An Agent-Based Computational Model for the Evolution of Trade Networks*

David McFadzean
Kumo Software Corporation
1400, 520-5th Avenue, S.W.
Calgary, AB T2P 3R7 Canada
david@kumo.com

Leigh Tesfatsion
Department of Economics
Iowa State University, Ames, IA 50011-1070
tesfatsi@iastate.edu

June 1997

Abstract. This paper describes an agent-based computational model for studying the formation and evolution of trade networks in decentralized market economies. A virtual economic world is constructed, populated by heterogeneous endogenously interacting traders with internalized data and modes of behavior. This virtual world can be used to study the formation and evolution of trade networks under alternatively specified market structures at three different levels of analysis: individual trade behavior; trade interaction patterns; and social welfare.

1 Introduction

Agent-based computational economics (ACE) is the computational study of economies modelled as evolving decentralized systems of autonomous interacting agents. A key concern of ACE researchers is to understand the apparently spontaneous appearance of global regularities in economic processes, such as the unplanned coordination of trading activities in decentralized market economies that economists associate with Adam Smith's invisible hand. The challenge is to explain how these global regularities arise from the local interactions of autonomous agents channeled through actual or potential economic institutions

*This paper is an abbreviated version of [4]. See http://www.econ.iastate.edu/tesfatsi/ for program code and related materials. Corresponding author: L. Tesfatsion.

rather than through fictitious coordinating mechanisms such as imposed equilibrium conditions.

The present paper discusses the C++ implementation of a particular ACE model developed by Tesfatsion [7] to study the endogenous formation and evolution of trade networks. This model, referred to as the *trade network game* (TNG), extends to an economic setting an earlier model ([1]), [6]) combining evolutionary game play with endogenous partner selection. In the TNG, successive generations of resource-constrained traders choose and refuse trade partners on the basis of continuously updated expected payoffs, engage in risky trades modelled as two-person games, and evolve their trade behavior over time.

In particular, traders are instantiated as autonomous endogenously interacting software agents ("tradebots") with internal behavioral functions and with internally stored information that includes identifiers for other tradebots. The tradebots can therefore display anticipatory behavior (expectation formation), and they can communicate with each other at event-triggered times. The tradebots use these and other capabilities to determine their trade partners and to evolve their trade behavior. The modular design of the TNG implementation permits experimentation with alternative specifications for market structure, trade partner matching, trading, expectation formation, and trade behavior evolution. All of these specifications can potentially be grounded in tradebot-initiated activities. The TNG implementation thus facilitates the general ACE study of trade networks.

The basic TNG model is briefly described in Section 2, the implementation of the TNG is outlined in Section 3, and the experimental study of the TNG is discussed in Section 4. Concluding remarks are given in Section 5.

2 The Basic Trade Network Game

The TNG consists of a collection of traders that evolves over time. As depicted in Table 1, each trader in the initial trader generation is constructed and assigned a trade strategy. The traders then enter a nested pair of cycle loops during which they repeatedly determine trade partners, carry out trades, update their expectations, and evolve their trade behavior. The current TNG implementation uses the particular specifications for these activities detailed in [7]. For completeness, these specifications are briefly reviewed below.

Alternative market structures are currently imposed in the TNG through the prespecification of buyers and sellers and through the prespecification of quotas on offer submissions and acceptances. More precisely, the set of players for the TNG is the union $V = B \cup S$ of a nonempty subset B of *buyer traders* who can submit trade offers and a nonempty subset S of *seller traders* who can receive trade offers, where B and S may be disjoint, overlapping, or coincident. In each trade cycle, each buyer m can submit up to O_m trade offers to sellers and each seller n can accept up to A_n trade offers from buyers, where the offer quota O_m and the acceptance quota A_n can be any positive integers.

Although highly simplified, these parametric specifications permit the TNG

```
int main () {
  Init() ;                          // Construct initial trader generation
                                    //   with random or specific trade strategies.
  For (G = 0,...,GMAX-1) {          // Enter the generation cycle loop.
                                    // Generation Cycle:
    InitGen();                      //   Configure traders with user-supplied
                                    //     parameter values (initial expected
                                    //     payoff levels, quotas,...).
    For (I = 0,...,IMAX-1) {        //   Enter the trade cycle loop.
                                    //   Trade Cycle:
      MatchTraders();               //     Determine trade partners,
                                    //       given expected payoffs,
                                    //       and record refusal and
                                    //       wallflower payoffs.
      Trade();                      //     Implement trades and
                                    //       record trade payoffs.
      UpdateExp();                  //     Update expected payoffs
    }                               //       using newly recorded payoffs.
                                    //   Environmental Step:
    AssessFitness();                //     Assess and output trader information.
    Dump();                         //     Output fitness statistics for the
                                    //       current trader generation.
    EvolveGen();                    //   Evolution Step: Evolve a new trader generation.
  }
  Return 0 ;
}
```

Table 1: Pseudo-Code for the TNG

to encompass two-sided markets, markets with intermediaries, and markets in which all traders engage in both buying and selling activities. For example, the buyers and sellers might represent workers and employers, lenders, banks, and borrowers, or barter traders. The offer quota O_m indicates that buyer m has a limited amount of resources (labor time, deposits, apples,...) to offer, and the acceptance quota A_n indicates that seller n has a limited amount of resources (job openings, investment earnings, oranges,...) to provide in return.

Example: A Labor Market With Endogenous Layoffs and Quits

The set B consists of M workers and the set S consists of N employers, where B and S are disjoint. Each worker m can make work offers to a maximum of O_m employers, or he can choose to be unemployed. Each employer n can hire up to A_n workers, and employers can refuse work offers. Once matched, workers choose on-the-job effort levels and employers choose monitoring and penalty levels. An employer fires one of its current workers by refusing future work offers from this worker, and a worker quits his current employer by ceasing to direct work offers to this employer. This TNG special case thus extends the standard modelling of labor markets as assignment games by incorporating subsequent strategic interactions between matched pairs of workers and employers and by having these interactions iterated over time.

The determination of trade partners in the TNG is currently implemented using a modified version of the well-known Gale-Shapley deferred acceptance mechanism (see [5]). This modified mechanism, hereafter referred to as the *de-*

ferred choice and refusal (DCR) mechanism, presumes that each buyer and seller currently associates an expected payoff with each potential trade partner. Also, each buyer and seller is presumed to have an exogenously given *minimum tolerance level*, in the sense that he will not trade with anyone whose expected payoff lies below this level.

The DCR mechanism proceeds as follows. Each buyer m first makes trade offers to a maximum of O_m most-preferred sellers he finds tolerable, with at most one offer going to any one seller. Each seller n in turn forms a waiting list consisting of a maximum of A_n of the most preferred trade offers he has received to date from tolerable buyers; all other trade offers are refused. For both buyers and sellers, selection among equally preferred options is settled by a random draw. A buyer that has a trade offer refused receives a negative *refusal payoff*, R; the seller who does the refusing is not penalized. A refused buyer immediately submits a replacement trade offer to any tolerable next-most-preferred seller that has not yet refused him. A seller receiving a new trade offer that dominates a trade offer currently on his waiting list substitutes this new trade offer in place of the dominated trade offer, which is then refused. A buyer ceases making trade offers when either he has no further trade offers refused or all tolerable sellers have refused him. When all trade offers cease, each seller accepts all buyer trade offers currently on his waiting list. A trader that neither submits nor accepts trade offers during this matching process receives a *wallflower payoff*, W.

The buyer-seller matching outcomes generated by the DCR mechanism exhibit the usual static optimality properties associated with Gale-Shapley type matching mechanisms. First, any such matching outcome is core stable, in the sense that no subset of traders has an incentive to block the matching outcome by engaging in a feasible rearrangement of trade partners among themselves [7, Proposition 3.2]. Second, define a matching outcome to be B-optimal if it is core stable and if each buyer matched under the matching outcome is at least as well off as he would be under any other core stable matching outcome. Then, in each TNG trade cycle, the DCR mechanism yields the unique B-optimal matching outcome as long as each trader has a strict preference order over the potential trade partners he finds tolerable [7, Proposition 3.3].

Trades are currently modelled in the TNG as two-person prisoner's dilemma (PD) games. For example, a trade may involve the exchange of a good or service of a certain promised quality in return for a loan or wage contract entailing various payment obligations. A buyer participating in a trade may either cooperate (fulfill his trade obligations) or defect (renege on his trade obligations), and similarly for a seller. The range of possible payoffs is the same for each trade in each trade cycle: namely, L (the sucker payoff) is the lowest possible payoff, received by a cooperative trader whose trade partner defects; D is the payoff received by a defecting trader whose trade partner also defects; C is the payoff received by a cooperative trader whose trade partner also cooperates; and H (the temptation payoff) is the highest possible payoff, received by a defecting trader whose trade partner cooperates. More precisely, the payoffs are assumed to satisfy $L < D < 0 < C < H$, with $(L + H)/2 < C$.

The TNG traders use a simple form of learning algorithm to update their expected payoffs on the basis of new payoff information. Suppose, for example, that a trader v receives a payoff P from an interaction with another trader k, either a refusal payoff R or some type of trade payoff. Trader v immediately updates his current payoff count with k, $N_v(k)$, by 1, and then updates $\omega_v(k)$, the *memory weight* that he currently associates with k, by setting $\omega_v(k) = N_v(k)/[N_v(k)+1]$. The expected payoff $U_v(k)$ that v currently associates with k is then replaced by $\omega_v(k)U_v(k) + [1-\omega_v(k)]P$. This implies that trader v gives equal weight to each payoff that he has obtained in past interactions with tradebot k.

The trade behavior of each trader, whether he is a pure buyer in $V-S$, a buyer-seller in $B \cap S$, or a pure seller in $V-B$, is currently characterized by a finite-memory pure strategy for playing a PD game with an arbitrary partner an indefinite number of times, hereafter referred to as a *trade strategy*. Each trader thus has a distinct trading personality even if he engages in both buying and selling activities. At the commencement of each trade cycle loop, traders have no information about the trade strategies of other traders; they can only learn about these strategies by engaging other traders in repeated trades and observing the payoff histories that ensue. Moreover, each trader's choice of an action in a current trade with a potential trade partner is determined entirely on the basis of the payoffs obtained in past trades with this same partner. Thus, each trader keeps separate track of the particular state he is in with regard to each of his potential trade partners.

The evolution of the current generation of traders at the end of their trading activities is meant to reflect the formation and transmission of new ideas rather than biological reproduction. The only aspect of a trader that currently evolves is his trade strategy. Specifically, successful trade strategies are mimicked and unsuccessful trade strategies are replaced by variants of more successful strategies.

3 TNG Implementation

The TNG is implemented with support of *SimBioSys*, a general C++ class framework for evolutionary simulations developed by McFadzean [3]. This section briefly outlines this implementation, hereafter referred to as the TNG platform. The static and dynamic structures of the TNG platform are discussed in turn.

3.1 TNG Class Definitions and Relationships

The static structure of the TNG platform is expressed through definitions and relationships for three principal classes:

- *tngSimulation*, which manages the overall simulation;
- *tngPopulation*, which manages the evolution of the traders;

```
class tngTradeBot
{
    Public Access:
        // Internalized Institutional Rules
                Methods for determining my trade partners;
                Methods for conducting my trades;
        // Other Publicly Accessible Methods Used by This TradeBot
                Methods for constructing my trade strategy;
                Methods for updating my expected payoffs;
                Methods for calculating my fitness score.
    Private Access Only:
                Data about myself;
                Data I have recorded about other tradebots;
        // Data permitting this tradebot to communicate with other tradebots
                Addresses for all tradebots.
};
```

Table 2: Schematic Description of a Tradebot

- *tngTradeBot*, which simulates a single trader, either a pure buyer, a buyer-seller, or a pure seller.

The TNG platform constructs a single instance of tngSimulation, which in turn constructs a single instance of tngPopulation; and tngPopulation then constructs a collection of tngTradeBot instances hereafter referred to as *tradebots*.

The heart of the TNG platform is the representation of each trader as a tradebot, i.e., as an instance of the class tngTradeBot. All of the trade-related activities of a tradebot are implemented as tradebot methods, meaning they are implemented by means of member functions of tngTradeBot. In principal, any attribute of a tradebot, either its parametric configuration or the form of its member functions, could be subject to evolutionary selection pressures. In the current implementation, however, the only aspect of a tradebot that evolves over time is its trade strategy for playing iterated prisoner's dilemma games with other tradebots.

A schematic description of the internal structure of a tradebot is given in Table 2. Two features of this description are of particular interest. First, institutional constraints regarding the determination of trade partners and the conduct of trades are expressed through the form of the tradebot's member functions. Second, each tradebot stores an identifier for itself and for each other tradebot, which permits the tradebot to identify itself to other tradebots it interacts with and to pass messages to other tradebots at event-driven times.

Prior to running the TNG platform, the user can supply values for the TNG parameters in a configuration file, tng.ini. A sample specification of tng.ini is given in Table 3. Note that tng.ini currently only accommodates identical offer quotas for all buyer tradebots, identical acceptance quotas for all seller tradebots, and identical initial expected payoff levels for all tradebots. Also the minimum tolerance level for each tradebot does not currently appear in tng.ini; it is hardcoded to 0.

```
// VIRTUAL ENVIRONMENT PARAMETERS
    GMax = 50                  // Total number of generations.
    IMax = 150                 // Number of trade cycles in each trade cycle loop.
    RandomSeed = 20            // Seed for pseudo-random number generator.
    MutationRate = .005        // GA bit toggle probability.
    FsmStates = 16             // Number of internal FSM states.
    FsmMemory = 1              // FSM memory (in bits) allocated to past move recall.
    TraderCount = 30           // Total number of tradebots.
    SellerCount = 30           // Number of pure sellers and buyer-sellers.
    BuyerCount = 30            // Number of pure buyers and buyer-sellers.
    Elite = 20                 // Number of elite tradebots.
    RefusalPayoff = -0.6       // Payoff R received by a refused tradebot.
    WallflowerPayoff = +0.0    // Payoff W received by an inactive tradebot.
    BothCoop = +1.4            // Mutual cooperation PD payoff, C.
    BothDefect = -0.6          // Mutual defection PD payoff, D.
    Sucker = -1.6              // Lowest possible PD payoff, L.
    Temptation = +3.4          // Highest possible PD payoff, H.
// TRADEBOT PARAMETERS
    BuyerQuota = 1             // Buyer offer quota.
    SellerQuota = 30           // Seller acceptance quota.
    InitExpPayoff = +1.4       // Initial expected payoff level.
```

Table 3: Sample Specification for the TNG Configuration File

3.2 TNG Dynamic Structure

The TNG platform uses the *SimBioSys* hierarchy of simulation cycles to implement the dynamic structure of the TNG shown in Table 1.

The first step executed by the TNG main program, Main(), is the creation of derived instance of the class tngSimulation, called tng. Main() next invokes a tng member function, Init(), which configures the virtual environment with user-supplied parameter values extracted from the configuration file tng.ini. In addition, in the current TNG implementation, the trade strategy assigned to each tradebot in the initial generation is randomly determined.

The generation cycle loop then commences. At the start of the first generation cycle, Main() invokes a tng member function, InitGen(), that prompts each tradebot to configure itself with user-supplied parameter values extracted from tng.ini. The result is that each tradebot is publicly identified as a pure buyer, a buyer-seller, or a pure seller, and each tradebot is characterized by parameter values in accordance with its type. In particular, in the current TNG implementation, each pure buyer and buyer-seller has an identical offer quota, each pure seller and buyer-seller has an identical acceptance quota, and each tradebot has an identical initial expected payoff level that represents its initial assessment for each of its potential trade partners.

The initial generation of tradebots then participates in a trade cycle loop consisting of IMax trade cycles, where IMax is extracted from the TNG configuration file. During each trade cycle the tradebots determine their trade partners via the DCR matching mechanism and engage in trades modelled as two-person prisoner's dilemma games. Any time during the trade cycle that a tradebot receives a payoff as the result of an interaction with another tradebot, whether it is a refusal payoff or a trade payoff, the tradebot updates its expected payoff

for this other tradebot using the learning algorithm outlined in section 2. The tradebots then enter into an environmental step in which they are sorted by fitness, where the fitness score of a tradebot is measured as the average payoff per interaction that it obtained during the course of the previous trade cycle loop.

Finally, the tradebots enter into an evolution step. The trade strategies associated with the most fit tradebots (the elite) are retained unchanged, where the number of elite is extracted from the TNG configuration file. The trade strategies associated with the remaining tradebots are then replaced by variants of more successful trade strategies. As detailed in [4], the latter evolutionary process is currently implemented by means of a standardly specified genetic algorithm employing both mutation and two-point crossover operations. The tradebots then enter into a new generation cycle and the entire process repeats.

A more detailed discussion of the implementation of these cycle and step activities is given in [4].

4 Experimental Study of the TNG

The TNG platform detailed in the previous section permits the experimental study of TNG outcomes at three different levels of analysis. First, individual tradebot attributes can be directly sampled to assess the evolution of trader types. Second, detailed information concerning who is trading with whom can be collected to track the endogenous formation and evolution of trade networks. Third, the fitness scores attained by the tradebots in each successive generation can be used to assess social welfare.

With regard to individual attributes, the trade strategies of the tradebots can be directly examined to determine their stance toward strangers and to assess the types of trade behaviors that they are potentially capable of expressing in repeated trades. More precisely, trade strategies are currently represented as finite state machines (FSMs) with bit string encodings, and these encodings can be output for direct assessment. As shown in Tesfatsion [7], in some circumstances this information may permit the analytical determination of trade networks, i.e., who will actually end up trading with whom during the course of a trade cycle loop.

In general, however, even if the number of FSM states is restricted to be small, the ability to choose and refuse trade partners on the basis of continually updated expected payoffs results in such complicated contingencies that the prediction of trade network formation from an ex ante assessment of individual trade strategies is extremely difficult. It is therefore important to be able to obtain information on the trade networks that form during each TNG simulation run.

To aid in this task, at the end of each trade cycle loop each tradebot is prompted to output the data it currently has stored for each of its potential trade partners. This data includes: the sum of payoffs received from interactions with this potential trade partner; the number of payoffs received from interactions with this potential trade partner; the current expected payoff associated with

this potential trade partner; the number of trades actually undertaken with this potential trade partner; and, if trades have taken place, a complete ordered list of the trade partner's actions (cooperate or defect) in each of these trades. The user can thus observe who has traded with whom, and how often, during the course of the trade cycle loop, as well as the degree to which the trades have resulted in mutual cooperation, exploitation (successful defection against a cooperating partner), or mutual defection.

Finally, with regard to social welfare, various fitness statistics can be calculated for each successive generation of tradebots. For example, in the current implementation of the TNG, the minimum, average, and maximum fitness scores attained by each generation are calculated and recorded. These fitness scores give a rough idea of the level of social welfare attained by the tradebots during the course of a single evolutionary run.

To illustrate, Figure 1 depicts the minimum, average, and maximum fitness scores attained by 50 successive generations of tradebots starting from initially random trade strategies, given the parameter settings in Table 3. These parameter settings were chosen to match as closely as possible the "standard scenario" used as a benchmark case by Ashlock et al. [1, Table 3] for an IPD game with choice and refusal of partners. The latter authors found that rapid convergence to mutual cooperation took place for this benchmark case, and the TNG results qualitatively replicate this finding. The reason for this rapid convergence becomes clear from an inspection of individual strategies: the refusal mechanism helps cooperative players protect themselves against defectors without having to defect themselves, and the choice mechanism helps cooperative players direct their offers to other cooperative types. Consequently, nice players are not as evolutionarily disadvantaged as they tend to be with random or round-robin partner matching.

More generally, the fitness scores attained by the tradebots can be used to judge the relative merits of alternative market structures with regard to inducing good social outcomes. For example, preliminary TNG experimental findings reported in Tesfatsion [7] suggest that the conventional optimality properties used by economists to evaluate market structures in static contexts, such as core stability and Pareto efficiency, may be inadequate measures of optimality from an evolutionary perspective. In particular, use of the deferred choice and refusal (DCR) partner matching mechanism outlined in section 2 imposes high transactions costs on the tradebots that these conventional optimality properties do not take into account.

5 Concluding Remarks

The TNG platform has been developed to facilitate the study of trade in decentralized market economies from a bottom-up perspective. The key feature of the TNG platform is that traders are modelled as autonomous endogenously interacting software agents ("tradebots") with internally stored data and with internal behavioral functions that can evolve over time.

The platform design is modular and extensible, reflecting the view that the current implementation is only a first step in a long journey to come. Specifically, in the current implementation the tradebots are market situated, but explicit price setting behavior has not yet been introduced. In addition, no attempt has yet been made to exploit the capability of the underlying *SimBioSys* abstract base classes to situate the tradebots within a virtual spatial environment that reflects biological processes (e.g., plant growth) as well as physical laws (e.g., conservation of energy).

In the past, tractability issues have generally forced economists to focus their attention on specialized aspects of economic behavior, without detailed consideration of psychological, sociological, biological, or physical constraints. The recent development of agent-based frameworks such as the TNG platform raise new questions concerning the appropriate bounds of economic analysis, for these frameworks can potentially model social activity from a much more inclusive perspective. Indeed, as stressed in Epstein and Axtell [2], such frameworks may at last provide a common paradigm for social science as a whole.

References

[1] D. Ashlock, M. D. Smucker, E. A. Stanley, and L. Tesfatsion. Preferential partner selection in an evolutionary study of prisoner's dilemma. BioSystems. 37:99-125, 1996.

[2] J. Epstein and R. Axtell. *Growing Artificial Societies: Social Science from the Bottom Up*. MIT Press, Cambridge, 1996.

[3] D. McFadzean. SimBioSys: A class framework for evolutionary simulations. Master's Thesis. Department of Computer Science, University of Calgary, Alberta, Canada, 1995. [See http://www.kumo.com/ david/SimBioSys/ for code and related materials.]

[4] D. McFadzean and L. Tesfatsion. A C++ Platform for the Evolution of Trade Networks. ISU Economic Report Number 39. December 1996. To appear in Computational Economics (Special Issue on Programming Languages). [See http://www.econ.iastate.edu/tesfatsi/ for code and related materials.]

[5] A. Roth and M. A. O. Sotomayor. *Two-sided matching: A study in game-theoretic modeling and analysis*. Cambridge University Press, Cambridge, 1990.

[6] E. A. Stanley, D. Ashlock, and L. Tesfatsion. Iterated prisoner's dilemma with choice and refusal of partners. pp. 131–175 in C. Langton (ed.), *Artificial Life III*. Proceedings Volume 17. Santa Fe Institute Studies in the Sciences of Complexity, Addison Wesley, 1994.

[7] L. Tesfatsion. A trade network game with endogenous partner selection. ISU Economic Report No. 36, April 1995. [An abbreviated version appears as pp. 249–269 in H. Amman et al. (eds.), *Computational Approaches to Economic Problems*. Kluwer Academic Publishers, 1997.]

CAPTION FOR FIGURE 1

(Figure 1 is available upon request from L. Tesfatsion)

Fig. 1. TNG simulation run depicting the minimum, average, and maximum fitness scores attained by 50 successive generations of tradebots starting from initially random trade strategies. By generation 25 the average and maximum fitness scores closely fluctuate around the mutually cooperative level 1.4. The parameter settings for this run are explained more fully in Table 3.