Optimizing Complex Business Behavior With Evolutionary Computing

Andrew Hill
North Carolina State University
ahill6@ncsu.edu

ABSTRACT

At present, all analysis in many economic fields is limited to equations which permit closed-form solution and analysis. This results in a necessary simplification of all modeling assumptions and independent variables, both of which can reduce correlation with real-world behavior (for example, precluding non-differentiable models). By depending on automated methods to "exercise" the model rather than manual analysis, a significantly more complex model can be used - one which includes real-world behavior that is neglected in current techniques used in the field.

This paper presents work attempting to show the usefulness of model-based software engineering in non-traditional applications by developing a model of product development which also includes business decisions, signaling behaviors, and a market behaviors. This model is optimized using Particle Swarm Optimization, Differential Evolution, a Genetic Algorithm, and two sampling techniques. Results show gains of as much as 99% over baseline in some areas, and analysis is provided to describe the relative merits of the methods to one another, future extensions of this model, the potential of this type of approach to revolutionize the current state of the art in many fields, and some foundational efforts that need to be made for extension of these techniques to be fruitful.

Keywords

Model-based Automated Software Engineering, Software Engineering, Signaling Behaviors, Modeling

1. INTRODUCTION

At present, all analysis in many economic fields is limited to equations which permit closed-form solution and analysis. This results in a necessary simplification of all modeling assumptions and independent variables, both of which can reduce correlation with real-world behavior (for example, precluding non-differentiable models). By depending on automated methods to "exercise" the model rather than manual analysis, a significantly more complex model can be used - one which includes the real-world behavior that is neglected in current techniques used in the field.

One such example is signaling behavior during product development. Companies may choose to publicly announce that they are developing a new product if they believe such behavior will result in a competitor not entering the market (e.g. the competitor's development is behind and the announcement causes the competitor to believe it is in their interest to cut losses), thus reducing competition. Likewise, the company could refrain from announcing a timeline in order to hide the fact that their own product is behind schedule. Similarly, a company may signal in an effort to give customers information it believes will alter their purchasing behaviors (e.g. announcing a new product is coming out to induce loyal customers to wait rather than buy a competitor's product).

To complicate matters further, such signaling behavior takes place in an imperfect information situation, as competitors do not know the true state of a company's development and an announcement could be merely for show. (likewise demos and product pitches often overstate features). An example of economic analysis techniques in May 2016 includes only 2 companies and limited potential behaviors due to the need to manually analyze differentiable functions to describe optimal behavior.[2] In addition, product development and signaling behaviors do not take place in a vacuum, but as part of an integrated business plan that also includes elements such as raising or lowering price and attempting to schedule product release at an advantageous time (near the Christmas shopping season or when competitors do not yet have a directly competing product). These business decisions represent an important influence on technical behavior, but business models do not often include technical constraints, and to the author's knowledge, only one technical model attempts to include this type of external influence from business management. [1]

Finally, chance events can impact outcomes even after signaling (e.g. failure at the manufacturing plant, social event resulting in increased/decreased demand for their product, improbable breakthrough of competitor), thus adding a random perturbation to the economic situation in the case when a company's initial beliefs were accurate. These perturbative effects are not included in such models due to the difficulty or impossibility of maintaining requirements for

[1]The POM3 model of effort in Agile programming includes a factor meant to simulate the likelihood that a project will be terminated early

© 2016 ACM. ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

smooth, piecewise differentiable functions. This paper outlines the development of a model of economic processes including product development, business decisions, and signaling which includes 3 competitors (not including the company whose behavior is being optimized). While results are given for the 3-competitor case, the number of competitors is simply a parameter and can be set as desired.

The remainder of this paper is organized as follows: Section 2 describes the need for this type of model in Economics and how this work seeks to fill that niche. Section 3 describes the model itself from both an economic and Software Engineering standpoint. Section 4 gives background on the optimization algorithms which were used and justification for their selection. Section 5 reports the results of this work, both those of interest to Software Engineering and those for Economics. Section 6 describes potential threats to the validity of the model and its results; and Section 7 draws conclusions and describes needed future work.

2. BACKGROUND

It is assumed that the reader is not familiar with the business applications of Game Theory and the concept of signaling applied to business decisions. As such, the following background information is provided.

2.1 Problem

In Game Theoretical approaches to Economics, *signaling* and *credible commitment* are two of the most important concepts. Signaling is a voluntary disclosure of information by one party for the purpose of influencing the behavior of another party. The two most common practical applications of this behavior were mentioned above: Signaling by a company to influence the behavior of another company (whether competitor or not), and signaling by a company to influence the behavior of consumers. Obviously, there is also signaling by consumers to influence the behavior of companies (the theory of the free market is built on this concept), but that is subsumed in the discussion of our market simulation below.

The related concept of *credible commitment* (i.e the believability of a party's signals) is intimately connected with signaling, but for time constraints in developing this model it was assumed that all signals are taken as credible by all parties. Modifying the model to include the need for credible commitment and the possibility of misleading competitors would be a fascinating extension, but is left for future work.

2.2 Current Techniques

At present, economic work of such problems depends on mathematical analysis of equations which permit closed-form solution. This limits potential study to very simplified models, excludes non-differentiable functions, stochastic inputs, most chaotic or dynamic systems (systems with sensitive dependence on initial conditions), and recursive formulae which cannot be rewritten iteratively. As these are some of the most basic examples of the usefulness of computers in addressing analytically-difficult problems, it is surprising that the approach suggested in this paper has not been used to date.

2.3 Our Approach

Model-based Automated Software Engineering (MASE) seeks to formulate practical problems as searches over a space. In so doing, heuristic algorithms can be brought to bear to find good approximate solutions over practical timescales. Note the important differences with current methods. First, we do not attempt to solve the problem as formulated, meaning that we do not need the problem to be smooth, differentiable, or even deterministic. Secondly, by applying heuristics we acknowledge that solutions found by our techniques are not provably optimal (and indeed are almost certainly sub-optimal). However, by permitting approximations and heuristics we can address a far larger space of potential problems than is possible at present. This tradeoff of provable optimality for larger applicability is not appropriate for all situations. Still, it is believed that having a model which can generate testable claims and a means to optimize this model is a major step in a scientific approach to any field.

3. THE COMBINED MODEL

3.1 Overview

The goal of this model is to combine business and technical models in order to achieve a more holistic and accurate description of the business process.

The model operates in the following general way:

1. Generate all companies, their structural characteristics (POM3 decision variables such as Dynamism and Size). These represented the competitors of the company being optimized. The competitors were the same for each company, so that this is essentially a means of determining optimal behavior for a company within a given "world" of competitors.
2. For each timestep, the following actions occur:
   (a) If any company is signaling during this timestep, it does so
   (b) Any companies which have not already gone to market simulate one additional timestep in modified POM3. This results in additional features added to the product and new potential features being revealed.
   (c) All companies which have gone to market have their current products sent to the market simulator, discussed in the auxiliary section, which returns the market share captured by each company. Any company which has not yet gone to market is represented by a "base model" product whose desirability is significantly lower than standard products. This is to simulate the reality that last year's model is still available, and if a company's product is sufficiently poor, consumers may prefer to buy last year's model over a terrible "new" product.
   (d) Market share and price are used to calculate revenue.
   (e) Each company evaluates the expected outcome and potentially acts (i.e. if they expect to lose money, they can give up and leave the market).
3. At the end of all timesteps, one additional market simulation determines the demand for the final timestep. Note that revenue is the sum of revenue at each timestep,

and that more consumers are seeking to buy a product toward the end of the simulation (this is to simulate the reality that while some people need to buy a new product (e.g. cell phone) right now regardless of what is available, the majority will wait for new products to be released.

In this setup, each prospective solution (set of company actions in a given market) is tested against the same group of competitors by running a product development and marketing simulation for all companies. This is equivalent to the case where each company is running a simulation of all other companies (i.e. each company is constantly evaluating their advantage/disadvantage relative to their beliefs about their competitors). Equivalence is because the project assumes symmetric information (i.e. all companies know the same information about their competitors).

One simple (if computationally expensive) extension of this work would be to include imperfect information in company beliefs, for which signaling could have mixed effects (by releasing your timetable you intimidate a company that doesn't know anything about you, but let a competitor with better industrial intelligence know that they can beat you).[2] At present however, this model uses symmetric information.

3.2 Decision Variables

The basic parts of the POM3 model used for this model were essentially unchanged, and Table 1 gives all decision variables and their ranges. The generation of a feature tree with random constraints, partial knowledge of said tree, and the basic structure of the simulation as broken into steps of completing what work can be completed according to a given (but changeable) strategy is largely identical. As in POM3, groups are made of workers with different skill levels, but no use is made of that fact in this project. For objectives, the calculation of the Cost, Value, and Score are unchanged.

Some items from POM3 were removed or altered in order to facilitate integration into this new framework. For example, the majority of the randomness within a single model run was removed in order to standardize results between runs. In order to include signaling behavior it was necessary to add the ability to alter any decision variable in the middle of the simulation, to create the ability to reset the entire model to a saved baseline configuration before each simulation (so that each company is working within the same "reality" of what features depend on which and relative costs), and to be able to step through the simulation if needed rather than run the entire process from beginning to end. This last was necessary because business decisions and signaling behavior by their very nature interrupt ongoing processes and change assumptions, strategies, or other quantities modeled by decision variables.

Note that the POM3 decision variables allow description of a broad range of companies, from large traditional corporations to small, nimble startups. The model's flexibility was very beneficial in modeling a variety of company types as competitors. A more detailed description of the business variables is below.

Table 1: Decision Variables

Variables	**Range**
Culture	0.1-0.9
Criticality	0.82-1.20
Criticality Modifier	2-10
Initial Known	0.4-0.7
Inter-Dependency	1-100
Dynamism	1-50
Size	0-4
Development Plan	1-5
Signals	0-63
Time-to-Market	0-10
Price	0-10,0000

Table 2: Auxiliary Variables

Consumer loyalty
Previous market conditions
Cost/Value
Price
Score

3.2.1 *Signals*

Signals were represented as a number from 0 to 63. Written in binary, this number determined whether a company would signal at each timestep (5 time slots, can signal in as many as desired). Time was divided into 10 timeslots to facilitate other calculations, and each signaling event was randomly assigned to either the first or second half of the period.

3.2.2 *Time to Market*

As mentioned above, development was divided into 10 timeslots, with the eleventh possible value meaning the product was not taken to market until the entire simulation was complete. When a product was marketed, it's development was halted for all remaining timesteps, and it was added to a list of current products (replacing the base model for that company).

3.2.3 *Price*

Price is self-explanatory, but does merit a quick note. When price was initially added as a decision variable, I had not yet created the market simulation to determine demand. Because there was no penalty for an outrageous price, all successful companies immediately had product prices near the maximum allowed value. This behavior matches general belief that price influences on demand are a major factor in keeping prices reasonable, and also gave much needed troubleshooting guidance.

3.3 Model Auxiliary Variables

The secondary modeling of market forces was necessary to determine the success of each product. This was accomplished by a probability distribution across the available products (including base models) which was changed by Bayesian updating to account for both new products and the influence of customer loyalty. This loyalty started out uniform among the companies (as did the prior distribution for product sales). A fascinating detail about this sub-model is that the variety of inputs to the market - given in Ta-

[2]In fact, the first two versions of the model included the imperfect information system, but because it was not going to be used the simpler and more time-efficient current model was developed

Table 3: Objective Variables

Variables	Range
Cost	0-10,000
Value	0-10,000
Score	0-1
Market Share	0-1
Revenue	0-1,000,000

ble 2. Inputs included decision variables (Price), objective variables (Cost, Value, Score), and auxiliary variables (Consumer loyalty, previous market conditions). This greatly added to the complexity of the results.

3.4 Objective Variables

Table 3 gives all objective values and their ranges. The Cost, Value, and Score objectives are taken more or less completely from the POM3 model. Cost represents the total cost of developing the features, Value is the value accrued by including those features (both were originally developed to decide the next task to accomplish in an agile framework). The cost and value of each requirement is randomly generated on model instantiation, is encoded in the requirements tree, and is stored so that each run has the same costs and values for each potential feature. Score is a measure of the value accomplished compared to what could have been accomplished with perfect information. In the context of this model, it is used as a proxy for internal efficiency.

Market share is a result of the market simulator discussed above, and is a percentage of the market captured by the given product. As mentioned above, this depends on a variety of inputs, decision variables, objective variables, and auxiliary variables.

Once market share has been calculated, revenue is simply the product of the unit price and the number of units purchased (calculated by the percentage of market share multiplied by the number of consumers purchasing a product this timestep). Again, the number of consumers purchasing each timestep increases toward the end of the simulation in order to simulate the fact that while some consumers must by a replacement product now, many are willing to wait until all products are available. This also prevents the early-to-market strategy from dominating all other possible strategies - which is unrealistic.

4. OPTIMIZERS

All Genetic Algorithms (GAs) follow the same basic processes: generation, crossover, mutation, selection, repetition. First, an initial population is generated, either at random if the problem is not heavily constrained, or else by some means such as a Boolean or Propositional Satisfiability Solver (SAT solver) to generate valid candidates. These candidates are in some way combined (crossover) after the analogy of biological reproduction, combining pieces of information from two or more "parents". These newly-created "child" solutions are then randomly altered (mutated) at some probability in order to introduce more variety into the gene pool. The selection step chooses only the fittest of these candidates to survive; and this process is repeated until suitable solutions are found. While these basic steps are common to all the GAs used (PSO, DE, and GA) and it is important to bear the structural similarities in mind, the details of their implementation are often quite different. Each method description includes runtime considerations which will be summarized again in the results.

4.1 Particle Swarm

Particle Swarm Optimization (PSO) is inspired by the ability of flocks of bird or schools of fish to exhibit complex macro-behavior as a result of very simple individual rules. The goal of a PSO is to mimic many small animals spreading out to explore the search space, but always being drawn to the best values found so far. Particle Swarms are distinct from other optimizers used because each potential solution (particle) in the current generation (swarm) does not simply have decision values, but also velocity. Thus particles are not undergoing explicit mutation or crossover so much as literally flying around, being influenced by the swarm's global best value and their own inertia. Velocities were initialized to small values (random values of no more than 0.001 of the range of the variable for each decision variable). Initial particle swarms flew off the edge of the search space almost immediately despite the constriction factor, so for the final version any particle which flew off the edge of the search would both be placed at the boundary and have its velocity set to 0. Thus whenever a particle experienced this, it would be motionless and only experience a pull toward the current global maximum.

Particle swarms generally must have their parameters carefully balanced. If the constriction factor is too high, all particles will immediately fly together to the first global maximum without exploring the space. If it is too high, they can zip all over the search space and spend most of their time at the boundary having their velocity reset to zero. The PSO used for this project follows [1] in not using a local maximum, and so the only influences on particles are their velocity from the previous timestep and the global maximum. Also following that paper, $\phi_1 = 2.8$, $\phi_2 = 1.3$, and $k = \frac{2}{\left|2 - \phi - \sqrt{\phi * \phi} - 4\phi\right|}$, where $\phi = \phi_1 + \phi_2$.

PSO can be implemented in three matrices plus an update method which involves simple arithmetic, so it is not surprising that PSO had the shortest runtime of any method used. The PSO required 30,000 fitness function evaluations and called elitism once with size 30

4.2 Differential Evolution

Differential Evolution (DE) is an interesting method originally published in 1997.[5] The simplicity of the method and its surprising ability to continue to show results in the current climate of increasingly complex GAs makes it one of the interesting techniques available.

DE works by maintaining a Pareto frontier, from which candidates are selected to create the next "generation". A new candidate solution is created by taking a randomly selected point from the frontier, and altering it by the "average change" so to speak of two other frontier points. Because all points on the frontier are better than everything seen so far, this results in a method which uses the data encoded in the current bests to constructively guide mutation. DE is one of the only basic methods which are still effective in many instances, and it is often used as a tuner for the "magic parameters" of more complex techniques.

The DE ran 100 generations with a frontier size of 110 (10*number of decisions) for $f = 0.5$, $cf = 0.3$, and $\epsilon =$

0.01. This method did 11,000 fitness function evaluations and called elitism one time with size 110.

4.3 Generic Genetic Algorithm

The Genetic Algorithm used for this project was taken from the one coded for Workshop 4, with the addition of Continuous Domination for the fitness function. This is a very standard genetic algorithm, and for each generation creates a pool of child solutions via crossover and mutation. The size of the child pool is equal to that of the population, and the solutions are then cut in half by the selection process, which takes the fittest half. It had a population of 100 and ran 100 generations with single-point crossover at the midpoint of the decision variables between two population members selected uniformly at random, and had a mutation rate of 0.01.

The GA does 10,000 fitness function evaluations, and calls elitism 101 times with size 200.

4.4 Sampling

Random sampling was used both as a baseline against which to compare the optimizers, and also because there is increasing data in many fields which use Genetic Algorithms suggesting that in many cases sampling can be as effective and much faster.[4, 3]

The sampling algorithm took 500 randomly-generated points, and returned the 15 best points. It required 500 fitness function evaluations, and called elitism once with size 500.

4.5 Adaptive Sampling

After early success of sampling, and in light of recent scholarship which calls into question the usefulness of mutation and crossover in exploring the search space, the basic sampling method was added to provide a reference point. That sampling method was experiencing significantly longer runtimes than DE and PSO, with indications that the elitism step on a population five times larger was causing significant slowdown. It seemed natural to try a sampling method which used an initial smaller sample to find areas which merited more attention. The technique needs some further work, and future adaptations could include a larger initial sample (at present it generates only as many points as the GA creates for the initial population), and perhaps putting a hill climber on each point to find the local maximum rather than simply taking a few local samples and selecting the best.

Adaptive sampling samples a single GA generation (here 100 points), then conducts small samples of the local areas for the best 10% from the initial sample. Thus the entire method samples only 150 points (1.5 GA generations). This method does 150 fitness function evaluations, and runs elitism once for size 100 and ten times for size 5).

4.6 Continuous Domination As A Fitness Function

The fitness function for all these techniques was continuous domination (cdom). A very popular alternative, Binary domination (bdom) is often used, even in state-of-the-art tools such as NSGA-II. Binary domination counts point A as dominating point B only if every member of point A is at least as good as the corresponding member of point B *and* at least one member of point A is strictly better than its counterpart in point B. Unfortunately, as the dimension of the space under consideration increases, it becomes prohibitively difficult to dominate many other points under these strict rules. As a result, methods which use bdom often have a large number of points which do not dominate one another (thus all technically optimal), and reducing this gaggle of solutions to a reasonable size requires some type of secondary sort. Worse still, bdom says only that A dominates B, giving no indication of how much. For bdom, a grasshopper is bigger than an ant, and so is an elephant; there is no difference between the grasshopper and the elephant.

Continuous domination raises differences between objective values to an exponential, drastically exaggerating the difference. Exponentiation resolves the issue in two ways. First, it provides a measure of *how much better* A is than B, rather than just saying it is better. Second, by turning a binary Yes/No into a real number, it greatly reduces the likelihood of ties. While bdom may generate a horde of mutually non-dominating points, cdom is virtually guaranteed to show some points dominating others and maintain a reasonably-sized set of ”best” points. However, it is important to note that because cdom does not require a secondary sort, it can lose the benefits bdom methods have been forced to accrue. GAs which use bdom must have a secondary sort because its primary sort (bdom) is not powerful enough. Yet they can select a secondary sort in such a way as to gain desirable behavior: automatically improving spread by using cuboid distance for example. Also, because cdom raises differences to an exponent, large differences in a single plane can dominate difference calculations. Two solutions which are identical except for a moderate difference in one variable can be ranked much farther apart than solutions which are slightly (but significantly) different in every way. Finally, cdom in this project showed a distinct tendency to be dominated by one or a few very strong solutions. It may be that the method needs some help of the type cuboid distance gives to bdom to overcome this tendency.

4.7 Neighborhood Equivalence in Fitness Functions

One practical consideration is that there are often differences without meaning. For example, two prospective jerseys for a sports team are different colors, but by such a minute amount that the human eye cannot tell the difference. An automated technique can penalize the ”not quite red” jersey for not meeting the specification, even though it is completely acceptable practically. In order to remedy this lack of automated common sense, the idea of neighborhood equivalence can be used. This determines an epsilon beyond which two different values are considered the same. Restaurants make common use of this, in that the amount of every ingredient in each burger or batch of sauce is not necessarily identical down to the microliter, and if someone eats a Quarter Pounder from McDonalds that differs from 0.25 lb in the fifth decimal, they are unlikely to complain.

In this project, epsilon neighborhoods were set at 0.01% of the range. Anything varying by less than one hundredth of one percent was considered essentially identical. The exception to this is Adaptive Sampling, which depends on finding distinct interesting regions. Because the concept of a few of the most interesting regions found only works if those regions are in fact different, for that method epsilon was set to 0.1%.

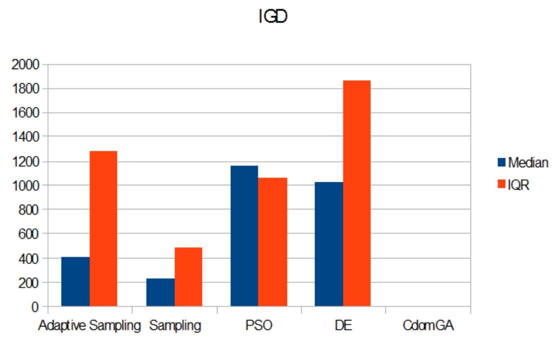

Figure 1: IGD (smaller is better)

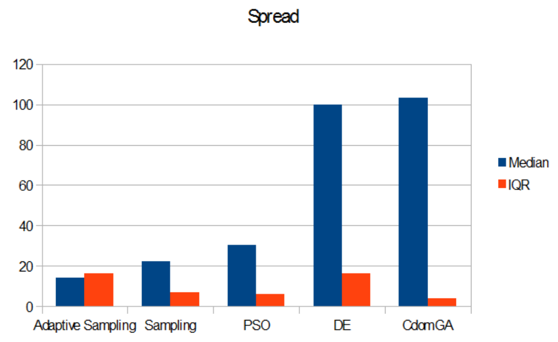

Figure 2: Spread (smaller is better)

5. RESULTS

5.1 Comparison of Optimizers

5.1.1 IGD

Inter-Generational Distance(IGD) is a measure of the average distance of solutions in a given method's Pareto front to the true Pareto front. That is, "How far are my solutions from perfect?" Since direct calculation would require knowledge of global optima, IGD is commonly calculated by creating a reference set from the final populations of the methods under review (being careful not to bias the composition because of different final population sizes). In that sense, IGD is a measure of how close your solutions are to the best solutions found by any method under study, so it is still a valid means of comparing different approaches.

Figure 1 shows the IGD values for the methods used. This is a slight update of the chart showed in the presentation of this work, as I have since modified Adaptive Sampling to improve both IGD and Spread. All except PSO and DE are in separate Scott-Knott groups, but there are still major groupings. The first group is the high-performer: the GA. However, it is clear from the spread that this is due to the GA being focused on a few dominating solutions, as spread is terrible.

The second group of interest is the sampling methods. They work quite well compared to PSO and DE, but are certainly not doing as well as the GA. Note also that for the sampling methods, median behavior is strong, but there is wide variance depending on whether they randomly sampled good solutions. Again, we will see in spread that they are finding larger portion of the Pareto front than the GA, and in some situations it may be desirable to trade some type of optimality for a more complete picture of the Pareto front.

Finally, we have PSO and De. Both these methods are useful in certain situations, and while PSO has decent spread (suggesting it may also be finding a broader section of the Pareto front, DE has poor IGD and poor spread. This is likely due to poor tuning, as it is known that finding the right values for the "magic parameters" in DE has a considerable impact on the success of the method.

5.1.2 Spread

Spread is a measure of spread-out-ness. Definition 1 contains the formula, in which the last term in the numerator is most noteworthy.

Definition 1. Given N obtained solutions d_i on the Pareto frontier, with average distance $\overline{d}$ between them, d_f and d_l being the two extreme points,

$$spread = \frac{d_f + d_l + \sum_{i=1}^{N-1} |d_i - \overline{d}|}{d_f + d_l + (N-1)\overline{d}}$$

Notice that when members of the Pareto front are evenly spaced, this term becomes zero. Spread is worthwhile because Automated Software Engineering is no longer simply about finding a single optimum. Business users and technical design groups alike are interested in a deeper understanding of the tradeoffs of models, and having a good spread means that a method is finding a variety of solutions rather than finding one good solution repeatedly. In addition, Spread, IGD, and Hypervolume all have weaknesses as metrics, so it is best to use at least two of these measures in conjunction in order to have a clear picture of the effectiveness of different optimizers on a space.

Figure 2 shows the spread values for the different techniques, and comparing to Figure 1 for IGD we begin to have a clearer picture of optimizer behavior. DE does indeed appear to be either broken, poorly tuned, or unsuitable for this space. It is in the worst Scott-Knott group for both Spread and IGD. Furthermore, in both cases the IQR is so large that it may be the absolute worst method in both cases. The GA had the best IGD by a good margin, but has the worst spread. As previously mentioned, this suggests that the GA is being dominated by very strong solutions, and having trouble breaking away from those few points to look for other solutions. For a GA with bdom this would like have been solved (almost accidentally) by using cuboid distance as a secondary sort.

PSO and Sampling are in the same Scott-Knott group, which taken together with the IGD data and upcoming information about optimizations suggests that PSO in this case is simply a slightly inefficient sampling technique. It is not surprising that either PSO or Sampling have good spread values, since the PSO algorithm requires particles to fly everywhere, and random sampling by definition attempts to cover a large area. Adaptive Sampling is in a league of its own for Spread, which is to be expected since the algorithm intentionally isolates the n best areas, then picks the best point from a secondary sample of the local area.

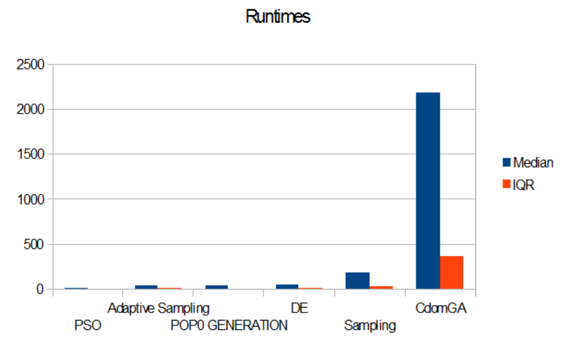

Figure 3: Runtimes (smaller is better)

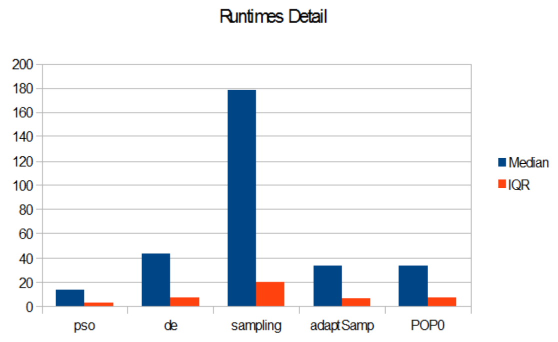

Figure 4: Detail View of Previous Chart

5.1.3 Runtime

Runtimes are not a measure of the correctness of a method so much as of its practicality. While most leading methods in the field (e.g. NSGA-II, SPEA2) are GAs, there is a widespread acknowledgement that execution times are extremely high. This necessitates either finding methods which are faster and almost as good, or else heavy investment in High Performance Computing (HPC) architecture. Since investing in expensive computers is a solution which does not scale well, finding fast techniques with appropriate gain is preferable. In addition, it is always possible to use the fast techniques to seed the first generation of GAs, jump-starting their processes by allowing them to begin with a high quality population rather than a randomly selected one.

Runtimes are as expected. PSO is implemented in three lists and a simple update method, so it is not surprising that it is the fastest. Likewise, since Adaptive Sampling is doing only 150 fitness function evaluations, it is not surprising that it comes in next. Note that POP0 in this chart is the time required to generate the baseline set (i.e. generate 100 points, evaluate them, and return). POP0 is in the same Scott-Knott group as Adaptive Sampling, meaning that there is not a significant difference in the runtime between a method which had the best spread and outperformed PSO and DE in IGD and simply generating initial points for starting the GA. Adaptive Sampling was also substantially faster than Sampling (c.f. Figure 3). It is likely that this difference is due to the cost of performing a sort on 500 items while the majority of Adaptive Sampling sorts were on five items.

It is not clear from Figure 3, but Figure 4 shows the expanded runtime chart for all non-GA methods. In that it is clear that DE is slower than the methods described previously. While it is in the bottom half of methods on this chart, the reader should note that DE is a fast method compared to anything in common use. Finally, the detail view makes clear that despite being very fast, Sampling is 4-10 times slower than the other methods. Since it is only evaluating 2.5 times more fitness functions than Adaptive Sampling, and 10,500 fewer than DE, it is incredibly unlikely that fitness function evaluations are the primary factor. Even fast sorts are $n \cdot \log(n)$, and when compared to these other methods, that can be an eternity.

Table 4 reports the number of fitness function and elitism evaluations for each technique (elitism was initially discounted, but found to dominate runtime costs if called for a large enough population). Recall that each call to the fitness function requires running the POM3 variant and market simulation. Also, because the fitness function only calls the model if objectives have not already been calculated, it is unlikely that fitness function calls in the elitism method account for its runtime effects.

5.2 Economic Results

The first question for economists is whether the optimizers did in fact improve company behavior. The IGD, Spread, and runtime data is interesting, but unless they recommended behaviors that improved outcomes, the software engineers will want to know how the optimizers are broken and the economists and businessmen will quickly grow bored. Table 5 summarizes the improvements of each method for each of the objective values over baseline. This baseline was 100 randomly-generated decisions (i.e. equivalent to evaluating generation 0 for the Genetic Algorithm). All values are in percent improvement. For example, 99 in Cost for GA means that the GA using cdom improved costs (i.e. cut costs) by 99%. Likewise 150 in revenue for Sampling means that technique increased revenue by 150%.

This table is based on the most recent run, after modifying the model to replace a reference to the POM3 objective Idle with Market Share. First, the consistency of these values with previous results suggests that the previous values were not anomalies, but actual program behavior. A company would love to cut costs by 99% while increasing the efficiency of their techniques by over 660% and almost doubling revenue. However, the shocking results in Market Share across the board suggest either that the implementation is buggy (my initial thought), or else that the decision variables associated with these cost reductions are incredibly unpopular. Because of the novelty of these methods and the fact that the implementation of market forces is barely one week old, the most prudent conclusion would be that there is an error in the implementation of market forces.

That error notwithstanding, it is clear from the remainder of the data that business users have two general choices with these optimizers: a sampling method for decisions which need to be made on very rapid timetables, and a GA if the time and resources are available to wait for an optimal solution.

Additionally, it is interesting to remember that the results of any model which are of interest to a specialist in that field are not the objective values, or even the improvements over baseline. The decision variables associated with those

Table 4: Runtime Components

Method	Fitness Function Evals	Elitism Calls (Size)	Runtime (Sec)
PSO	30,000	1 (30)	13
DE	11,000	1 (110)	43
GA	10,000	101 (200)	2186
Sampling	500	1 (500)	177
Adaptive Sampling	150	1 (100), 10 (5)	33

Table 5: Percent Improvement

	PSO	DE	GA (cdom)	Sampling	Adaptive Sampling
Cost	45	8	99	75	68
Value	6	5	11	17	7
Score	15	11	669	317	203
Revenue	5	18	186	150	122
Market Share	17	41	99	99	99

improvements are the true treasure, since they suggest what actions should be taken to achieve the gains. There were many interesting economic results throughout the course of this project, of which a few are offered below.

5.2.1 Signaling Behaviors

One of the most striking (and in retrospect, obvious) tendencies of companies found in the first review of results was that successful companies tended to signal either constantly (during all possible time intervals) or else during the first half of the development period only. This makes sense from a business perspective. If Apple is trying to convince customers to wait for the iPhone7 rather than buy a Google Pixel now, they must make some kind of announcement early enough that customers have not already purchased a Pixel. Likewise, if the signaling is to dissuade potential competitors from entering the market, the signaling must occur before the potential competitor has sunk enough money to feel committed.

This concept is already widely understood in Economics, but the model's ability to generate valid economic principles without external guidance lends weight to the case for its usefulness.

5.2.2 Demand Influence on Price

Supply and Demand are the foundation of modern market economics, and the idea that increasing price will lower demand is one of the earliest insights of all. The behavior of the model prior to relating demand to price was for all companies to charge very high prices independent of all other variables. Once demand was related to price, this behavior disappeared. Again, this behavior is not a new or unexpected discovery; but rather speaks to the validity of the basic model setup in its ability to reproduce known economic truths.

5.2.3 Naturally Occurring Market Segmentation

Markets are often divided amongst companies which supply cheap, low-end goods, moderately-priced goods of reasonable quality, and high-end luxury goods (possible with additional gradations). Our model produced this result in the Pareto front, yielding Pareto optimal behavior corresponding to low-cost, low-quality goods as well as companies which sold very few goods, but sold them at an extremely high price. This seems to be emergent behavior from the model itself.

6. THREATS TO VALIDITY AND FUTURE WORK

6.1 Software Engineering Threats

The first and most serious threat to validity is that none of the optimizers being compared are state of the art. To make firm claims about the relative value of sampling and GAs for IGD, Spread, and runtime would require running this model with NSGA-II or another state of the art technique(e.g SPEA2, IBEA2, MOEA/D). Without this data, all conclusions presented here must be taken with a hefty dose of salt.

The unexpected behavior in the Market Share objective requires either a satisfactory explanation or discovery of the implementation error. It is certainly possible that the way the optimized solutions are cutting costs by so drastically is causing a loss of market share, but many of objective values were on the order of 10^{-16} or smaller, which suggests that these poor showing on market share are not accidental. Another potential check would be to vary the number of competitors. If the market share calculation is working correctly, adding additional competitors should (to a point) cause a statistically significant decrease in market share.

Another potential weakness is the small number of settings used. The model was run with three companies only, and only 90 times over three full runs. With this limited data, it is difficult to generalize any conclusions or make claims for wider application of the method. Also, the IQR values for the percentage improvements were quite high, in some cases high enough to go from significantly improving company outcomes to significantly harming the company. The quartile values and not simply spreads will need to be consulted, to determine if this is a threat to validity, which will require altering the implementation of the statistics method somewhat.

Finally, analysis of results for interpretation required manual inspection of decision variables, a task which introduces an element of subjectivity and creates the potential for bias in the result. If this technique or an analogue is to be used, some standards for data analysis of optimal decision variables is required. Due to the nature of the work, it is unlikely that expert judgment can be entirely removed; but standards in interpretation should be developed, perhaps including some type of binning whose parameters are set prior to the experiment (following the example of epsilon equivalence in fitness functions).

6.2 Economic Threats

All assumptions used in the construction of this model were validated through Dr. William Caylor at SMU, as well as periodic checks of results to verify that the behavior suggested was economically sensible. However, that does not include the implementation of Market Share (which could also explain its unexpected behavior). Furthermore, these methods are sufficiently novel and the results sufficiently startling that widespread economic acceptance will likely not occur without the ability to predict appropriate behaviors for a problem which has not been solved, with verification of the correctness of the prediction serving as empirical support for the method's validity. At present Dr. Caylor and myself have been unable to find a suitable problem.

Until such a problem is found, continued analysis of the decision variables which the model claims offer such drastic improvements should yield a rich discussion on the economic ramifications of those choices. It is possible to gain some limited acceptance for these methods if enough of those suggestions line up with accepted economic practice and can be demonstrated to derive from the model itself and not from the interpretation of the researcher.

6.3 Future Work

6.3.1 Model Changes Needed

This work is more closely related to next release planning than product development, because the next version should treat the development as a decision of what features to put in the next release, together with when to release it and how to signal (and perhaps other things). That approach would more accurately reflect economic and business reality, as well as more accurately describe the relationship between the business and development halves of this model. A decision about how to balance the area of overlap between those processes in this method needs to be further developed.

6.3.2 Approach Changes Needed

The current model is custom-made for this application. While it is easy to see how the concept could be expanded to other fields (both within Economics and outside it), the code would need to be rewritten with better separation of concerns and modularization order to facilitate model interchangeability.

6.3.3 Application Extension

There are many signaling behaviors and business decisions that were not included in this model. Signaling with different quantities of information, counter-signals, misleading information (this model assumes all signals are honest representations of reality), asymmetric information for companies. This is solely in other signaling behaviors and only a sample. There are innumerable other economic and business decisions and processes that could be integrated. A fully-conceived framework for the integration of smaller models for each portion of the world will have to await a different paper, but there are certainly many avenues to extend this work.

Further extensions will, however, increase runtimes - potentially to the point of making the method non-viable for business users and of merely academic interest. Since the fastest techniques are taking 10 seconds per run, this is not the simplest model. Still, I would propose a standard runtime of one workday on a standard computer as a reasonable upper bound for what business users could accept. There are many conceptual extensions that could be made while staying within those bounds, particularly when considering that this code was not at all optimized for speed and was implemented in Python. Simply porting it to another language could likely give modest runtime improvements.

There is nothing about this technique which limits its applicability to Economics. Systems which require optimization in other fields and do not permit closed-form analysis could easily be introduced, as could all problems involving chaotic or dynamic systems (systems which are sensitive to initial conditions). With a slight reformulation, this approach could be broadly applicable.

If this field is to become a new means of investigating the universe, a foundational comparative theory needs to be developed for how to apply this technique to a give problem, which this approach is appropriate, and limits to the strength of the conclusions which can be drawn. The latter of these are largely engineering exercises in applying known statistical principles, but the former represent questions about the fundamental rules of any field and evoke problems in theory of knowledge. Additionally, the value of both empiricism and thought experiments has been appreciated since ancient times, but this would combine the two in the creation of knowledge based on empirical thought experiments. There are many and troubling potential consequences of living in a world constructed on the average of our imaginings, and a much thought should go into the foundational premises of such work before it is applied widely.

7. CONCLUSION

This technique successfully found solutions which reduced costs by 99% while increasing revenue by 186%, and improving process efficiency by over 600%. Because these were not correlated in the model (e.g. process efficiency did not directly influence cost, and revenue was purely income), it is likely that if these results translate to the real-world the true savings would be much greater.

From a Software Engineering perspective, the method found a problem for which PSO is very poorly suited, reaffirmed the importance of tuning for DE performance, discovered a weakness in cdom that it allows strong solutions to dominate by removing the need for a secondary sort, and added further fuel to the forest fire of results showing that sampling methods are often competitive with genetic techniques.

Overall, this method is definitely useful in reproducing known economic facts and shows scalability to tackle much more complex problems than current techniques in the field permit. Prior to widespread adoption or the expansion of this method to other fields, more validation is required for both this particular model and the approach as a whole. If successful, this approach could greatly expand the scope of problems about which science can make meaningful statements, but it must be stressed that this would be in a stochastic rather than deterministic way. Thorough training in Statistics is already to some extent prerequisite for many fields, and this approach only deepens that trend.

8. PROTIPS

Things learned from running this experiment that are good to know when starting:

8.1 When To Run Metrics

When initially considering the results, IGD and Spread were run on all results from all 30 trials as a group (thus IGD was the distance for each technique in each trial from the best solutions found over all 30 trials). This resulted in shockingly large numbers and drastically different results than if IGD was run after each trial (so that you are in essence only testing to see which optimizer was most effective on this particular trial). The former essentially turns all 30 runs into a single giant experiment, while the latter is equivalent to running the experiment 30 times (and thus more desirable statistically. The latter also makes more sense as a measure of which optimizer was the most effective, since it measures how each optimizer did per trial.

Spread did not meaningfully change from one way of evaluating it to the other (two methods changed by a single point in IQR and one changed two points in median, no Scott-Knott groups were affected). This makes sense considering that spread is an inherently "local" measure (in the sense of only considering the population under study and not the results from other optimizers)

8.2 Needed Tools

If these type of methods are to begin extensive use in other fields (whose experts are interested in decision variables, not objectives, IGD, etc.) then some automated analysis of decision variables is needed. As alluded to above, analyzing decision variable behavior is needed to draw any economic conclusions and to confirm that the model is valid, but at present must be done manually - which is quite tedious.

9. ACKNOWLEDGMENTS

Many thanks to Dr. William Caylor in the Economics Department at SMU for his input on the economic portions of this project.

10. REFERENCES

[1] A. Carlisle and G. Dozier. An off-the-shelf pso. In *Proceedings of the Workshop on Particle Swarm Optimization*, 2001.

[2] W. Caylor. On competitive pricing and strategic announcement of future products. 2016.

[3] Y. Qi, X. Mao, Y. Lei, Z. Dai, and C. Wang. The strength of random search on automated program repair. In *Proceedings of the 36th International Conference on Software Engineering*, pages 254–265. ACM, 2014.

[4] Z. Qi, F. Long, S. Achour, and M. Rinard. An analysis of patch plausibility and correctness for generate-and-validate patch generation systems. In *Proceedings of the 2015 International Symposium on Software Testing and Analysis*, pages 24–36. ACM, 2015.

[5] R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.