction accuracy, which was the original motivation for the compositing. Here it is apparent that having additional components does help ameliorate tail-end inaccuracies. Therefore, the choice of number of components depends on the scenario. Luckily, even if imperfect, using 5-10 components with between 50-100 strategies per population proves to be very good in the practice. If higher precision proves necessary, then a simple search can be performed.

3.4 All Pay & Second Price Auction

We now expand our auctions to consider the all-pay sealed-bid auction, where the highest bidder receives the item, but all bidders must pay the bid they submitted, and the second-price auction, where the highest bidder receives the item but only pays the bid of the second-highest bidder. The all-pay is a trivial transform from the first-price auction, where the bid is now subtracted from expected profit in all cases, rather than being conditioned on winning. The second-price auction, although easier to solve theoretically ($b(v) = v$), is actually more complicated to implement computationally.

Suppose we have n draws from the same distribution X . If the draws are sorted from lowest to greatest, the k th lowest draw, the k th order statistic, is denoted $X^{(k)}$.

Its cdf is given by:

$$\begin{aligned}
F_{X^{(k)}}(y) &= \text{Prob}(\text{at least } k \text{ draws } < y) \\
&= \sum_{i=k}^n \text{Prob}(\text{exactly } i \text{ draws } < y) \\
&= \sum_{i=k}^n \binom{n}{i} [F(y)]^i [1 - F(y)]^{n-i}
\end{aligned} \tag{3.6}$$

To evaluate the expected profit in the second-price auction, a bidder needs to know the expected price conditional on winning with a bid. As an intermediary step, the highest order statistic of the opponents' bids is required. However, because the distribution of bids by each bidder is dependent on both the bidder's value distribution and the submitted bid function, both of which can be asymmetric across bidders, the order statistic needs to be calculated for non-identical distributions. Although cumbersome, this can be calculated.

Let C_k^n be the set of all combinations of k selections among n unique objects. Let $c \in C_k^n$ be one such combination. Define c_i to be a binary indicator for whether object i is selected in the combination c . It will be 1 if the object is selected in the combination, and 0 otherwise. Then, for n distributions, X_1, X_2, \dots, X_n , cdf of the k th order statistic $X^{(k)}$ is given by:

$$\begin{aligned}
F_{X^{(k)}}(y) &= \text{Prob}(\text{at least } k \text{ draws } < y) \\
&= \sum_{i=k}^n \text{Prob}(\text{exactly } i \text{ draws } < y) \\
&= \sum_{i=k}^n \sum_{c \in C_k^n} \prod_{j=1}^n c_j F(y) + (1 - c_j) [1 - F(y)]
\end{aligned} \tag{3.7}$$

Given the distribution of the highest competing bid $X_{-i}^{(n-1)}$, a bidder can then calculate the expected second highest bid conditional on them winning with bid y :

$$E(x|x < y) = \frac{\int_{\underline{v}}^y x f_{X_{-i}^{(n-1)}}(x) dx}{\int_{\underline{v}}^y f_{X_{-i}^{(n-1)}}(x) dx} \tag{3.8}$$

With this, the second-price auction can be evaluated.

Table 3.2: Estimated Attributes for Common Auctions

	Bidder Surplus	Revenue	Efficiency
FPA	0.3336	0.3329	99.99%
SPA	0.3333	0.3333	100.0%
APA	0.3318	0.3347	99.98%
(a) 2 Bidders			
	Bidder Surplus	Revenue	Efficiency
FPA	0.1671	0.6662	99.99%
SPA	0.1666	0.6667	100.0%
APA	0.1596	0.6734	99.96%
(b) 5 Bidders			

Estimated auction attributes across 10 runs.

3.5 Results

The three auction types—first-price (FPA), second-price (SPA), and all-pay (APA), were evaluated for environments with known solutions. Each was run with either 2 or 5 bidders, with each bidder drawing values symmetrically from a standard uniform. The convergence can be seen in Figure 3.10.

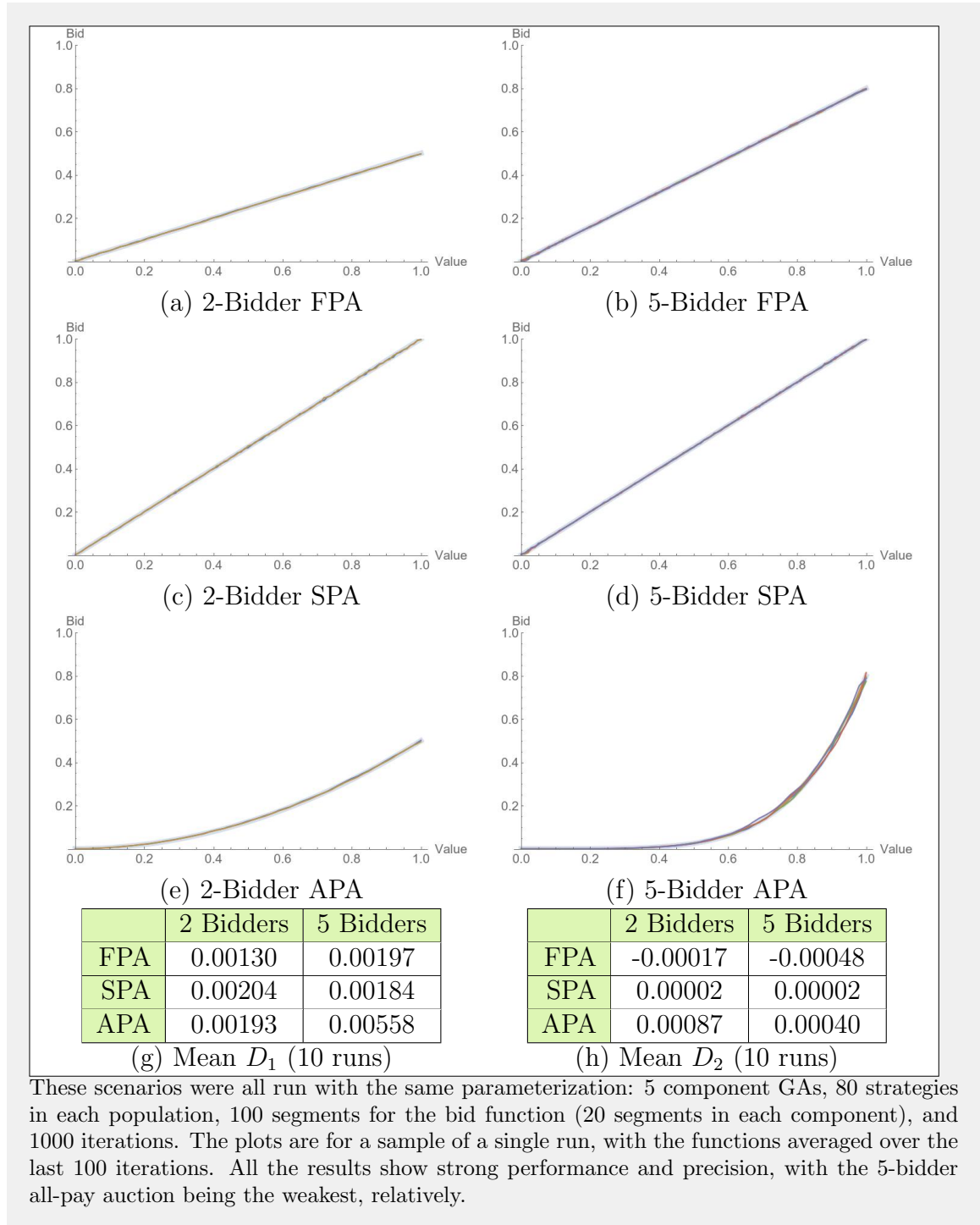
In addition to the bid functions, GENEE can also output the bidders’ profits, their realized values, and the auctioneer’s expected revenue. The estimate for these auction attributes are shown in Table 3.2. They comport with the revenue equivalence theorem [30], which states that changing the auction type will not yield different revenues when bidders are symmetric and the auction mechanism is efficient.

3.6 Conclusions

Building off of French’s work and the IEL library, the GENEE library has been shown to provide highly accurate estimates to auctions with known solutions. The predicted revenues and efficiencies also comport with theory.

Unlike previous attempts however, the GENEE library can handle multi kinds of auctions such as the second-price and all-pay auctions, and can provide estimates for more exotic environments, with non-linear utility functions and subjective probability functions, as well as asymmetric value distributions with differing supports. The flexibility extends past even this, enabling auctions with multiple items and bidders with completely different representations.

Figure 3.10: Convergence for Common Auctions



The remaining chapters detail two applications: simultaneous multi-unit auctions, and common value auctions with multiple signals. Each detail additional extensions required to the model, and showcase the ability of the GENE library to handle a large domain of auction environments.

Chapter 4

Simultaneous First Price Auctions

Consider two items, A and B. The items are to be sold simultaneously in separate first-price sealed-bid auctions. The simplest case is when the item values, v^A and v^B , are drawn independently from one another, and the value received is the sum of the values of the items won. In such a case, the two auction can be treated independently, and they each reduce to the single item case.

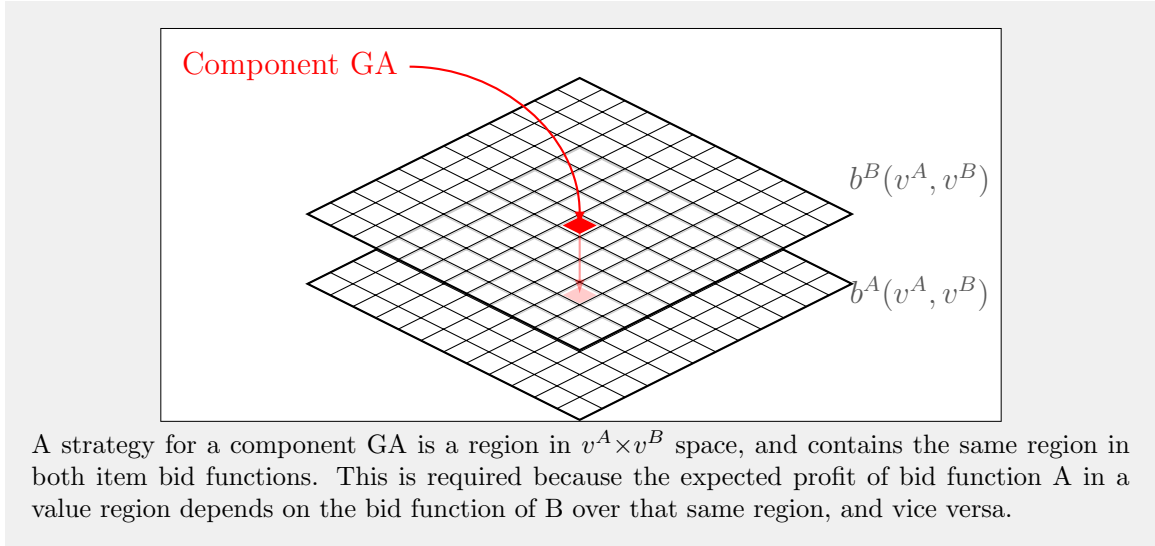
But, often times the items are somehow coupled with one another. In this chapter, I focus on finding bid function equilibria when the items have non-complementary values. In this scenario, the bidder receives v^A in value for winning item A, v^B for winning item B, and $g(v^A, v^B) \leq v^A + v^B$ when both items are won.

There has been considerable research into these auctions. Feldman et al. [15] proved that when values are non-complementary the efficiency of simultaneous first-price auctions is at least 50%, and the efficiency of second-price auctions is at least 25%. Unfortunately, Cai and Papadimitriou [7] were able to prove that finding a Nash equilibrium bid function in first and second price simultaneous auctions is NP-hard. Even worse, they were able to prove even finding an ϵ -approximate equilibrium is NP-hard in the case of second-price auctions¹. While still unproven, it is strongly suspected that first-price auctions suffer the same restriction.

Cai and Papadimitriou aptly summed up the hopelessness of finding the equilibrium bid strategies by writing, “even recognizing a Bayesian Nash equilibrium is intractable.”

¹An ϵ -approximate equilibrium is one that is provably within some distance *epsilon* of the true equilibrium

Figure 4.1: Component GAs in Simultaneous Auctions



4.1 Extending GENE

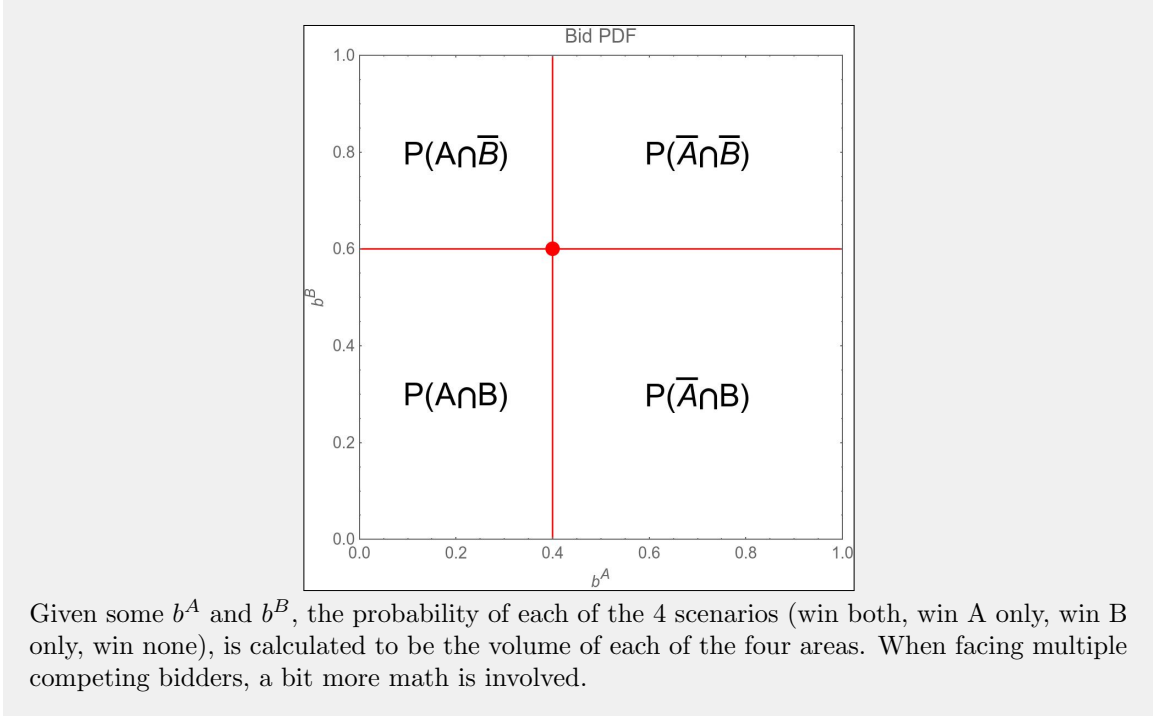
While the problem does quickly become intractable as the number of items being sold increases, it is still manageable with only 2 items. To implement it, GENE must be extended beyond 1-dimensional functions.

4.1.1 Strategy Representation

For the 2-item auction, each bidder has two bid functions, one for item A, and the other for item B. Furthermore, each of the bids depend on both values drawn. Let the bidder i 's bid function be given by $b^A(v^A, v^B)$, and $b^B(v^A, v^B)$. To represent these, a strategy is two, 2-dimensional grids bid samples that are interpolated by bilinear interpolation.

In the single item auction, a 99-segment bid strategy required 100 points. Here, a 99-by-99 segment grid requires 10,000 points. And, because there are two bid functions per strategy, there are ultimately 20,000 points in such a strategy. Fortunately, convergence can be aided by creating component GAs, just as was done in the single-item case. Figure 4.1 displays how the each component GA is akin to a piece of a quilt, and the compositor must stitch the components together into a pair of patchworks (one for each item). Just as in the single-item auctions, the compositor takes the mean of any overlaps, which is just two in most cases, but at the interior intersections there are four component GAs overlapping at a single point.

Figure 4.2: Relevant Areas of Bid PDF



An additional wrinkle compared to the single item environment is sorting the bid function. While a bid function for A should never decrease as the value of A increases, it may decrease when the value of B increases. The equivalent is true for the bid function of B. This means that for each bid function, we may sort the samples in only 1 dimension, and not the other.

4.1.2 Expected Profit Calculations

The expected profit calculations, while computationally cumbersome, are not especially more complicated conceptually than the single-item case. Given player i 's bid functions, let $b_i^A(v^A, v^B)$ and $b_i^B(v^A, v^B)$, let B_i represent the joint distribution of the bids. Given this, its cdf is defined as:

$$F_{B_i}(x, y) = P(b_i^A < x, b_i^B < y) \quad (4.1)$$

Figure 4.2 connects the areas of the bid distribution pdf to the probability of different auction outcomes given a set of bids. The probability of winning both items, $P(A \cap B)$, can be determined by finding the probability of beating all competing

bidders on both items. This is calculated by:

$$P(A \cap B) = \prod_{j \neq i} F_{B_j}(b_i^A, b_i^B) \quad (4.2)$$

Meanwhile, the probability of winning only item A is the probability of outbidding all bidders on A, but not winning B, , $P(A \cap \bar{B})$. It can be calculated with the following:

$$\begin{aligned} P(A \cap \bar{B}) &= (P(A \cap \bar{B}) + P(A \cap B)) - P(A \cap B) \\ &= P(A) - P(A \cap B) \\ &= \prod_{j \neq i} F_{B_j}(b_i^A, \infty) - \prod_{j \neq i} F_{B_j}(b_i^A, b_i^B) \end{aligned} \quad (4.3)$$

The probability of only winning item B can be calculated similarly to only winning item A. Winning no item is just 1 subtracted by the other three probabilities. With all these probabilities, the expected profit can now be calculated²:

$$\begin{aligned} E(\pi(b_i^A, b_i^B)) &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (g(v^A, v^B) - b_i^A - b_i^B) P(A \cap B) + \\ &\quad (v^A - b_i^A) [P(A) - P(A \cap B)] + \\ &\quad (v^B - b_i^B) [P(B) - P(A \cap B)] f(v^A, v^B) dv^A dv^B \end{aligned} \quad (4.4)$$

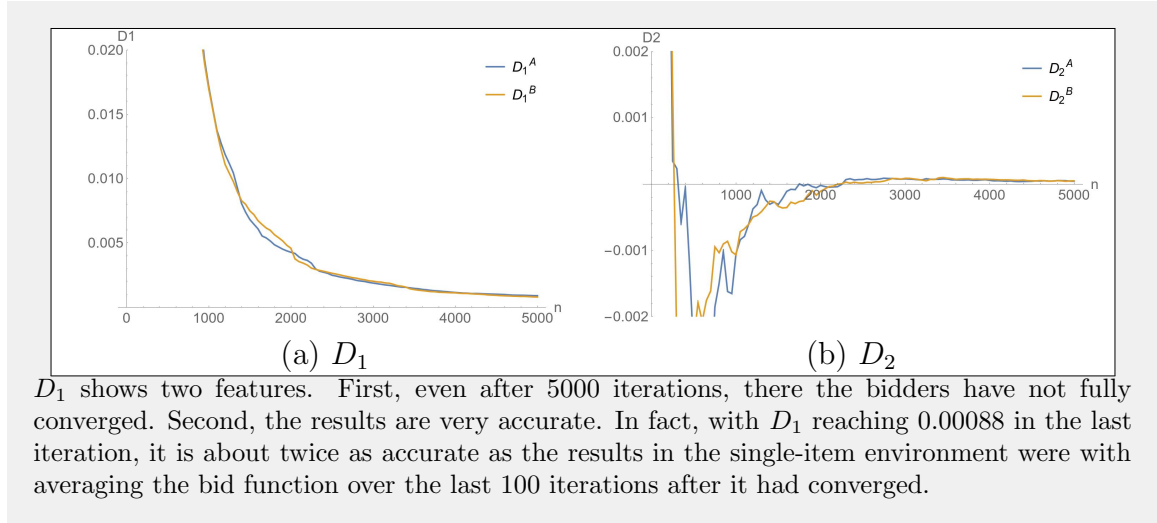
4.1.3 Accuracy Measurement

The final step is to extend the accuracy measurements, D1 and D2 to handle 2-dimensional bid functions. In this case it is a simple matter of replacing the single integral with a double integral. Additionally, each bid function, $b^A(v^A, v^B)$ and $b^B(v^A, v^B)$, will be measured independently.

$$\begin{aligned} D_1^j &= \frac{1}{(\bar{v}^A - \underline{v}^A)(\bar{v}^B - \underline{v}^B)} \sqrt{\int_{\underline{v}^A}^{\bar{v}^A} \int_{\underline{v}^B}^{\bar{v}^B} (b^j(v^A, v^B) - \hat{b}^j(v^A, v^B))^2 dv^A dv^B} \\ D_2^j &= \frac{1}{(\bar{v}^A - \underline{v}^A)(\bar{v}^B - \underline{v}^B)} \int_{\underline{v}^A}^{\bar{v}^A} \int_{\underline{v}^B}^{\bar{v}^B} (b^j(v^A, v^B) - \hat{b}^j(v^A, v^B)) dv^A dv^B \end{aligned} \quad (4.5)$$

²Note that the bids are both functions of the values, but the notation was suppressed for clarity.

Figure 4.3: Accuracy for Additive Values



D_1 shows two features. First, even after 5000 iterations, there the bidders have not fully converged. Second, the results are very accurate. In fact, with D_1 reaching 0.00088 in the last iteration, it is about twice as accurate as the results in the single-item environment were with averaging the bid function over the last 100 iterations after it had converged.

4.2 Results

Recall that it is assumed bidders receive the value for the item if only 1 item is one, but receive $g(v^A, v^B)$ if both items are won. Three different non-complementary value functions were explored:

$$\begin{aligned}
 g(v^A, v^B) &= v^A + v^B \\
 g(v^A, v^B) &= 0 \\
 g(v^A, v^B) &= \max(v^A, v^B)
 \end{aligned}
 \tag{4.6}$$

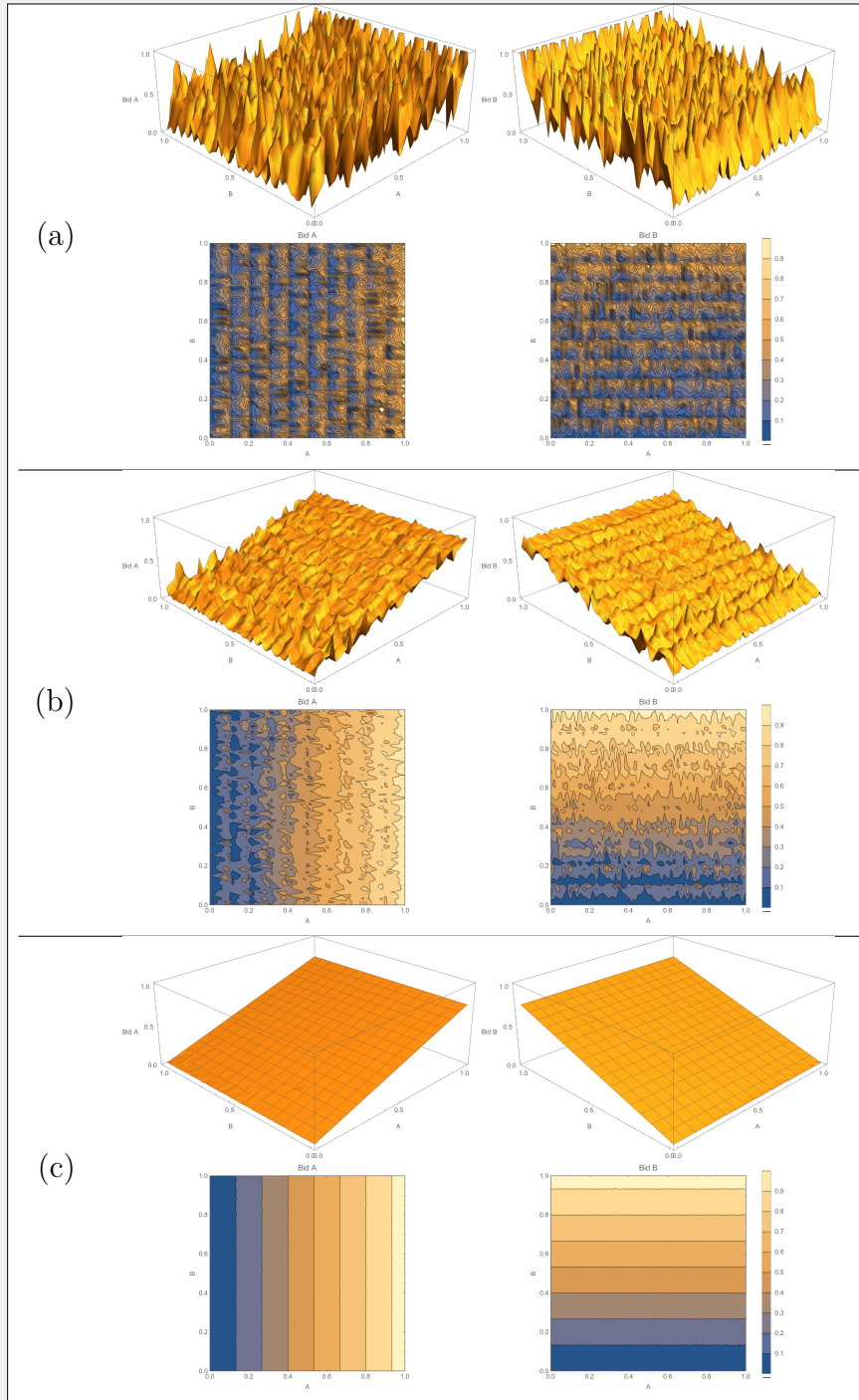
The additive valuation serves as a baseline and will be addressed first.

4.2.1 Additive Values

When $g(v^A, v^B) = v^A + v^B$, the items are independent of one another, and so we should expect each bid function to be the same as single-item auctions, with no dependence on the other item's value. Figure 4.4 shows the evolution of a 4-bidder auction over 5000 iterations, and 4.3 shows the accuracy measurements. The results prove to be highly accurate, if computationally expensive³.

³Executing this 2-item auction requires about 100 times as much computational time as the single-item auctions.

Figure 4.4: Convergence for Additive Values



One bidder's functions in a 4-person environment, with each drawing values from the standard uniform. Displays are after (a) 50 iterations, (b) 500 iterations, and (c) 5000 iterations. After 50 iterations, the patchwork nature of the bid function is evident, as well as the discriminatory nature of only sorting in the A direction for $b_i^A(v^A, v^B)$ and vice versa for bid on B. By the end, the patchwork is unidentifiable, and the system has evolved to have each bid function be independent of the drawn value for the other item.

4.2.2 Dating Game

Now suppose $g(v^A, v^B) = 0$. This means that if a bidder wins one item, they receive it's value, but if they win both, they receive nothing. Such a function is admitted contrived, but a plausible scenario is that of dating. Expending resources in the pursuit of multiple lovers can be conducted, but in this model, being successful with both ultimately leads to rejection by both.

Intuitively, we would expect the bidder to focus on item with the higher value draw, and bid low enough to not win the other item, perhaps even going as far as bidding nothing on it. A naive approximation would be:

$$b_i^A(v^A, v^B) = \begin{cases} \prod_{i=1}^{n-1} \left[\frac{i+1}{i+2} P(i) \right] v^A & \text{if } v^A > v^B \\ 0 & \text{if } v^A \leq v^B \end{cases} \quad (4.7)$$

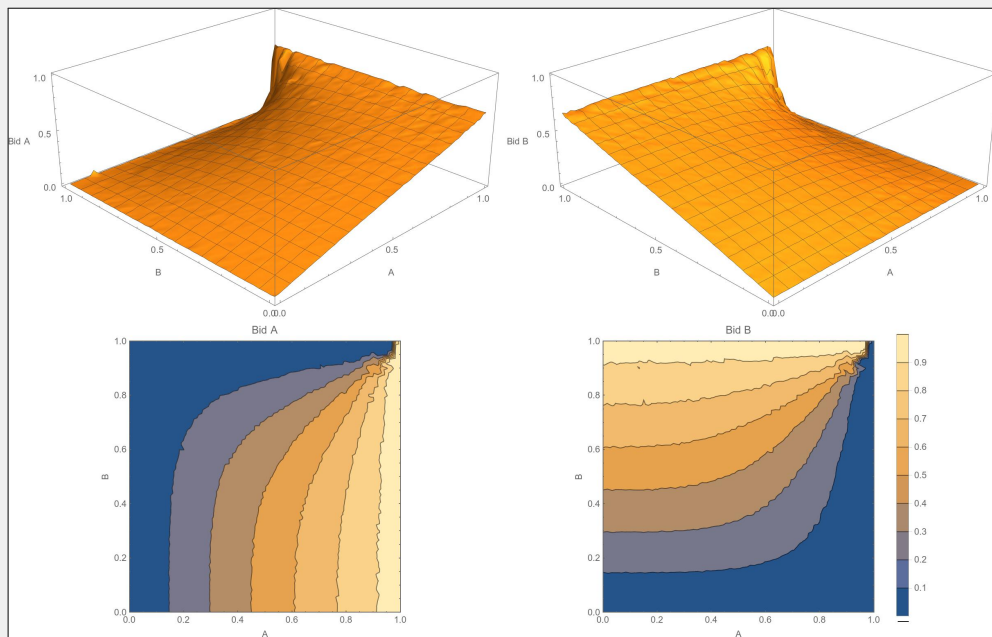
Which just says to only bid when the value for item A exceeds that of item B, and when bidding, bid a average of the single-item equilibria weighted by the probability of that many other bidders choosing to bid on item A. With symmetric uniform draws and 4 bidders, this works out to be:

$$b_i^A(v^A, v^B | v^A > v^B) = \frac{101}{160} v^A \approx 0.63v \quad (4.8)$$

This is just a first-order naive approximation. The actual solution remains an open question in the literature. Figure 4.5 shows what GENEe converges to in the 4-bidder scenario. It finds that the bidders do bid on both items sometimes, as long as they believe there is a significant probability of losing the item they value more. In fact, it is observable that conditioned on the value of item A, the bid on A is strictly non-increasing with value for B. This means that as their value for B increases, and thus they bid more on item B, the probability of winning B increases, and in turn the expected profit from winning item A decreases, finally leading to a decrease in the bid on item A.

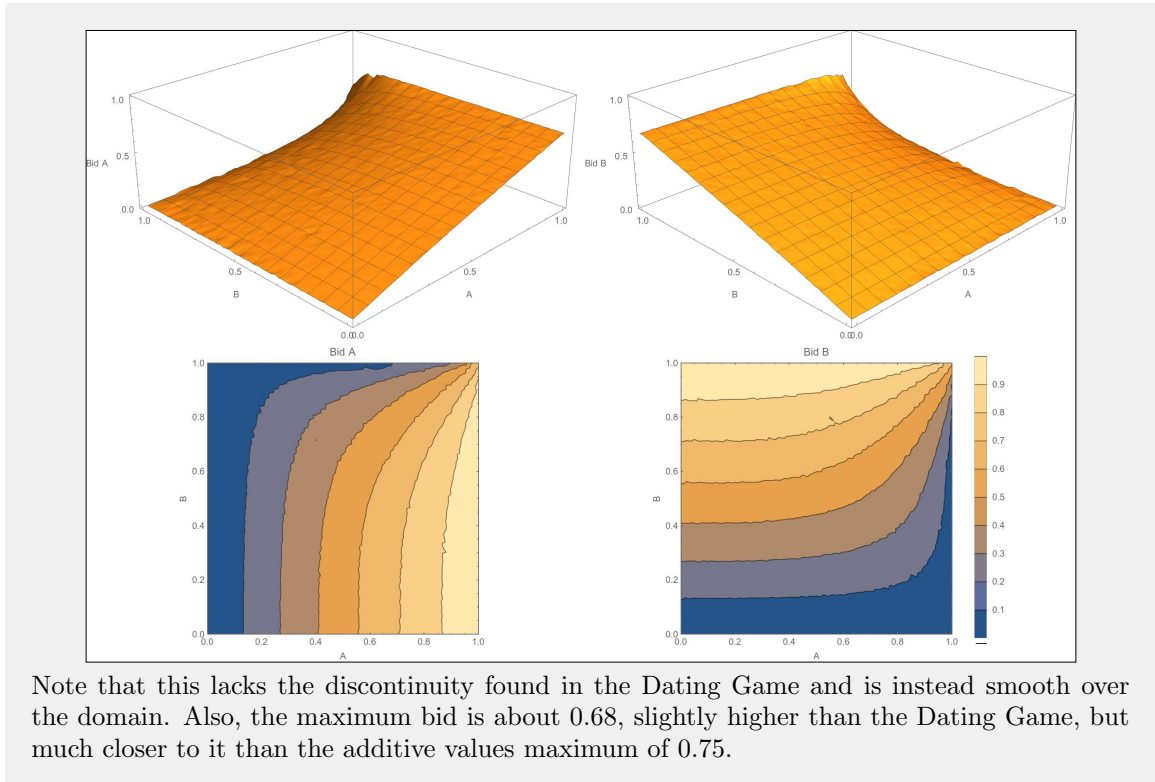
The maximum bid is 0.67, slightly higher than the naive estimate, which is expected given that bidders sometimes bid on both item, leading a higher expected number of competitors.

Figure 4.5: 4-Bidder Dating Game



A particularly notable result is the discontinuity around $v^A = v^B = 1$. Currently, GENEE does not support discontinuities. The best it can do is drop from 1 to 0 across a single segment, which looks similar. With this simulation having 60-by-60 segments, the quickest it could switch between items is over a $\Delta v = \frac{1}{60} \approx 0.01667$.

Figure 4.6: 4-Bidder with Perfect Substitutes



4.2.3 Perfect Substitutes

Finally, suppose $g(v^A, v^B) = \max(v^A, v^B)$. In this case, the items are perfect substitutes, and the bidder only receives value equal to the most valuable item. This is perhaps the best approximation of some of the bidders in the New Zealand Spectrum auction. Many only need one spectrum license, and so winning multiple wastes money. Once again, there is no theory on how to bid in such an environment, but it should lie somewhere between the additive values and dating game behavior.

Figure 4.6 shows the result from GENE. Indeed, it does appear to fall between the two more extreme value functions.

4.2.4 Estimated Auction Attributes

While the bid behavior in different scenarios is interesting in and of itself, of greater interest are estimates for the auction properties such as efficiency and auctioneer revenue. From the model, revenue, profit, and realized value can be directly extracted. By running Monte Carlo simulations to determine the expected maximum attainable value, efficiency can be calculated.

Table 4.1: Estimated Auction Characteristics

(a) Efficiency	# Bidders	Dating	Substitutes	Additive
	2	0.97	0.98	1.00
	3	0.96	0.97	1.00
	4	0.97	0.98	1.00
	5	0.98	0.98	1.00
	6	0.98	0.99	1.00
(b) Revenue fraction	# Bidders	Dating	Substitutes	Additive
	2	0.11	0.28	0.50
	3	0.45	0.53	0.67
	4	0.61	0.66	0.75
	5	0.73	0.73	0.80
	6	0.76	0.78	0.83

(a) The efficiency of the mechanism across the different value parameterizations changes very little actually confirms that the mechanism is perhaps a good choice for subadditive values. (b) The fraction of the surplus the auctioneer receives through bids strongly depends on the revenue, and interacts with the parameterization. As the number of bidders approach the number of items, the auctioneer can expect very little revenue depending on how subadditive values are.

Table 4.1 gives estimates to the efficiency for the three parameterizations discussed while varying the number of bidders from 2 to 6. It also shows what fraction of the realized value is collected by the auctioneer from bids.

From these, it is apparent that very little efficiency is lost in any of the 2-item environments, but the loss is non-zero. The loss occurs in the subadditive parameterizations during situations such as when the buyer with the highest value on A has an even higher value on B, but not the highest, and so underbids on A, and loses it to someone with a lower value for it, but no value for B.

Auctioneer revenue suffers greatly as the number of bidders approach the number of items, becoming even lower as valuations for the pair decrease. This might provide some insight into the revenue shortcomings of the New Zealand spectrum auctions. A lack of bidders resulted in lower and perhaps fewer bids to avoid accidentally over-purchasing.

4.3 Conclusions

Extending GENE to support 2-dimensional bid functions empowers it to tackle auctions with scant theoretical predictions. It was able to provide point estimates to the auction performance where none was previously available, and could only have been estimated through experiment data.

This also provides a base that can be built on to examine not just simultaneous second-price auctions, but also auctions with complementary values and the submission of combinatorial bids (a bid which demands either both items or none).

Chapter 5

Common Value Auction with Multiple Signals

Consider an auction to lease a gold mine. The gold mine has the same value, v , to all bidders, but each bidder has a different signal, s , as to the mine's value. How should they bid? First, suppose each signal is unbiased, such that $E(v|s) = s$. Additionally, assume all bidders simply submit their expected value as their bid. It turns out that even though each bid is an unbiased estimate, the highest estimate among them is biased upwards. Because of this, the winner of the auction will on average overpay for the lease. This phenomena is known as the Winner's Curse, and is prevalent in practice. To avoid it, bidders must recognize that $E(v|s = s_{\max}) < E(v|s)$, and adjust their estimate of, v , downwards by accounting for this fact that winning implies their estimate was highest.

5.1 Single-Signal Scenario

To give the scenario concreteness, suppose the true value, v , is drawn uniformly in $[500,9500]$. Each bidder receives a single signal, s , of the true value with an added error ϵ drawn uniformly in $[-500,500]$. The goal is to find the equilibrium bidding strategy as a function of the signal, $b_i(s)$.

Wilson first solved for the bid function [34], and Milgrom and Weber [29] as well as Levin et al. [26] developed it further. Before turning to more complicated variants, it will be instructive to use GENE to reproduce the theoretical predictions and ensure it works in this auction environment.

The expected profit of a bidder is given by:

$$E(\pi(b_i(s))) = \int_{500}^{9500} \int_{-500}^{500} (v - b_i(v + \epsilon)) \prod_{j \neq i} B_j(b_i(v + \epsilon)) f_V(v) f_\epsilon(\epsilon) d\epsilon dv \quad (5.1)$$

While this integral can be evaluated easily enough, it cannot be split up in the signal space, a requirement for running GENE with component GAs. To remedy this, the integral is rewritten as follows:

$$\begin{aligned} E(\pi(b_i(s))) &= \int_{500}^{9500} \int_{-500}^{500} (v - b_i(v + \epsilon)) \prod_{j \neq i} B_j(b_i(v + \epsilon)) f_V(v) f_\epsilon(\epsilon) d\epsilon dv \\ &= \int_{500}^{9500} \int_0^{10000} (v - b_i(s)) \prod_{j \neq i} B_j(b_i(s)) f(v, s) ds dv \\ &= \int_0^{10000} \int_{s-500}^{s+500} (v - b_i(s)) \prod_{j \neq i} B_j(b_i(s)) f(v, s) ds dv \end{aligned} \quad (5.2)$$

This integral has the signal, s , only vary through the outer integral, and allows for a region of the bid function to be evaluated independently of other regions. Also note that while this requires double integration, the bid function is only 1-dimensional, so while some of the procedures developed in chapter 4 are useful, the process is more akin to that seen in chapter 3.

Figure 5.1 shows the GENE output for the single-signal scenario with 4 bidders. The algorithm successfully models all regions except the lower endpoint. This is solved by avoiding it. For the remainder of the chapter, the model will focus on solving the center region of the bid distribution. This has been done elsewhere in the literature [21], and allows for a much simpler model. In the center signal region, the relative bid, ($r_i(s) = b_i - s$), reduces to a constant, since the actual value conveys no special information.

It is important to note that this reduction only holds because of our assumption of operating sufficiently far from the endpoints. At the endpoints, some bidders may receive signals outside the value range, allowing for a more precise estimate of the value. Bidders with signals near the value endpoints must adjust their bids to account for the possibility of competing bidders receiving these more informative signals. This then propagates to bidders further from the endpoints who react to these changes in bidding. This is visible in Figure 5.1 (b), where the signal region $s > 1000$, in which the true value cannot be at the endpoint, exponentially decays towards discounting

Figure 5.1: Convergence for Single-Signal Scenario

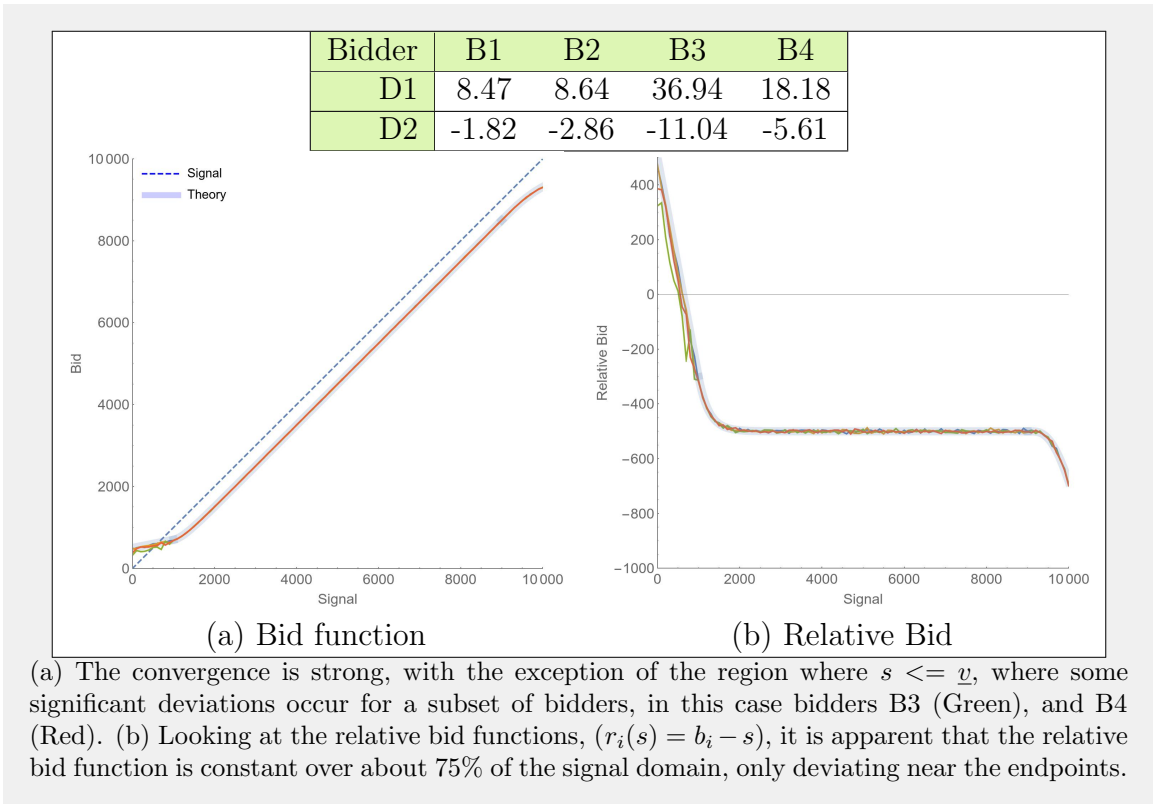
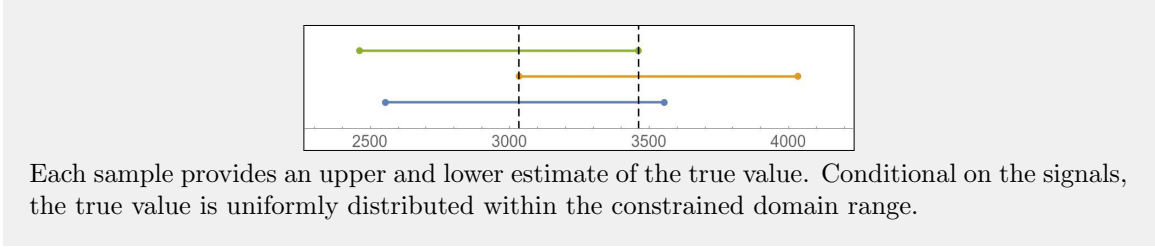


Figure 5.2: Sample Signals



the observed signal by a fixed amount of 500. Note that this fixed discount region not only avoids the Winner’s Curse in expectation, but actually reduces its possibility of occurring to zero.

With this admittedly strict assumption of operating near the center of the signal domain, the bid function reduces to a single relative bid value, r_i , and the expected profit equation can be reduced to the following:

$$\begin{aligned}
 E(\pi(r_i)) &= \int_{-500}^{500} (0 - (r_i + \epsilon)) \prod_{i \neq j} F_{B_j}(r_i + \epsilon) f(\epsilon) d\epsilon \\
 &= \int_{-500}^{500} (0 - (r_i + \epsilon)) \prod_{i \neq j} \frac{(r_i + \epsilon) - r_j + 500}{1000} f(\epsilon) d\epsilon
 \end{aligned} \tag{5.3}$$

Notice that because the signal value does not affect the relative bid, the expected profit is reduced to only integrating over the domain of possible errors, and the true value can be normalized to zero. This simplified version is easy to solve for by modifying GENE to support estimating point bids.

5.2 Multiple Signal Scenario

Now suppose a more realistic environment where bidders must pay for signals, rather than being endowed with them exogenously. Prior to observing any signals, a bidder must decide how many signals to purchase between 1 and 5. Each signal is still drawn with an error uniformly distributed in $[-500, 500]$.

As established in Cox and Haynes [10], the sufficient statistics for multiple draws from a uniform are the minimum and maximum draws, and draws between the minimum and maximum provide no additional information. Figure 5.2 shows an example of a bidder receiving three signals. The signals only set the bounds for the true value, and the true value is still uniformly distributed within the constrained range.

Therefore, the unbiased estimate for the value given the signals is the midpoint, $m = (s_{\max} + s_{\min})/2$, of the max and min draws. The range the true value can take can be determined from both m , and the precision of the draws, $p = (s_{\max} - s_{\min})$, with a higher precision translating to a smaller possible range of values the true value can take.

In the general case, bids are a function of both the precision and midpoint, $b_i(m, p)$. However, from the assumption previously stated that we are far from either endpoint, the relative bid function, $r_i(m, p) = b_i(m, p) - m$, does not depend on the midpoint, reducing to just $r_i(p)$. Given this, we can write the expected profit equation as follows (normalizing the midpoint to 0):

$$E(\pi(r_i(p))) = \int_0^{1000} \int_{-500}^{500} (0 - (r(p) + m)) \prod_{j \neq i} F_{B_j}(r(p) + m) f(m, p) dm dp \quad (5.4)$$

The joint pdf of the midpoint and precision, $f(m, p)$, can be calculated from the joint order statistic of the min and max draws. Figure 5.3 (a) gives the marginal distribution of the precision as a function of the number of signals purchased, and Figure 5.3 (b) and (c) show the joint pdf for signal draws of 3 and 5. As stated earlier, conditional on a precision or range, the midpoint is uniform.

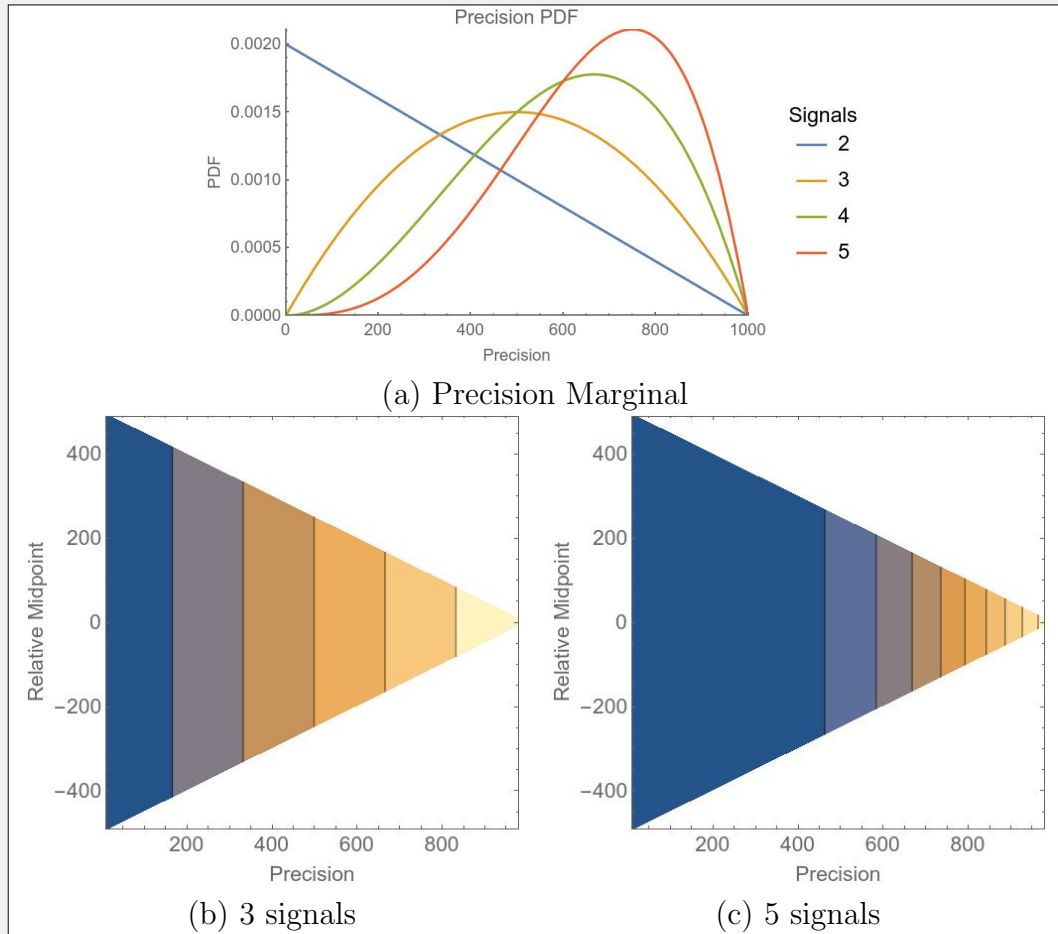
5.3 Model Results

With the expected profit equation defined and the joint distributions known for given number of signals, the relative bid functions, $r_i(p)$ can be estimated for m bidders, conditional on the number of signals purchased by each bidder, S_i . Given $m = 4$ bidders, there are 70 unique signal scenarios (all bidders have 1 signal each, 3 have 1 and the last has 3 signals, etc.).

Figure 5.4 shows the results of two representative signal scenarios. In no scenario does a bidder submit a bid that has the potential of experiencing the Winner's Curse and yielding negative profit. Instead, bids always track with the lowest possible true value at low precisions, with bid shading occurring as precision increases. The magnitude of this bid shading decreases as the number of signal draws competing bidders purchased increases.

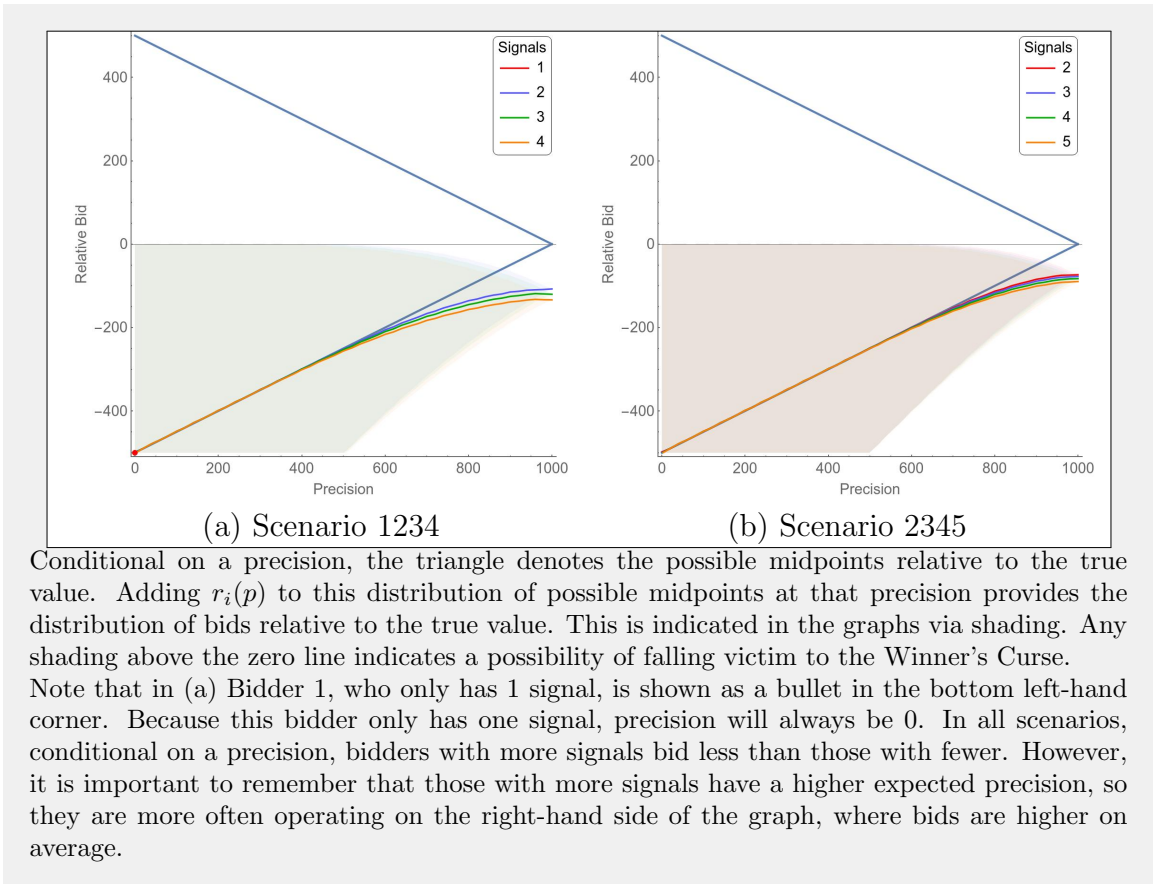
After running GENE for all 70 signal scenarios, the value of signals can be estimated. In each signal scenario, the profit of each bidder is estimated. If increasing

Figure 5.3: Precision-Midpoint Distributions



(a) With only 1 signal, the precision will always be zero. Otherwise, the precision follows a beta distribution, $\beta(n - 1, 2)$, where n is the number of signals. (b-c) The joint distribution has the midpoint normalized about 0. All midpoints are equally likely conditional on the precision, but without conditioning, midpoints nearer to the true value are more likely.

Figure 5.4: Model Bidding



the signals purchased by 1—while holding all other bidder signal purchases constant—increases the expected profit of the bidder by more than the cost of the signal, the bidder would prefer to have purchased an additional bid. The reverse is true when determining whether the bidder would have preferred to have purchased fewer signals. If no bidder would have preferred to change from the purchased number of signals, then that signal scenario is an equilibrium.

Figure 5.5 shows each bidder’s preference in all 70 scenarios for three levels of signal costs, 3, 5, and 10. Each price has a single pure-strategy Nash equilibrium:

- Signal cost 3: 3 bidders purchase 3 signals, 1 purchases 4 signals.
- Signal cost 5: 3 bidders purchase 3 signals, 1 purchases 2 signals.
- Signal cost 10: all 4 bidders purchase 2 signals.

5.4 Experiment

An experiment was performed to evaluate people’s behavior in the environment. Each auction had 4 bidders, values and signals were drawn in accordance to the model denominated in cents. Therefore, the minimum value in an auction was \$5.00, and the maximum possible value was \$95.00. Signals were within \$5.00 of the true value in either direction. The cost of a signal was set to 10 cents.

5.4.1 Procedure

A full description of the procedure can be found in Deck et al. [12]. Briefly, each session had 12 subjects. Subjects competed in 30 auctions and were paid their cumulative earnings. Subjects were given an endowment of \$25.00. Auction losses and the costs of purchased signals were deducted from the endowment. Subjects received a \$7.00 participation payment in addition to their earnings, which averaged \$12.13¹.

At the start of each auction, subjects were randomly and anonymously placed in groups of 4. First, each bidder simultaneously decides how many signals to purchase. Next, each bidder was informed of the information contained in their own signals and the number of signals acquired by the other 3 bidders in the auction (but not the

¹Note that purchasing the minimum 1 signal each auction and otherwise not participating nets \$22.00 in earnings.

Figure 5.6: Subject Bidding

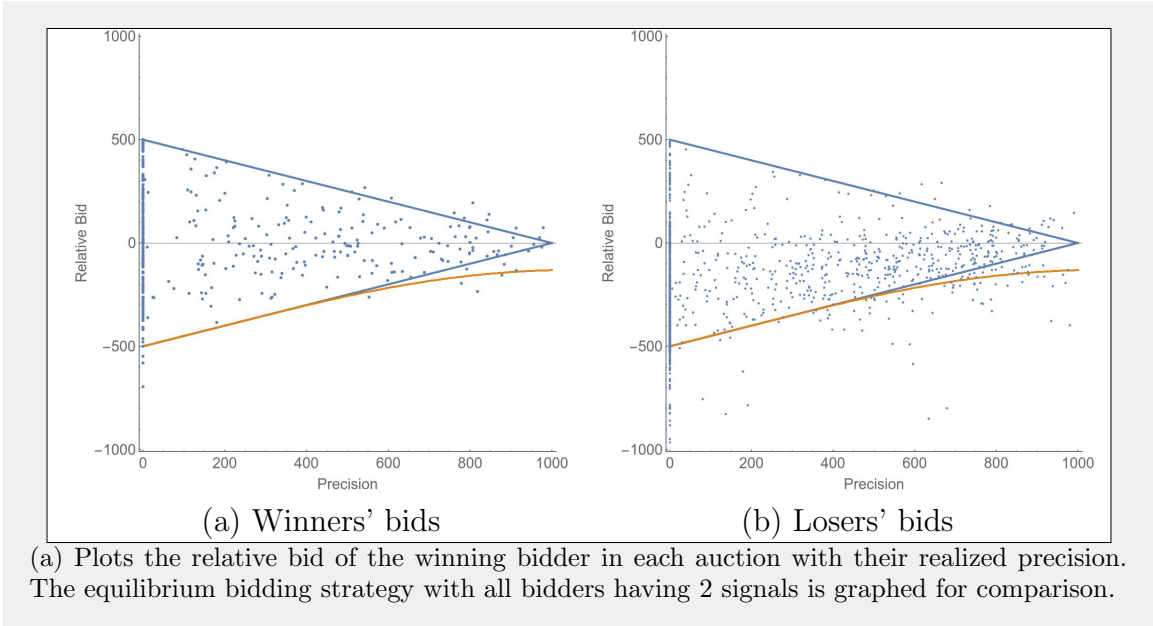


Table 5.1: Realized profit of Winners

Signals	1	2	3	4	5
Frequency	213	42	91	47	27
Avg. Profit	-278.2	-162.6	-133.7	-120.1	-144.7

The average winner fell prey to the Winner's Curse and lost substantial sums. Interestingly, due to their propensity towards overbidding, the value of signals is substantially higher than the model suggested.

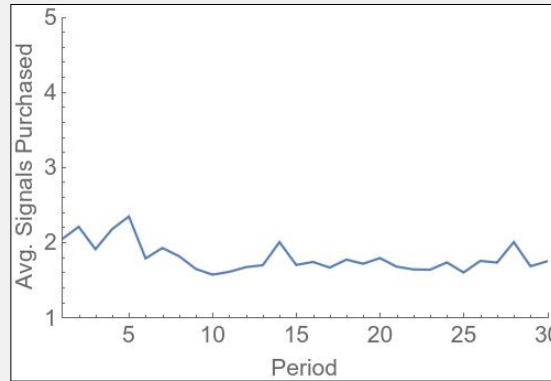
realization of those signals). Then the actual auction occurred. Finally, the auction results were displayed to everyone as was the true value of the asset to enhance transparency.

5.4.2 Results

The auctions were filtered to those with true values between 1000 and 9000. Figure 5.6 shows how subjects bid in the auction. Both eventual winners and losers risked falling prey to the Winner's Curse a majority of the time. On rare occasions, bidders even placed bids above the maximum possible value based given their signals, even though the experiment's interface provided the allowable range of the true value given signals.

Table 5.1 displays the average profits of the winners in each auction (Losers just

Figure 5.7: Signal Purchasing



The average number of signals bought remains relatively constant over the horizon, with an average of about 1.7 signals purchased per bidder per auction.

incur the cost of the signals purchased). The Winner’s Curse proved to be ever-present, in line with previous experiments.

Conditional on winning, signals increased expected profit significantly more than was predicted in the model. However, this does not mean we would expect subjects to purchase more signals than originally predicted. To see why, consider a rational bidder in this auction, armed with Table 5.1 and Figure 5.6. Given that the winning bid is usually above the true value, the expected probability of winning at a price below the true value is rather low, leading the rational bidder who does not overbid to “lose” most of the time. Given such a situation, signal purchasing is largely a waste of money for such a bidder.

Bidders who do win may or may not benefit from signal purchasing. Clearly, conditional on winning, signals help stem the bleeding. However, purchasing additional signals could lead to lower overall profit if it increases the likelihood of winning. Ultimately, Figure 5.7 shows the average number of signals purchased each auction, and the average remains steady across time at about 1.7 signals per bidder, per auction, not far in aggregate from the Nash equilibrium of 2 signals per bidder.

5.5 Conclusions

The GENE library proved invaluable in evaluating such a complex environment. Not only was it able to provide estimates to the equilibrium bid functions, but it was also able to estimate the values of signal purchasing, leading to the solving of the

meta-equilibrium of signal purchasing.

A clear next step is to relax the assumption that the true value is far from the endpoints. Handling this more general case allows for more precise estimates of the value of signals, and for more observations from the experiment to be analyzed. However, it remains to be seen whether the expected profit can be calculated quickly enough to be practical, or if the functions can converge reliably at the lower endpoint.

Chapter 6

Conclusions

I have created a computational program that provides significant value to the field of mechanism design. It contributes a generalized implementation for both genetic algorithms and individual evolutionary learning that can be reused for other applications. More importantly, it provides a robust library of tools to evaluate many forms of auctions, both common and more exotic.

Furthermore, it allows for more complex bidder behavior, with asymmetric value draws, non-linear utility functions and subjective probability functions allowable. This alone opens the doors to computational investigations of auction domains that were previously impossible to explore in any systematic way.

Because the computational evaluation of equilibrium bid functions has remained largely unexplored beyond that of simple asymmetries in first-price auctions, the avenues for future research is vast. The most obvious routes would be to apply it to more parameterizations, such as superadditive values in the simultaneous first-price auction environment. Another route is to extend GENE to work with more auction mechanisms, such as randomized auctions, where the auctioneer chooses a mechanism (first-price, second-price, all-pay, etc.) at random after bids are submitted. Another possible auction extension would be the simultaneous second-price sealed bid auction, or to allow combinatorial bids in all the simultaneous auctions.

Besides extensions, there are also many potential routes to improve the underlying library. Auction processing speed can likely be improved by shifting some calculation to the GPU, and special case algorithms may exist that can improve evaluation of certain auctions. But beyond just speed enhancements, some options I passed on, such as cubic interpolation and Monte Carlo evaluation of the expected profit function, are

still worth being investigated. An even more dramatic change would be to replace the Individual Evolutionary Learning backend with an alternative such as adversarial neural networks. Such changes would have dramatic effects on convergence speed and accuracy, possibly for the better. A final possible change would be to add hooks for the library into higher-level programming languages such as Python to increase accessibility to other researchers.

Even without these potential extensions and improvements, the tool stands on its own, having already provided unique results that no alternative method can currently generate, including the simultaneous first-price auction and the common value auction with asymmetric signals. The fact the same algorithm can estimate the Nash equilibrium bid functions in these vastly different scenarios is a testament to its flexibility, and bodes well for its continued use in the field.

Bibliography

- [1] J. Andreoni and J. H. Miller. “Auctions with Artificial Adaptive Agents”. In: *Games and Economic Behavior* 10.1 (1995), pp. 39–64. ISSN: 0899-8256.
- [2] J. Arifovic. “Genetic algorithm learning and the cobweb model”. In: *Journal of Economic Dynamics and Control* 18.1 (1994), pp. 3–28. ISSN: 0165-1889.
- [3] J. Arifovic and J. Ledyard. “A behavioral model for mechanism design: Individual evolutionary learning”. In: *Journal of Economic Behavior & Organization* 78.3 (2011), pp. 374–395. ISSN: 0167-2681.
- [4] J. Arifovic and J. Ledyard. “Learning to alternate”. In: *Experimental Economics* 21.3 (2018), pp. 692–721.
- [5] J. Arifovic and J. Ledyard. “Scaling Up Learning Models in Public Good Games”. In: *Journal of Public Economic Theory* 6.2 (2004), pp. 203–238.
- [6] J. E. Baker. “Adaptive Selection Methods for Genetic Algorithms”. In: *ICGA*. 1985.
- [7] Y. Cai, C. Daskalakis, and C. H. Papadimitriou. “Optimum Statistical Estimation with Strategic Data Sources”. In: *arXiv e-prints* (2014), arXiv:1408.2539.
- [8] C. Camerer and T. Hua Ho. “Experience-weighted Attraction Learning in Normal Form Games”. In: *Econometrica* 67.4 (1999), pp. 827–874.
- [9] K. Chernomaz. “Adaptive learning in an asymmetric auction: genetic algorithm approach”. In: *Journal of Economic Interaction and Coordination* 9.1 (2014), pp. 27–51. ISSN: 1860-7128.
- [10] J. C. Cox and S. C. Hayne. “Barking up the right tree: Are small groups rational agents?” In: *Experimental Economics* 9.3 (2006), pp. 209–222. ISSN: 1573-6938.
- [11] C. Deck and B. J. Wilson. “Auctions in near-continuous time”. In: *Experimental Economics* (2019). ISSN: 1573-6938.

- [12] C. Deck et al. “Common Value Auctions with Costly Signal Acquisition”. In: (2019 Manuscript in Preperation).
- [13] A. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer-Verlag Berlin Heidelberg, 2003.
- [14] I. Erev and A. E. Roth. “Predicting How People Play Games: Reinforcement Learning in Experimental Games with Unique, Mixed Strategy Equilibria”. In: *The American Economic Review* 88.4 (1998), pp. 848–881. ISSN: 00028282.
- [15] M. Feldman et al. “Simultaneous Auctions are (almost) Efficient”. In: *arXiv e-prints* (2012), arXiv:1209.4703.
- [16] R. French. “Computation and Machine Learning: Applications in Economics”. PhD thesis. Chapman University, 2017.
- [17] G. Guennebaud, B. Jacob, et al. *Eigen v3*. <http://eigen.tuxfamily.org>. 2010.
- [18] J. H. Holland. *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: University of Michigan Press, 1975.
- [19] T. P. Hubbard and H. J. Paarsch. “Chapter 2 - On the Numerical Solution of Equilibria in Auction Models with Asymmetries within the Private-Values Paradigm”. In: ed. by K. Schmedders and K. L. Judd. Vol. 3. *Handbook of Computational Economics*. Elsevier, 2014, pp. 37 –115.
- [20] K. A. D. Jong. “Genetic Algorithms Are NOT Function Optimizers”. In: vol. 2. *Foundations of Genetic Algorithms*. Elsevier, 1993, pp. 5 –17.
- [21] J. H. Kagel and D. Levin. “The Winner’s Curse and Public Information in Common Value Auctions”. In: *The American Economic Review* 76.5 (1986), pp. 894–920. ISSN: 00028282.
- [22] T. R. Kaplan and S. Zamir. “Asymmetric first-price auctions with uniform distributions: analytic solutions to the general case”. In: *Economic Theory* 50.2 (2012), pp. 269–302. ISSN: 1432-0479.
- [23] O. Kirchkamp, E. Poen, and J. P. Rei. “Outside options: Another reason to choose the first-price auction”. In: *European Economic Review* 53.2 (2009), pp. 153–169.
- [24] V. Krishna. *Auction Theory*. Academic Press, 2002.

- [25] J. O. Ledyard, D. Porter, and A. Rangel. “Experiments Testing Multiobject Allocation Mechanisms”. In: *Journal of Economics & Management Strategy* 6.3 (1997), pp. 639–675.
- [26] D. Levin, J. H. Kagel, and J.-F. Richard. “Revenue Effects and Information Processing in English Common Value Auctions”. In: *The American Economic Review* 86.3 (1996), pp. 442–460. ISSN: 00028282.
- [27] H. Li and J. G. Riley. “Auction choice”. In: *International Journal of Industrial Organization* 25.6 (2007), pp. 1269–1298.
- [28] P. Milgrom. *Putting Auction Theory to Work*. Cambridge University Press, 2004.
- [29] P. R. Milgrom and R. J. Weber. “A Theory of Auctions and Competitive Bidding”. In: *Econometrica* 50.5 (1982), pp. 1089–1122. ISSN: 00129682, 14680262.
- [30] R. B. Myerson. “Optimal Auction Design”. In: *Mathematics of Operations Research* 6.1 (1981), pp. 58–73. ISSN: 0364765X, 15265471.
- [31] G. Rudolph. “Convergence analysis of canonical genetic algorithms”. In: *IEEE Transactions on Neural Networks* 5.1 (1994), pp. 96–101. ISSN: 1045-9227.
- [32] T. Salimans et al. “Evolution Strategies as a Scalable Alternative to Reinforcement Learning”. In: *arXiv e-prints* (2017), arXiv:1703.03864.
- [33] W. Vickrey. “Counterspeculation, Auctions, and Competitive Sealed Tenders”. In: *The Journal of Finance* 16.1 (1961), pp. 8–37.
- [34] R. Wilson. “A Bidding Model of Perfect Competition”. In: *The Review of Economic Studies* 44.3 (1977), pp. 511–518. ISSN: 00346527, 1467937X.