Flea Market Simulator: A Market Simulator for Experiments on the Computational Evolution of Trading Strategies for Economic Agents

Omri Bernstein
Cognitive Science
Hampshire College
Amherst, MA 01002
ob08@hampshire.edu

ABSTRACT
In this paper, we present the Flea Market Simulator: a minimal market simulator, where agents have strategies they use to determine which goods to trade and at what rate. The primary goal is to understand how intelligent strategies can develop in a dynamic economic environment. The simulator is designed to run multiple independent strategy generating functions at once, for isolated sub-populations of agents in the market. The timing and extent of interaction between these sub-populations can be flexibly controlled by the user. Resultant differences between sub-populations provides data with which to compare the different strategy generating functions. Specifically, these strategy generating functions are intended to be evolutionary computational (EC) methods, and as such, the simulator provides the framework for tournaments between EC methods, in terms of their ability to produce "successful" agent trading strategies. The definition of success is left purposefully open-ended.

Categories and Subject Descriptors
I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence —*Intelligent Agents*; I.6.8 [**Simulation and Modeling**]: Types of Simulation—*Distributed*; J.4 [**Computer Applications**]: Social and Behavioral Sciences—*Economics*

Keywords
Economics, experimentation, theory, market simulation

1. INTRODUCTION
1.1 Background
Agent-based economic simulation has a very deep history. In terms of evolutionary computation (EC), the earliest experiments were performed at the Santa Fe Institute in the late 80s and early 90s. Arifovic (1989) used genetic algorithms, in a variety of models, to evolve strings representing agent strategies [2]. Based on John Holland's schema, each string was essentially a table for matching predefined market conditions with decisions. More broadly, the Santa Fe Artificial Stock market was also based upon "condition-matching" (again, based on Holland's schema) [5].

GECCO'12 Companion, July 7–11, 2012, Philadelphia, Pennsylvania, USA.

Andrews & Prager (1992) used genetic programming to evolve agents in a double auction market simulator [1]. In a double auction, agents "call out" both buying and selling offers, then trades occur when there is a match. Andrews & Prager showed that genetic programming could beat many hand-coded strategies. Chen (2000) uses a very similar double auction market with genetic programming [4].

The above simulations have various disadvantages. One frequent limitation is the constraining of agent strategies to a series of weights for various market status indicators. In other words, agent strategies in previous market simulators were really just specific combinations of preset conditions and rules. On the other hand, the Flea Market Simulator is very open-ended in terms of what an agent strategy can be: simply, an expression that evaluates to a number. Arthur (1992) argues that deductive reasoning is an incomplete explanation for rational behavior in economic agents [3]. He claims that inductive reasoning must operate in agent strategies in heterogeneous conditions—which are the conditions of all real-world markets.

Another limitation has been something we might call "need mandating". For example, in Andrew & Prager (1992), agents were rigidly divided into "buyers" or "sellers" for any given round. The fitness function for the genetic programs would compare agent's performance to some mandated ideal price (specific to each agent). Although this ideal price was determined intelligently, it could limit the performance of agents by what is currently understood. In the Flea Market Simulator, as shall be shown, agents develop their own conceptions of "need".

1.2 Motivations
The primary motivation for this market simulator is to understand how to use EC to produce successful agent trading strategies in a dynamic, heterogeneous market. Even more fundamentally, the goal is to understand the limitations of current EC methods on dynamic problems such as this one. The hope is to tease out what may be needed to resolve these limitations.

The secondary motivation for this market simulator is to understand the emergence of different human social and economic dynamics under varying conditions. In this case, a given simulation would have only one active genetic programming paradigm. Simulation parameters (e.g. market size) would be varied, and the object of scrutiny would be emergent properties of the market that develops.

These motivations, although conceptually independent, have the potential for synergy. After all, different market parameters should catalyze or cultivate successful strategies to different degrees—where effective parameters may be those that most closely produce human-like emergent market properties. Reciprocally, robust agent strategies should be a significant factor in accurately simulating human social and economic dynamics.

2. FLEA MARKET SIMULATOR

2.1 Overview

The Flea Market Simulator is a trading market simulator written in Clojure[1]. The simulation proceeds in discrete time steps, or rounds. Each round, agents (called fleas) trade in pairs, then consume essential goods, and potentially "die" (are removed) if they hold zero of any essential good. Then the market is repopulated with fleas (to a constant size) and goods repossessed from the dead fleas are distributed equitably among the "living". The two fleas in any trading pair mutually determine (haggle) the goods and rate of trade. The details of trading mechanics will be explained later, but for now, think of the simulator determining the terms of trade (which goods and at what rate) by calculating the "maximally mutually beneficial trade" for each flea trading pair. Refer to Table 1 for definitions of terms.

The simulator is purposed for use in tandem with one or many EC methods, which it relies upon to generate the "strategy" of a new flea given the current market state (which includes the current flea strategies). A viable "flea strategy" is an expression that always evaluates to a number. However, to understand the context of this number, you must understand that it will be interpreted by the simulator as a "relative value". The following sections describes what a "relative value" is.

In turn, a relative value is itself just a number, and only has meaning in relation to other relative values generated by the same flea. In some abstract sense, a flea's relative value for good x stands for the flea's perceived utility of good x. The simulator uses these relative values to generate, for each flea, an exchange table of any good for any other good, where exchange rate of good x to good y (E_{xy}) is simply the relative value of x (R_x) divided by the relative value of y (R_y): $Exy = R_x/R_y$. In summary, a strategy should produce a set of numbers. Subsequently, the simulator extrapolates this set of numbers into an exchange table for each flea. The trading mechanics using these exchange tables is described later.

Table 1. The meanings of terms used in the text.

Term	Working definition	Critical function
Flea	Trading agent	Generates relative values for each good
Clan	Reproductively isolated group of fleas	Generates a viable strategy for each new flea
Market	Group of clans	Manages interaction between fleas of different clans by determining trade groups and good redistribution
Relative value	A flea's perceived utility for a given good	A value used in conjunction with other relative values of the same flea to generate an exchange table of proposed trading rates
Flea strategy	What a flea uses to compete for goods in the market	An expression that evaluates to a relative value for each good
Flea strategy generator	A clan-specific EC method for generating new flea strategies	Must output a viable strategy given the current fleas in a clan

1 Clojure is a recently created impure functional Lisp dialect which compiles to Java Virtual Machine bytecode. See: http://clojure.org.

2.2 Market Structure

Although useful, the definitions from Table 1 do not provide a complete story. It is also important to understand the elements composing a flea, a clan, and a market. These elements, and their components, are what we here call the "market structure". Refer to Table 2 for an outline of the market structure.

The market is composed of "clans", in which each clan is reproductively isolated from the others. Each clan is defined by its own EC method for generating new flea strategies. This "flea strategy generator" must take as input a list of the current fleas of the given clan, and must output a viable flea strategy. There are no restrictions or requirements on how the method could or should do this. In fact, the simulator itself will not know or care whether the method involves EC or not. The market determines trade groups and good redistribution each round. As such, it manages interactions between fleas of different clans.

Table 2. Outline of the structure in the simulator.

Term	Property	Description
Flea	Holdings (for each good)	An integer representing how much of each good the flea has
	Information about past trades	A look-up table with information on time and trade rate for every pair of goods, for all past trades
	Strategy	An expression that evaluates to a relative value
	Relative values (for each good)	A number representing the flea's perceived utility for each good
Clan	Fleas	A list of fleas currently in the clan
	Flea strategy generator	A function that outputs a viable flea strategy
	Population size	A constant integer for how many fleas compose the clan
Market	Clans	A list of clans in the market
	Trade group manager	A function that, for each round, outputs the trade groups
	Redistribution manager	A function that, for each round, outputs which clans will receive which repossessions

2.2.1 Structure of a Flea

The state of a given flea is characterized by: holdings for each good, information, a strategy, and relative values for each good. Holdings are simply an integer value for each good. "Information" can be thought of as a look-up table of past trading rates. The table can be indexed at any pair of goods, and it will return an associative pairing of times and lists of trade rates (for the first good to the second good) at those times. For example, let's say that a flea traded food for water at a rate of 1:2 exactly 3 time steps after it was born. Concurrently, it received information from its clan members that others traded food for water at a rate of 4:5, 6:7, and 8:9. Now, within its information map, indexed at "food:water", it will create a new sub-index "3", and within that store the list (1/2, 4/5, 6/7, 8/9). The simulator comes equipped with functions to retrieve information about pairs of goods and/or time steps. These functions can then be used in the function set for the flea strategy generators of clans. In this way, flea's can use exchange rate information within their strategy definitions.

2.2.2 Structure of a Clan

The state of a clan is given by: a list of fleas, a flea strategy generator, and a population size. A single clan has a constant population size. This is a simple measure to ensure each clan is always active, and that no EC reproductive method (i.e. a flea strategy generator) is ever killed off.

2.2.3 Structure of a Market

The state of a market is defined by: a list of clans in the market, a trade group manager function, and a redistribution manager function. Together, the latter two define how the market manages interactions between fleas of different clans. A trade group manager is a function that, based on the current time step, outputs a list of pairings of fleas (trade groups), thus defining which fleas will trade with which other fleas. The default trade group manager randomly pairs all fleas in all clans once every 50 rounds, and otherwise randomly pairs fleas only with fleas from the same clan. A redistribution manager is a function that, based on the current time step, outputs which repossessions will go to which clans. Each clan will then distribute its allotment equitably among its currently living members. By default, the redistribution manager will not disperse repossessed goods between clans. That is, repossessions will be clan-isolated by default: clan X will only receive repossessed goods from dead fleas from clan X.

2.3 Simulation Loop

Figure 1 is a flow chart representing a Flea Market Simulation run, with initialization remaining obscure for now. First, fleas are organized into trade-groups (pairs of fleas) by the trade-group manager. Next, fleas undergo trades through mechanics that will be explained later. For now, suffice to say that the trade is a maximally mutually beneficial trade of two goods, derived indirectly from the fleas' strategies. After all flea pairs have traded, all fleas consume essential goods. Any dead fleas are removed from the market, and their non-essential holdings are amassed as repossessions. Death is defined as having zero of any essential good. Then, clans are repopulated up to their constant size limit. New fleas are given a set starting amount of essential goods. Finally, repossessions are distributed to the clans (as managed by the redistribution manager), where they are then distributed equitably to fleas within clans to all the current living (both newly born and otherwise).

2.4 Initialization of a Run

For initializing a run, a simulation duration must be provided. A starting market is generated with a list of clans, where the defining factor of each clan is its flea strategy generator (EC method) and its size. The market can be initialized with a variety of optional settings. The most important settings are the trade group manager function and the redistribution manager function (both previously described).

Additional options allow for defining what goods are in the market, and/or for each good, the starting amount and use rate. If a good has a use rate, it is considered an essential good, such that fleas will die if they hold zero of it. New fleas are always given the starting amount of essential goods, but only the very first generation of fleas is given the starting amount of non-essential goods. Non-essential goods are not subsequently infused into the market, but rather recycled through redistribution from dead fleas to living fleas. In this way, the total amount of each non-essential good in the market is a constant, equal to the total number of fleas times the starting amount of that good.

By default, information is extremely local. Each flea only has information on its own trading rate history. However, optional settings allow for clan information sharing—where all members of a clan have access to the trading history for trades in which members of the same clan participated—and also entire-market information sharing—where every flea in the market has access to all trading rate histories for all trades in the market.

When initializing a clan, an optional clan setting is "maximum give fraction". This setting defaults to 0.5, and must be a value greater than 0 and less than or equal to 1. It defines the maximum fractional amount of any good a flea is allowed to give in one transaction. In a trade, the flea whose give rate (a function of maximum give rate and holdings) is more limited determines the actual trade amounts.

2.5 Trading Mechanics

For each trade group, the simulator determines which pair of goods, and what trade rate, would be most mutually beneficial to the two fleas. The simulator mandates which goods and at what rate the fleas trade. However, it is important to understand that although the fleas' strategies seem to play "passive" role in this process, they are, in fact, critical in determining the outcome. This is because each flea ultimately "proposes" exchange rates for

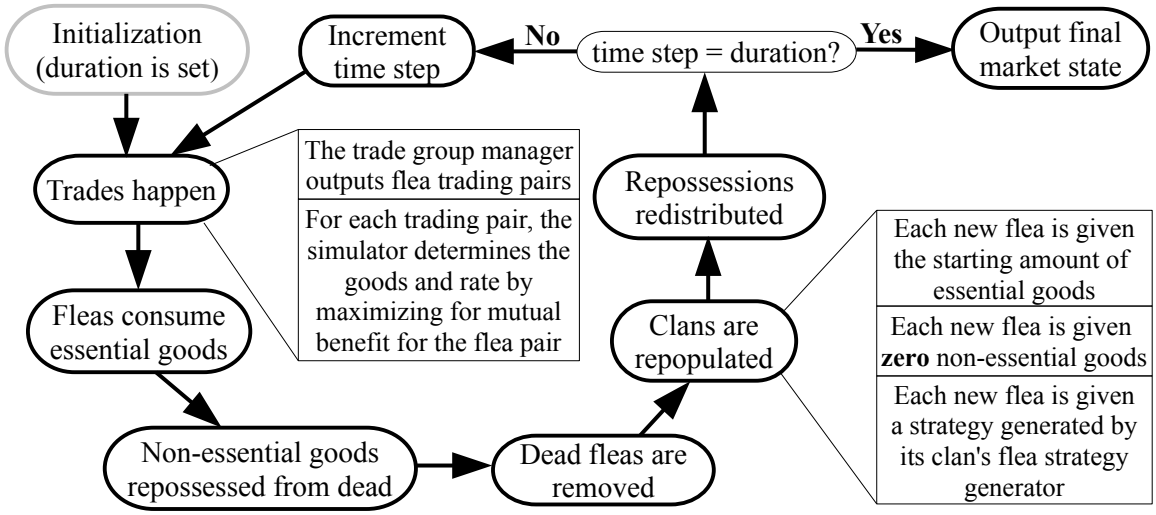

Figure 1. Flow chart of the simulation loop. Initialization is explained in the text.

every pair of goods. The simulator determines the maximally mutually beneficial trade through a stepwise process that is the same for every trade group:

1) As previously described, use the relative values of both fleas to determine their respective proposed exchange rates for every pair of goods. *Flea1:*E_{xy} and *Flea2:*E_{xy}.
2) For each pair of goods, calculate the geometric mean of the two fleas' proposed exchange rates. Each geometric mean represents the "ideal" trading rate for the given pair of goods. In essence, this step simulates "haggling" to a middle ground. Ideal rate, I = *square_root(Flea1:*E_{xy}**Flea2:*E_{xy}*)*.
3) Bound each ideal exchange rate based on the holdings of the two fleas for that pair of goods. Since good holdings are integer values, actual exchange rates must be constrained based on holdings. Actual rate, A = *bound_by_actual_trade_possibilities(I)*[2].
4) For each pair of goods, calculate the benefit of this actual exchange rate to each flea. Benefit is the absolute quotient of the actual exchange rate and each flea's proposed exchange rate. Here, the absolute quotient of x and y is defined as x/y if x>y or y/x otherwise. Benefit for *Flea1*, *Flea1:B* = *absolute_quotient(A, Flea1:*E_{xy}*)*; for *Flea2*, *Flea2:B* = *absolute_quotient(A, Flea2:*E_{xy}*)*.
5) For each pair of goods, take the minimum of the two benefit values (one per flea). This is the benefit value for this proposed trade. Trade benefit, B_{trade} = *minimum(Flea1:B, Flea2:B)*.
6) For all pairs of goods, select the trade that has the maximum benefit value. In turn, the trade rate will be equal to the actual exchange rate for that pair of goods. For the set of all pairs of goods, P, where $p \in P$, and $p{:}B_{trade}$ is the B_{trade} value for that pair of goods, select p where $p{:}B_{trade}$ is maximum. The trade rate R will be equal to $p{:}A$ (i.e. the value A for that pair of goods).

2.6 Properties of the Simulator

- Fleas are heterogeneous agents that trade in pairs.
- A flea strategy is open-ended. It is only required to be an expression that evaluates to a number.
- Functionally, a flea strategy is a method that outputs an exchange table for every pair of goods.
- A flea strategy *indirectly* determines the goods and ratio of each trade.
- In the default configuration, fleas have access to local information of only their own trading rate history. Optional settings allow for clan information sharing, and even entire-market information sharing.
- The market is characterize by overlapping generations.
- A flea strategy generator is also open-ended. It is only required that it be a function that takes a set of fleas as input, and outputs a viable flea strategy.
- Throughout a run, the total market amount for each non-essential good is constant.

2 Since holdings are integer values, the actual trade possibilities for a given flea are a function of its holdings and its clan-defined maximum give fraction. For example, for the possible trade of F1:F2 for food:water, let's say that I equals 2/9. However, F1 holds 2 food and F2 holds 6 water, where both of their maximum give fractions is equal to 0.5. As a result, F1 can only give 1 food, and F2 can only give 3 water. So the actual trade rate will be 1/3. Even if F2 holds 10 water, the actual trade rate would then be 1/5 due to rounding. The only way F1 and F2 could trade at a rate of 2/9 would be if F1 had at least 4 food and F2 has at least 18 water.

3. DISCUSSION

3.1 Potential Advantages

There is no stable equilibrium for exchange rates of goods: The simulator creates an unstable environment by leveraging the destabilizing effects of heterogeneous dynamic trading strategies. As a result, successful strategies may need to be dynamic themselves.

Fleas are theory blind: Flea strategies do not, by default, have any economic theory or principles embedded into them. Potentially, this means that fleas are not biased to act in ways that are already understood. Ideally, this allows for possibility of new theoretical discoveries.

Agent strategies are really abstract methods for determining relative need and utility of various goods: Although flea strategies only indirectly determine the goods and rate of trade, the strategies are elegant in that they abstractly represent a function for determining relative utility. This broadens the application of this simulator to modeling any "ecology" of agents competing for limited resources, where an agent is allowed to determine its own relative needs of those limited resources.

Information use of fleas may lead to sophisticated strategies: In addition to the pricing history information available to fleas, strategies can also indirectly utilize information about other fleas' strategies. Primarily, this may exist due to the overlapping of generations. When new fleas are generated based on extant flea strategies, they could theoretically "predict" the actions of other fleas by incorporating important pieces of their parents' code as conditionals in their own code. This principle also applies in the context of the genetic relatedness of peers in a population.

Different strategies can be compared in an open-ended user-defined context: Fleas from different clans can trade (with a user-defined schedule). This feature provides some way to understand the performance of flea strategies. Within a clan, fleas are constantly competing internally. Without comparison to some external strategies, quantifying strategy performance is limited to analyzing the actual code of individual strategies (yuck). These external strategies could be hand-coded or themselves could evolve. Which leads to an exciting implication: the potential for meta EC. In theory, one could use a "meta EC method" to generate flea strategy generating functions (the "lower level EC methods") and define some "meta fitness function" that compares whole clans' performances.

3.2 Potential Disadvantages

Analytically poor: Currently there are no built-in methods for analyzing strategies or markets using theories or methods from economics. Although theory blindness of flea strategies is potentially a good thing, theory blindness of analytic tools is not.

Simple results: Perhaps the dynamics are too simple to produce truly interesting strategies. The minimal nature of the simulator is useful for accessibility and analysis, but also may have the disadvantage of only allowing for simple or uninteresting strategies. Further testing is required to determine the utility of this simulator.

Potential problems with local optima: Due to the large gap in "need" between essential and non-essential goods, fleas may only be able to achieve locally optimal strategies that extend life slightly through extremely high relative valuation of essential goods. However, the function that determines trades does act to neutralize this behavior.

Flea strategies only indirectly determine the terms of a trade: This property limits the power of fleas to directly choose the terms of their own trades. By taking this power away from fleas, the potential for sophisticated strategies may be watered down.

3.3 Future Directions

Controlling good availability: Instability in the market dynamics is currently created by heterogeneous agent trading strategies. Controlling good availability would be another way to produce unstable environmental conditions, with different implications.

Incorporating more complicated good properties: Currently, goods have only one differentiable property: use rate. This may lead to strategies that are similarly limited in complexity. One possible additional good property could be transfer rate. If some goods were only slowly transferred after a trade, then those goods would potentially be less valuable. This property would be similar, but not identical, to the concept of liquidity in economics.

Incorporating game modifiers: It may be interesting to add goods that act to modify the behavior of other goods. For example, "ice" could be a good that decreases the use rate of food, but only to a certain extent. More complicated good:good interactions could significantly enhance the potential complexity of strategies without significantly damaging the minimal nature of the simulator.

Incorporating information trading: Information, if is useful, could also be something that fleas value and trade for. Incorporating this feature would require a big change to the program, but does seem enticing.

4. CONLUSION

The Flea Market Simulator is a minimal market trading simulator, composed of agents (fleas) that compete for essential goods. Fleas are divided into reproductively isolated sub-populations (clans)—where each clan has its own user-defined reproductive method, which generates flea strategies. When a flea dies (runs out of some essential good), it is replaced by a new flea with a new strategy generated by its clan's strategy generator. A flea's strategy can be thought of as a "utility determining function", where each flea's strategy is used to derive an exchange table of proposed trading rates for every pair of goods. Each round of simulation, fleas are organized into trading pairs through a user-define function. The simulator determines the "maximally mutually beneficial" trade for every trade group, by using both fleas' exchange tables to determine which pair of goods compose the ideal trade. "Haggling" is simulated by taking the geometric mean of proposed trading rates of those goods.

One important aspect of the simulator is an agent "strategy" is, abstractly, a method for determining the relative "need" for every good. This framework should provide a flexible and elegant way for fleas to determine trades. As a result of heterogeneous trading strategies, this simulator creates an unstable environment for its agents. In turn, in order for agents to survive for an extended period of time, they will likely have to develop highly dynamic strategies. Furthermore, through user-defined scheduling of flea trading between clans, strategies will intermingle. This intermingling allows for open-ended measures of strategy performance.

5. ACKNOWLEDGMENTS

Thanks especially to Lee Spector, who provided constant advice, support, and creative brainstorming throughout. Also thanks to Thomas Helmuth, Kyle Harrington, and Emma Tosch for their feedback, and to Hampshire College for support for the Hampshire College Institute for Computational Intelligence. This material is based upon work supported by the National Science Foundation under Grant No. 1017817. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

6. REFERENCES

[1] Andrews, M.; Prager, R. (1994). "Genetic Programming for the Acquisition of Double Auction Market Strategies", in Advances in Genetic Programming. Kinnear, K. Jr. (Ed). Cambridge, MA: The MIT Press.

[2] Arifovic, J. (1989). "Learning By Genetic Algorithms in Economic Environments". University of Chicago (Chicago, IL). Doctoral dissertation.

[3] Arthur, W. B. (1992). "On Learning and Adaptation in the Economy". Santa Fe Institute (Santa Fe, NM) Paper 92-07-038.

[4] Chen, S.-H. (2000). "Toward an agent-based computational modeling of bargaining strategies in double auction markets with genetic programming". Lecture Notes in Computer Science **1983**: 517-53.

[5] Lebaron, B. (2002). "Building the Santa Fe Artificial Stock Market". Working paper, Brandeis University (Waltham, MA).