

**A Comparison of Two Methods of Network Formation:
Top-Down and Bottom-Up**

Submitted in partial fulfillment of the requirements of the Non-Thesis option for the
Master's of Arts of Geography

James Tedrick
Advisor: Catherine Dibble
07/17/2003

Abstract:

Multiple types of networks play key roles in modern society. Transportation networks, a type traditionally studied by geographers, reduce the time and cost of transporting goods across a landscape. Inefficient networks impose unneeded costs on a society and hinder its development by wasting resources. This paper presents two computer tools designed to observe the performance of the different ways networks could be formed. The first, an agent-based model, simulates the creation of a network among nodes by having the nodes determine to best locations for network links in a ‘bottom-up’ fashion. A genetic algorithm searches for the most efficient network configuration without regard to any node’s preference. This works in a top-down fashion, with all decision-making occurring in a single authority. The results generated by these models over a set of 50 random landscapes are compared, with the finding that there is no significant difference between the two types of tools.

Table of Contents:

Introduction— Networks	1
Agent-Based Models.....	4
Genetic Algorithms.....	6
Methodology.....	8
Agent-Based Implementation	10
Genetic Algorithm Implementation	12
Experimental Design.....	13
Results.....	14
Agent-Based Models.....	14
Genetic Algorithms.....	16
Analysis.....	18
Discussion.....	19
Extensions.....	21
Conclusion	23
References.....	25
Appendix A: Code of the Agent-Based Model.....	29
Appendix B: Code of the Genetic Algorithm	33

List of Figures:

Figure 1: An example of shortcuts.....	9
Figure 2: Result of a single agent-based model run.....	15
Figure 3: Results of all agent-based model runs.....	16
Figure 4: Result of a single genetic algorithm run.....	17
Figure 5: Comparison of the agent-based model and the genetic algorithm	18

I. Introduction— Networks

Networks form the heart of modern economic geography. They allow faster transmission of goods and information than could be accomplished by walking over the surface of the earth. Classically, the study of networks in economic geography focused on transportation networks that allowed people and goods to flow; recently, communication networks, which are conduits for flows of information, have become more visible because of their role in modern society.

Networks can also inhibit growth and development. Many types of networks, such as roads, railroads, and communications networks, incur high capital costs. Many types of networks also generate operating expenses to maintain the condition of the network. On an inefficient network, these expenses can overwhelm the benefits provided (Rioja, 2003).

In addition to geography, networks are of interest in several related fields. Civil engineers and urban planners also study transportation networks, as they represent vital components of the infrastructure of our world (Lyons, Chatterjee, and Marsden, 1999). Communication networks, another component of the infrastructure of our society, are of interest to information technologists as well as to geographers (Tutschku and Tran-Gia, 1998). Both economists and sociologists study social networks, the connections between people that can allow for trade or group dynamics (Bala and Goyal, 2000; Watts, 2001).

By understanding how networks form, researchers can gain greater insight into how modern transportation systems developed. The ability to model network evolution also allows the study of how a network reacts to the addition or subtraction of links. Simulating the effects of technological change on a network can also provide insight into

the evolution of settlement patterns.

Recent research in other fields also recognizes the importance of other networks. In the fields of economics and sociology, research in social networks, how people are interconnected, has become important areas of research. While these networks do not necessarily have spatial characteristics (most research approaches are aspatial), they feature many of the same characteristics and are subject to similar analyses as spatial transportation networks (Jackson and Watts, 2002).

Much of the research in transportation networks is dominated by the civil engineering and transportation planning realms, which, for the most part, focus on the operational characteristics of the network and not its structure. *Transportation Research* is a well-respected set of six specialized journals; these specializations include *Policy and Practice*, *Methodological*, *Emerging Technologies*, *Transport and Environment*, *Logistics and Transportation Review* and *Traffic Psychology and Behaviour*. Only the first and fifth of these headings indicate the possibility of looking at how transportation networks connect locations.

Within geography, the study of networks includes both practical applications in economic geography, such as the analysis of airline networks, and the evolution of structure of networks and economic activities caused by the networks (Graham, 1995; Peeters, Thisse, and Thomas, 1998). Both geographers and planners study accessibility, the role of transportation networks in shaping society by enabling or limiting individual's abilities (Shen, 2001).

There are several ways to design and utilize resources. Two contrasting examples are the top-down and bottom-up approaches. The top-down approach consists of a single

or small group of authorities designing and directing the utilization of resources. This method is generally associated with centralized planning forms of governments. A bottom-up approach's characteristics include direct beneficiaries and other people directly involved with the resource designing the utilization; in economics, the ideal free market, where the only actors are buyers and sellers, typifies this concept.

One of the major advances to theoretical research has been the introduction of computer tools; among others two tools of import are agent-based simulation models and genetic algorithms. Agent-based models enable the recreation of real-world events and processes, allowing the researcher to modify and examine the processes involved quickly and allowing the creation of scenarios to more accurately predict what would happen with a given set of conditions, not just what has happened (Langton, 1988). Genetic algorithms are searching tools; they encode the parameters of a solution in such a way that parts of one possible solution may be combined with parts of other possible solutions (Holland, 1992). These possible solutions are evaluated, and those that perform the best are used to create a new set of possible solutions. This process quickly leads to solutions that highly correspond to the criteria of the evaluation function and, therefore, the problem.

The nature of these tools lend themselves well to use in different organization structures. Agent-based models are composed of multiple independent agents interacting to create a process; it behaves in a bottom-up fashion. Genetic algorithms can be designed to search for an optimal result for a complex system as a whole; as a single authoritative source, it typifies the idea of centralized planning.

Abstracting these organizational models to computer programs allows us to create

scenarios in which the two can be compared. In this paper, we will use the two to model the creation of a network on a landscape with thirty prospective nodes. The agent-based model will consist of each node attempting to maximize its benefit according to a spatial interaction model; the genetic algorithm will attempt to produce a solution for the entire landscape that brings maximum benefit under the same spatial interaction model. The total benefit provided to the landscape will be used as a comparative metric.

The next sections of this paper present a basic orientation to agent-based modeling and genetic algorithms. The fourth section explains how the agent-based model and genetic algorithm were designed and how they were compared. The results are then presented, analyzed, and discussed. The paper concludes with a discussion of possible extensions to the agent-based model.

II. Agent-Based Modeling

One of the newest modeling tools already in use is agent-based simulation modeling. These models allow for the study of complex systems by providing a base to create multiple, relatively simple actors and a method of observing the interactions of the actors within the simulation.

Agent-based models originated out of theoretical work of John von Neumann, who proposed a machine that was capable of producing a copy of itself and giving that copy the ability to create further copies (Langton, 1988). This concept was transformed into cellular automata by plotting the machines as cells of a grid on paper. Cellular automata are a type of modeling tool that creates a tessellated landscape where each cell may have one of several defined values. The rules for assigning a value to a cell are uniform for every cell within the landscape. The rules are then applied iteratively over

the landscape, with the value of a cell being changed as the cell fails the criteria for its current value and meets the criteria for another value. Often the criteria for the value of a cell are dependent on the local neighborhood of the cell. One of the most well known cellular automata is the Game of Life, developed by John Horton Conway (Gardner, 1970).

Within a cellular automaton, each cell is required to have the same behavior. In contrast, in an agent-based model, each agent forms its own decision. This allows for different types of agents to interact, greatly increasing the complexity of the simulation. Agents may also be mobile, moving to different locations on a landscape and different situations with respect to other agents; the individual cells in a cellular automaton are fixed. Also, while traditionally cellular automata are developed over a raster landscape¹, agent-based models can occur in a raster landscape, a vector graph, or models can use the two in concert.

Early development in agent-based models focused in two areas: simulating natural behavior in animals and economics. Several of the original uses for agent-based models were to simulate group behavior in animals, such as Reynolds's (1987) model of flocking and herding behavior or food collection by ants (Bonabeau, Dorigo, and Theraulaz, 1999). Economists have focused on the ability to use agents in trading games, allowing for researchers to study economic behavior without having other forms of behavior interfere (McFadzean, Stewart and Tesfatsion, 2001).

Recently, different fields have begun to use agent-based models. Sociologists have been able to simulate group interactions and social phenomena, such as the

¹ Recently, graph-based cellular automata have been developed to overcome this limitation (O'Sullivan, 2001).

occurrence of civil disobedience and genocide (Epstein, 2002). The infection and transmission of diseases can be modeled for use in epidemiological simulation (Eubank, 2002).

Agent-based models are becoming more visible in areas of interest to geographers. One area in which agent-based models began is the interaction of different animal or plant types, such as the succession of different trees within a forest (Savage, Sawhill, and Askenazi, 2000). The interaction of humans and the environment can be modeled, where the environmental limitations affect the ability of settlements to collect resources and perpetuate (Axtell, et al., 2002). Interactions between people over a landscape, as in location selection, can also be researched (Dibble, 2001).

III. Genetic Algorithms

Genetic algorithms are a method to efficiently search for solutions to complex problems. The basis for genetic algorithms derives from biology and evolutionary theory (Holland, 1992). The core of a genetic algorithm is the string— a possible solution to the problem arranged in a linear format. Each string is composed of several genes, each representing a particular parameter of the solution. The string is processed by an evaluation function, which calculates the effectiveness of the solution proposed by the values of the parameters contained within the string and assigns a “fitness score” for comparative purposes.

Genetic algorithms are an iterative process. The candidate solutions that best meet the criteria are selected, proportional to their fitness score, to create the possible candidates the next turn. Segments of different existing candidates are combined to form the next round of prospective solutions. Mutating of some parts of the candidates’ strings

also modifies these candidates; the probability for mutation is set to a low level so that most of the improvement is derived from the fitness of the segments of the previous candidate solutions.

Genetic algorithms originated with the work of John Holland in the mid-1970s. The first practical applications were developed in the early 1980s to be run on high-performance supercomputers. Today, genetic algorithms can be used with high-end personal workstations; the primary determinant for the computing power needed is the complexity of the evaluation function and the how many different solutions the algorithm must examine to optimize the problem. Complex problems still require relatively powerful computers.

The transportation field has found several uses for genetic algorithms. One of the most famous problems that use genetic algorithms is the ‘traveling salesman’ problem, in which a route is identified that minimizes distance or cost for connecting a set of locations (Qu and Sun, 1999). Such system optimization is an area where genetic algorithms perform well; another example is the logistics of vehicle scheduling is being done on public transport systems (Bielli, Caramia, and Carotenuto, 2002). The design of infrastructure can also be aided through genetic algorithms; physical characteristics and constraints can be used as the evaluation criteria (Fwa, Chan, & Sim, 2002).

Within geography, genetic algorithms have been used primarily within the field of economic geography. Solutions for location-allocation problems, in which the locations of facilities are determined, can be found using genetic algorithms (Dibble and Densham, 1993). Genetic algorithms can also be used to develop other complex tools by acting as the ‘training’ stage, discovering which set of parameters work the best in evaluating data;

for example, a genetic algorithm can be used to help identify neural networks that are good at solving problems (Fischer, M., Leung, Y. 1998).

Genetic algorithms are also suitable for solving a variety of network problems. This is true over a variety of network types. Some examples of these types include transportation network connecting different locations together (Drezner and Wesolowsky, 2003), social networks showing how people are connected to each other in an organization (DeCanio, Dibble, Amir-Atefi, 2000), or the layout of a computer network (Deeter and Smith, 1998).

IV. Methodology

The agent-based model and genetic algorithm were developed using a gravitational model for spatial interaction. This model (eq. 1) predicts the interaction between two places (I_{ij}) as being dependent on a distance measure between two places and some form of interaction value at each node, such as demand or population (Sheppard, 1995). The model is calibrated and converted into usable units with the constant k . For the model being used, the interaction weight of all the nodes will be identical, thus $k \cdot P_i P_j$ can be simplified to Q . The square of the distance between the nodes is being used in this version of the model; such a figure has been traditionally used in these forms of estimates. This leads to equation 2, which forms the basis for evaluation within the network formation models.

$$I_{ij} = k \frac{P_i P_j}{D_{ij}^2} \quad (1)$$

$$I_{ij} = \frac{Q}{D_{ij}^2} \quad (2)$$

There are two different types of links connecting nodes. The initial distance measure between two nodes is the square of the Euclidean distance; the models modify this with the presence or absence of a shortcut links. A shortcut link between two nodes reduces the distance measure to the value to the square of the Euclidean distance multiplied by the shortcut value. Other nodes, aside from those directly connected by a shortcut, may make use of the shortcut; this can reduce the distance between two nodes that are not directly connected by a shortcut. As can be seen in Figure 1, the shortest distance may be less than the Euclidean distance between two points, even when the two points are not on the same subgraph of shortcut links. The normal distance between nodes 3 and 2 is 36 units, as is the distance between nodes 3 and 1. The distance between nodes 1 and 2 is 18 units. A shortcut with a shortcut factor of one-third between nodes 1 and 3 reduces this distance to 12 units, resulting in a total distance of 30 units (12 + 18) between nodes 3 and 2.

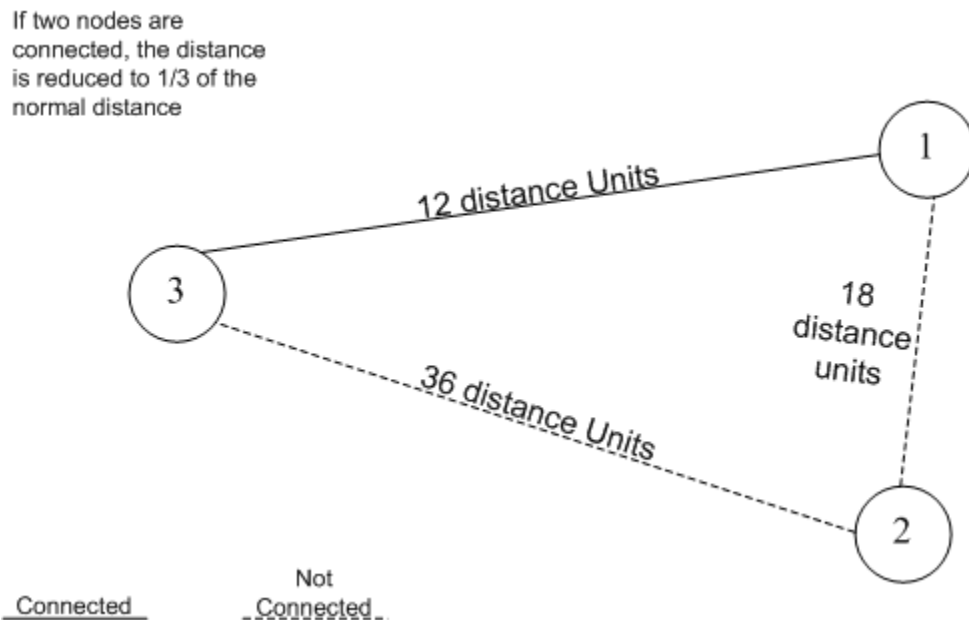


Figure 1: So long as the link between nodes 1 and 3 remains, the shortest path between nodes 3 and 2 is 30 units, via node 1.

In addition to the benefit calculated through the gravity model, a maintenance cost is associated with each network link. This cost is proportional to the Euclidean distance of the network link. In the agent-based model, one-half of the cost of the link is deducted from the total benefit received by each node that the link connects to; thus if a node decides to connect to another node by placing a link, it imposes a cost onto the other node.

The addition of other nodes introduces complexities, in the same manner of the complexities introduced when going from the 2-body to the n-body gravitational problem: the presence of other nodes influences the equation. In the case of this model, the presence of other nodes in close proximity to a target node with a network link can enhance the benefit experienced by the originating node by reducing the distance to the other nodes as well as the target node. Also, the presence of a link connecting two nodes together may inhibit the formation of links to other nearby nodes; the net benefit offered by a direct link may be less than the benefit from the partially reduced distance measure.

Agent-based Implementation

The agent-based model was developed using the RePast modeling software (Collier, 2003). RePast uses the Java programming language, which is designed to be easily portable across a broad spectrum of computers. It supports both raster landscapes and vector spaces, including networks. The specific software developed for this network formation model is called Village; because the model interprets nodes as villages. Appendix A presents the software program elements developed specifically for this model; it includes model setup (villageModel), node behavior and characteristics

(villageNode), and link characteristics and behavior (villageEdge).

Within the agent-based model, each node is able to add or delete a network shortcut from itself to another node. During each turn, each node evaluates which addition or deletion to the network results in the largest increase in benefit. Calculating the benefit generated between the subject node and all other nodes, and then subtracting the cost of network links from the benefit determines the net benefit for the subject node. This leads to the equation 3, which shows the comparison of benefit of node i connecting to node j , given that there are no other nodes; if true, a shortcut link will be formed.

$$\frac{Q}{D_{ij}^2} < \frac{Q}{(sD_{ij})^2} - c \frac{D_{ij}}{2} \quad (3)$$

The net benefit of an individual node is presented in Equation 4. It is the summation of the benefit it receives from all nodes less one half the distance of the shortcuts connecting the node multiplied by the cost factor. S_i is the total distance covered by shortcuts that connect to node i , whether created by node i or by another node.

$$B_i = \left(\sum_{j=1, j \neq i}^n \frac{Q}{D_{ij}^2} \right) - \frac{cS_i}{2} \quad (4)$$

Because of the time constraints of the project, a few small problems remain in the agent-based model; I do not believe any of them will affect the results of the experiment. The most serious of these is the fact that the same random number creator is used for creating the layout and determining the order in which nodes act; this unnecessarily couples the sequence of events with the landscape. This would be a major problem if the results of the model were heavily dependent on early actions; however, nodes within this model can easily create and remove links, so the importance of prior structure is lessened.

The model could also be improved from a visual perspective. Links that are not being used as shortcuts are drawn the same color as the background and all links are drawn over nodes; this causes several nodes, primarily in the center of the landscape, to appear faded or invisible. While this need not be corrected to get data for analysis, it would assist researchers working with the program if they could see all nodes clearly.

Genetic Algorithm Implementation

The genetic algorithm is designed to maximize the total happiness for a given system of nodes. For the processes involving networks, the string that represents a solution is fairly easy to implement; it is a linear representation of the connectivity matrix. For directed networks, this requires $n*(n-1)$ genes, where n is the number of nodes and each gene represents the status of a particular network link. For undirected networks, there are only one-half as many genes. This type of string is suitable for many types of networks problems, with the type of gene varying depending on the type of data representing the link value. For these experiments, a landscape of 30 nodes leads to a string of 435 binary genes, each allele denoting whether or not a shortcut is present. The evaluation function calculates the total cost of the links specified by the string and doubles the benefit experienced by each pair of nodes. The total benefit of the landscape is calculated using equation 5. The total distance covered by shortcuts is multiplied by the cost factor and then subtracted from the summation of the gross benefit experienced by the nodes.

$$B = \left(\sum_{i=0}^{i=n} \sum_{j=0; j \neq i}^{j=n} \frac{Q}{D_{ij}^2} \right) - cS \quad (5)$$

The program using genetic algorithms to calculate the maximum happiness for a system of nodes was written in the C programming language and can be viewed as Appendix B. The genetic algorithm capabilities were obtained through the use of PGAPack, a genetic algorithm library written for C and FORTRAN (Levine, 1996). This library automated many of the functions of setting up and using genetic algorithms, requiring the user only to define the evaluation function. The program reads in the coordinate locations of the nodes from the output generated by the agent-based model, and then attempts to find the maximum total happiness for the system of nodes.

Experimental Design

For this experiment, an initial set of 50 agent-based simulations was run; each simulation utilizes a different landscape. The landscape of nodes for each simulation was created at random from a landscape 430 units wide by 430 units high. Thirty nodes were placed at random upon the landscape. The interaction weight was set to 50,000 interaction units (analogous to population squared); the cost multiplier at 1 interaction unit per distance unit–square shortcut distance unit; and the shortcut value at 0.3 shortcut distance unit per distance unit. These parameters should allow a link between two nodes of less than approximately 115 units long to generate a positive net happiness. The landscape for each simulation was then used as the input for a genetic algorithm using the same parameters. This was used both to gauge the behavior of both the agent-based simulation and the genetic algorithm, and to fine-tune the number of simulations and their length for the second set of simulations

This second set of 50 simulations is used to actually test the differences between the agent-based model and genetic algorithm. These simulations used the same

parameters as the first, pre-trial simulations, and the agent-based model randomly generated each landscape. Paired comparisons of the total benefit the system provides to the nodes will be conducted. Several measures from the agent-based model will be compared to the best results provided by the genetic algorithm; these measures are the total happiness over the last 50 turns, the total happiness over the last 10 turns, the total happiness from the last turn, and the maximum total happiness achieved by the agent-based simulation.

V. Results

Agent-Based Models

The agent-based models behaved in a fairly consistent manner. Figure 2 presents the example of one model run; the y-axis shows measures the total happiness relative to the range of happiness values recorded during the run. There is a steep increase in happiness as the first, most beneficial shortcuts are created; by the tenth turn, the happiness level changed only minimally, and both increased and decreased. This pattern of minor increases and decreases continue until the end of the model's run.

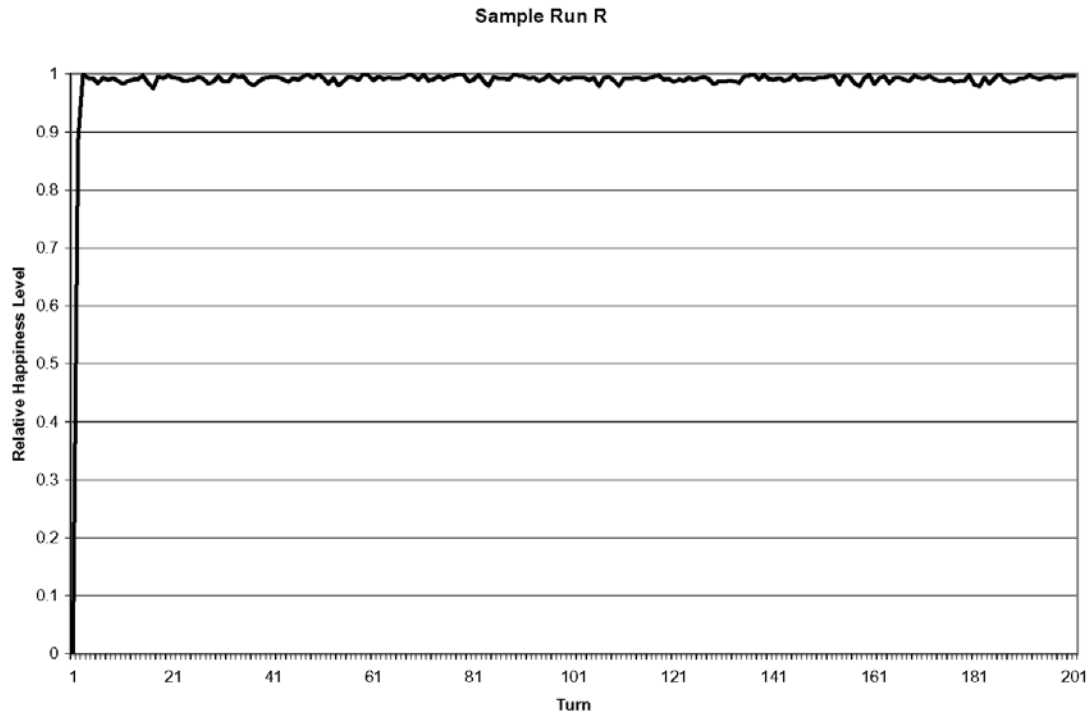


Figure 2: Sample run. The total Happiness is normalized, showing the value as a percent between the minimum and maximum values recorded during a run

This behavior is consistent across all the models, with only the amount of fluctuation changing, as seen in Figure 3. In contrast, the numerical value of total happiness varies over a wide range of values. This is because the maximum amount of happiness is dependent on the collective situation of the nodes. This characteristic is assigned in a random fashion; however, the distribution of values is not normal, though there are enough cases to assume normalness.

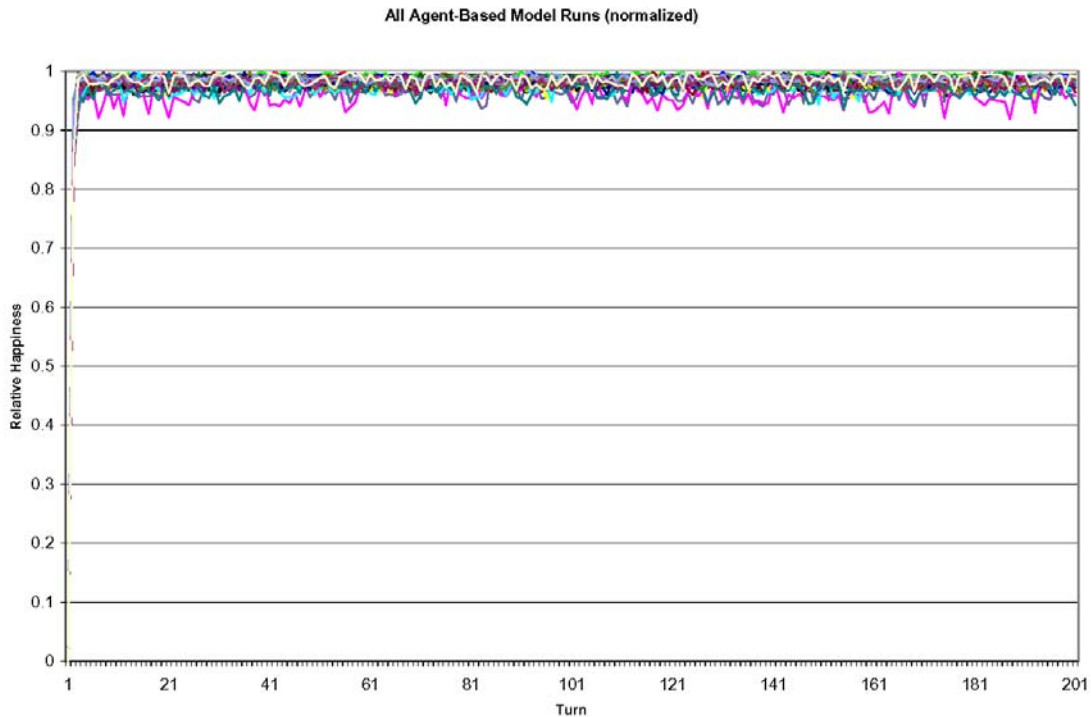


Figure 3: All agent-based runs, normalized. Note the same trend of a rapid initial increase followed by variation over time around a maximum value

Genetic Algorithms

Given that genetic algorithms begin with a set of random solutions to evaluate and from them evolve more fit solutions, the evolution of the results from the genetic algorithm occurs in a different manner. Figure 4 shows the evolution of the solution for the same spatial layout shown in figure 2; figure 5 presents the values of both runs overlaid. The improvement in results reported by the genetic algorithm occurs in a more gradual fashion, as the genetic algorithm starts with an initial solution based on randomly designed solutions. The solutions evolve, rapidly increasing until it approaches the optimum solution, with the algorithm stabilizing with a nearly optimum solution as the genetic algorithm attempts to find a slightly better solution. Indeed, in several of the simulations, the genetic algorithm did not appear to find the optimum solution; the agent-based model found a higher maximum value in thirteen of the 50 spatial layouts. Again,

as the values were dependent upon the spatial layouts, the distribution of values was not normal.

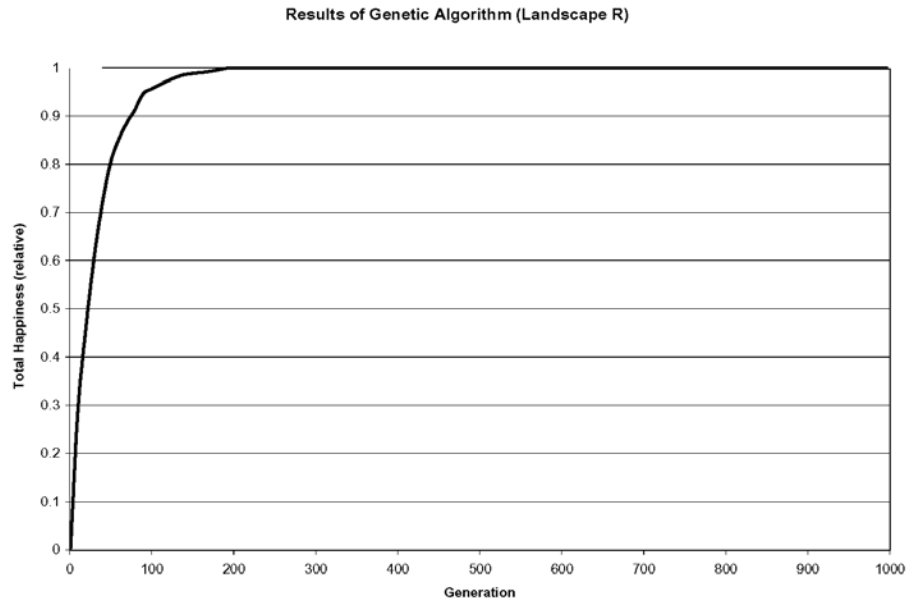


Figure 4: Results of the Genetic Algorithm. The best result found by the genetic algorithm gradually increases, and plateaus at a maximum value.

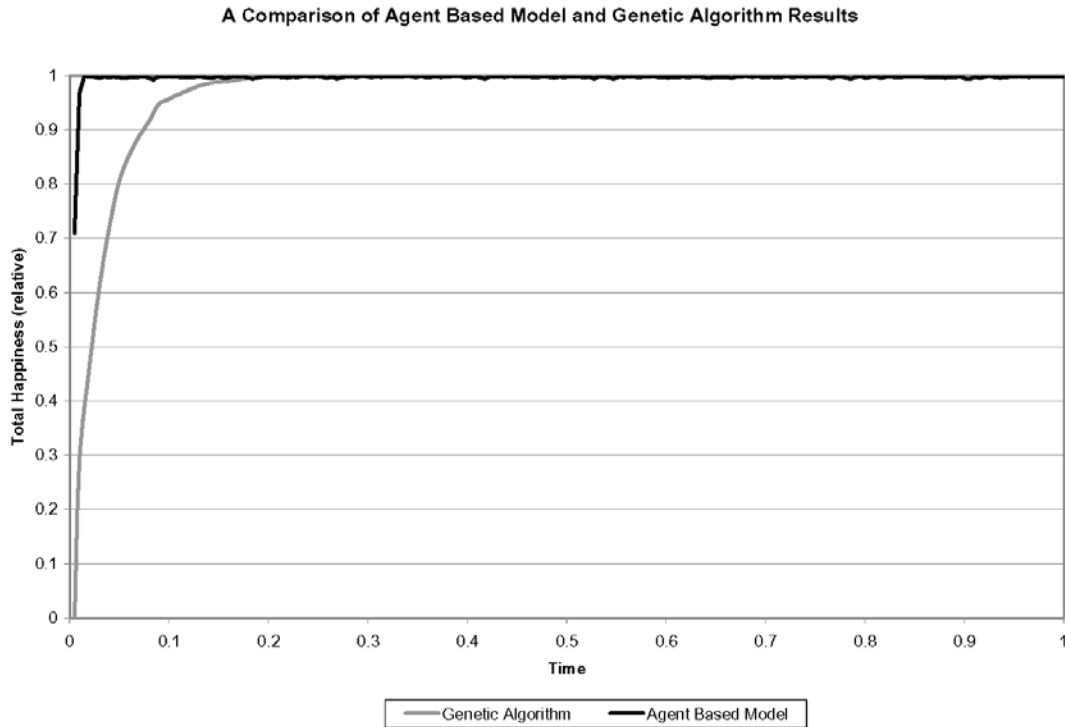


Figure 5: The results of the genetic algorithm and the agent-based model scaled together. The genetic algorithm initially presents inferior solutions; in this example it eventually outperforms the agent-based model, which happens in about 75% of the comparisons.

VI. Analysis

The results of the agent-based model and the results of the genetic algorithm were compared using a paired, two-tailed T-test. Several values from the agent-based model were used in comparison with the genetic algorithm: the maximum value achieved by the agent-based model; the average value of total happiness from the last 50 turns; the average value of total happiness from the last 10 turns, and the last value of total happiness. The only measure from the genetic algorithm was the maximum value found for a given layout. Table 1 presents the results of the T-test. For every comparison, there was no significant difference between the results of the genetic algorithm and the value from the agent-based model.

The distribution of values from both the genetic algorithm and the agent-based

model appeared non-normal. Therefore, a Wilcoxon rank-sum test was also used to evaluate the paired testing (Lehmann, 1975). The results from the rank-sum test are presented in the second column of Table 1. The rank-sum test supports the results of the T-test; for all comparisons, there was not a significant difference between the results of the genetic algorithm and the values from the agent-based model.

Measure	Paired T-test	Wilcoxon rank-sum test
(Total net benefit)	49 dof P-value: ($\mu_{GA} \neq \mu_{ABM}$)	(Probability that the distributions are the same)
Maximum Value achieved by agent-based model	T = 3.8774 P-value = 0.003	92.04%
Average of Last 50 turns of agent-based-model	T=13.2259 P-value = 0	84.96%
Average of Last 10 turns of agent-based model	T=12.9114 P-value = 0	84.96%
Value after last turn of agent-based model	T=11.2483 P-value = 0	84.96%

Table 1: Comparisons of the Results of the Genetic Algorithm and the Agent-Based Model

VII. Discussion

There are several reasons why the agent-based model could have performed as well as the genetic algorithm. Aspects of both the agent-based model and the genetic algorithm probably contributed to this result. The agent-based model provided a large quantity of information to the agents, allowing them to make rational choices. The genetic algorithm did not always find a solution better than that achieved with the agent-based model.

The nodes within the agent-based model were able to make rational, well-informed decisions. Each agent was able to accurately calculate the effect of an action to its benefit. Each agent methodically calculated the potential benefits of every possible

alternative available to it. In the real world, actors are not normally able to explore every option or reliably make decisions with complete rationality. In this respect, the genetic algorithm also is unrealistic; it is unlikely that a single deciding authority would be able to act completely rationally, have all available information, and be unbiased.

Other aspects of the agent-based model also allowed it to perform well. The ease with which network links can be added or removed prevented agents from experiencing penalties associated with bad decisions, and allowed them to quickly correct any bad decisions. As can be seen from the results of both the genetic algorithm and the agent-based model, most of the increase in benefit seems to be caused by a relatively small number of links; both models experienced rapid increases at the beginning of the model, with the remainder of the simulation or search essentially ‘fine-tuning’ the result.

In approximately 25% of the simulations, the genetic algorithm did not find a solution better than the maximum found by the agent-based model. While genetic algorithms are excellent in finding solutions of the same type as an ideal solution, they are not necessarily efficient in finding the best solution for a given problem. An advantage of genetic algorithms is that they join different parts of solutions that perform well together; genetic algorithms can be combined with other algorithms that excel in fine-tuning a preliminary result to find an optimum result. This experiment relied solely on the results from the genetic algorithm. Increasing the number of solutions evaluated per generation or number of generations run would have also increased the likelihood that of the genetic algorithm’s result being equal or greater than the results of the agent-based model, though that would increase the computing time for relatively little gain.

Another interesting result from the agent-based model is the variation in total

happiness over time. This is the result of coordination problems between nodes. As each node attempts to improve its benefit, it alters the benefit of other nodes. This alteration is often detrimental to other nodes; an example of this could be the removal of a long link that connects an outlying node to a node clustered with other nodes. This link provides benefit to the outlying node, but may incur a cost larger than the benefit to the node in the cluster; that node would then subsequently remove the link, improving its benefit while reducing the benefit experienced by the outlying node. These types of problems are common in agent-based models, and have been studied within the context of economics (Mehta, Starmer and Sugden, 1994).

VIII. Extensions

There are several ways to extend the results of this paper. One of the most apparent is to develop and incorporate more sophisticated models of network formation. A more sophisticated network could feature multiple types of shortcuts with such aspects as costs for the creation of network links and different levels of shortcuts. Evaluation methods could incorporate various forms of transfer costs that alter the benefit a node receives from another node depending on how many links lie on the shortest path. Such a system would more accurately reflect the real world, with several different modes competing for use based on the technical and political considerations.

Different ways to assess benefits and costs can result in more interesting and realistic models for use of policy makers. For example, it would be fairly easy to allow several nodes that are geographically close to collectively decide which links to form; this process could more accurately reflect the way transportation projects are decided in the real world, where generally a group of parties cooperate for a set of transportation

projects from which all the participants would receive benefit. This would be especially beneficial as a policy tool by the addition of loading pre-specified landscapes of nodes derived from the real world.

Research of how a pre-existing structure influences network development is another avenue for development. In the agent-based simulation, there was no cost above the change in benefit associated with the removal of a node. In the real world, the demolition of such a link can be a costly endeavor. Additionally, many networks form directly over or closely aligned to existing links of a prior network. This can be seen especially in the eastern United States, where several of the major regional highways originated as foot trails.

To explore more dynamic relationships, transportation formation models can be tied with models featuring mobile agents. By incorporating multiple types of dynamic models, more realistic models of processes such as migration and trade where recursive relationships develop between a resource and the groups that use a resource. By simply combining different types of existing models, these more complex models can be rapidly created and studied.

Finally, the behavior of the agents can be further examined. Agent-based models often feature the development of coordination problems. The effects of the problem can be seen in the simulations run as a slight variance in the total happiness after the network has largely stabilized. Economists are studying this area, but the role of geography in the development of coordination problems and the implications in real world systems has not been examined (Van Hyuck, Battalio and Biel, 1990; Lagunoff and Matsui, 1997). A transportation network model could for an ideal platform to investigate coordination

problems because, as in the model used in this paper, the only differing characteristic among otherwise homogenous nodes is their location.

IX. Conclusion

Networks play an important role in the geography of the world. The distance or cost across a network is often the spatial separation for economic activities, as it is only through networks that most of our transportation and communication technologies operate. An efficiently designed network allows goods and ideas to flow efficiently, encouraging development. Conversely, a poorly designed network can hinder a country's economy, not only by interfering with the movement of goods, but also by incurring unnecessary maintenance costs on the region's economy.

This paper presented the results of comparing an agent-based model of network formation with a genetic algorithm designed to find an optimum network configuration. These tools act in very different ways. The agent-based model emulates the process found within free market economics, a bottom-up approach in which the recipients of benefit attempt to construct the system that would provide them the most benefit. The genetic algorithm, in contrast, acts like a top-down central planning authority imposing a configuration upon a landscape. Both types of tools performed similarly, in accordance to the classic economic theory of perfectly competitive markets. With added complexity to the agent-based model, it is likely that results would diverge from the results of the genetic algorithm; it would become harder for agents, acting in their own interest, to be able to achieve solutions that provide maximum benefit to the entire landscape of agents.

The agent-based model of network formation presented in this paper can form a

basis for further studies in economic geography. Currently, the model has been developed to a very basic level; two types of networks exist, the evaluation function relies on a simplified gravity model, and links can be added or removed by a node with ease. By further developing the model to incorporate different types of networks, more complex interaction functions, and more complex methods of accounting for the costs of links, this model can be of use to transportation planners and other people studying regional development.

References

- Axtell, R.L., Epstein, J.M., Dean, J.S., Gumerman, G.J, Swedlund, A.C., Harburger, J., Hammond, R., Parker, J. & Parker, M. (2002). Population growth and collapse in a multiagent model of the Kayenta Anasazi in Long House Valley. *Proceedings of the National Academy of Science*, 99, 7275-7279
- Bala, V. and Goyal, S. (2000). A noncooperative model of network formation. *Econometrica*, 5, 1181-1229.
- Bielli, M., Caramia, M. and Carotenuto, P. (2002). Genetic algorithms in bus network optimization. *Transportation Research Part C: Emerging Technologies*, 10, 19-34.
- Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems*. New York: Oxford University Press.
- Collier, N. (2003). *RePast: An Extensible Framework for Agent Simulation*. [Online]. Available: http://repast.sourceforge.net/docs/repast_intro_final.doc [15 Feb., 2003].
- DeCanio, S.J., Dibble, C. and Amir-Atefi, K. (2000). The importance of organizational structure for the adoption of innovations. *Management Science*, 46, 1285-1299.
- Deeter, D. and Smith, A. (1998). Economic design of reliable networks. *IIE Transactions* 30, 1161-1174.
- Dibble, C. and Densham, P. (1993). Generating interesting alternatives in GIS and SDSS using genetic algorithms. *GIS/LIS Proceedings*.
- Dibble, C. (2001). *Theory in a Complex World: GeoGraph Computational Laboratories*. Doctoral Dissertation, University of California, Santa Barbara, 2001.

- Drezner, Z. and Wesolowsky, G.O. (2003). Network design: Selection and design of links and facility location. *Transportation Research Part A: Policy and Practice* 37, 241-256.
- Epstein, J.M. (2002). Modeling civil violence: An agent-based computational approach. *Proceedings of the National Academy of Science*, 99. pp. 7243-7250.
- Eubank, S. (2002). Scalable, efficient epidemiological simulation. In *Proceedings of the 2002 ACM Symposium on Applied Computing*, 139-145. New York: ACM Press.
- Fischer, M.M., Leung, Y. (1998). A genetic-algorithm based evolutionary computational neural network for modeling spatial interaction data. *Annals of Regional Science*, 32, 437-458.
- Fwa, T.F., Chan, W.T. and Sim, Y.P. (2002). Optimal vertical alignment analysis for highway design. *Journal of Transportation Engineering* 128(5), 395-402.
- Gardner, M. (1970). The fantastic combinations of John Conway's new solitaire game "life". *Scientific American*, 223(4): 120-123.
- Graham, B. (1995). *Geography and Air Transport*. New York: Wiley.
- Holland, J. (1992). *Adaptation in natural and artificial systems: An introductory analysis with applications to Biology, Control, and Artificial Intelligence*. (1st MIT press edition). Cambridge, MA: The MIT Press.
- Jackson, M.O. and Watts, A. (2002). The evolution of social and economic networks. *Journal of Economic Theory*, 16, 265-295.
- Lagunoff, R. and Matsui, A. (1997). Asynchronous choice in repeated coordination games. *Econometrica*, 65. pp. 1467-1477.

- Langton, C.G. (1998). Artificial life. In Langton, C. (ed.) *Artificial Life, SFI Studies in the Sciences on Complexity*, pp. 1-47. Reading, MA: Addison-Wesley Publishing Company.
- Lehmann, Erich (1975). *Nonparametrics: Statistical Methods based on Ranks*. Holden-Day: San Francisco.
- Levine, D. (1996). *Users guide to the PGAPack parallel genetic algorithm library*. Technical Report ANL-95/18. Argonne, IL: Argonne National Laboratory.
- Lyons, G., Chatterjee, K., and Marsden, G. (1999). Transport visions: A young professionals' perspective. *Traffic Engineering and Control*, 40, 578-581
- Mehta, J., Starmer, C. and Sugden, R. (1994). The nature of salience: An experimental investigation of pure coordination games. *The American Economic Review*, 84. pp. 658-673.
- McFadzean, D., Stewart, D., and Tesfatsion, L. (2001). A computational laboratory for evolutionary trade networks. *IEEE Transactions on Evolutionary Computation*, 5, 546-560.
- O'Sullivan, D. Graph-based cellular automata: a generalized discrete urban and regional model. *Environment and Planning B: Planning & Design*, 28, 687-705.
- Peeters, D., Thisse, J.-F., and Thomas, I. (1998). Transportation networks and the location of human activities. *Geographical Analysis*, 30, 355-373.
- Qu, Liangsheng and Sun, RuiXiang (1998). A synergetic approach to genetic algorithms for solving traveling salesman problem. *Information Sciences*, 117, 267-283.
- Reynolds, C. W. (1987) Flocks, herds, and schools: a distributed behavioral model. *Computer Graphics*, 21(4) (SIGGRAPH '87 Conference Proceedings) pages 25-

- 34.
- Rioja, F. (2003). The penalties of inefficient infrastructure. *Review of Development Economics*, 7, 127-137.
- Savage, M., Sawhill, B., and Askenazi, M. (2000). What happens when we rerun the tape? *Journal of Theoretical Biology*, 205, 515-526.
- Shen, Q. (2001). A spatial analysis of job openings and access in a US metropolitan area. *Journal of the American Planning Association*, 67(1), 53-68.
- Sheppard, Eric. (1995). Modeling and predicting aggregate flows. In Hanson, S. (ed.) *The Geography of Urban Transportation* (2nd ed.). New York: The Guilford Press. pp. 100-128.
- Tutschku, K. and Tran-Gia, P. (1998). Spatial traffic estimation and characterization for mobile communication network design. *IEEE Journal of Selected Areas in Communications*, 16, 804-811.
- Van Hyuck, J.B., Battalio, R.C., and Biel, R.O. (1990). Tacit coordination games, strategic uncertainty, and coordination failure. *The American Economic Review*, 80. pp. 234-248
- Watts, A. (2001). A dynamic model of network formation. *Games and Economic Behavior*, 34, 331-341.

Appendix A

Source code of the Agent-Based Model

The Agent-based model was written in Java and designed to work with the RePast 1.4.1 modeling libraries. The Apache Ant build system was used to compile the code. More information on RePast, including documentation, the libraries and sample models can be found at <http://repast.sourceforge.net>. Information on the Apache Ant system can be found at <http://ant.apache.org>.

In this paper, only the code for the activities related to the decision-making and activities of the agents is presented.

villageNode.java:

```
public void AddBest() {
    int h = this.getNodeNumber();

    //Get the current Adjacency and Shortest path matrix\
    //An entry in the Adjacency Matrix can have 3 values:
    // 1- normal euclidean distance
    // 0- no connection (in this case, used only for a node
    // connecting to itself.
    // edge_strength- the value of the shortcut factor.
    DenseDoubleMatrix2D exist_mat = getAdjacency_mat();
    DenseDoubleMatrix2D short_path = CalcShortPath(exist_mat);

    //Get current happiness
    happiness = EvalMatrix(short_path, h, inter_val);
    best_happiness = happiness;
    ArrayList outEdges = getOutEdges();
    int edge_num = outEdges.size();
    ArrayList candidates = new ArrayList (edge_num + 1);
    candidates.add(this);

    //Evaluate the links to all other nodes.
    for (int l =0; l < edge_num; l++ )
    {
        VillageEdge edge = (VillageEdge)outEdges.get(l);
        VillageNode oNode = (VillageNode)edge.getTo();
        int o = oNode.getNodeNumber();
        double test_link = exist_mat.get(h, o);
        if (test_link == 1)
        {
            exist_mat.set(h, o, this.edge_strength);
            exist_mat.set(o, h, this.edge_strength);
        } else {
```

```

    exist_mat.set(h, o, 1);
    exist_mat.set(o, h, 1);
}
DenseDoubleMatrix2D test_short = CalcShortPath(exist_mat);
double test_happiness = EvalMatrix(test_short, h, inter_val);
exist_mat.set(h, o, test_link);
exist_mat.set(o, h, test_link);

//If it's better, erase the previous candidates and add
//If it's as good, add to the list.
if (test_happiness > best_happiness)
{
    best_happiness = test_happiness;
    candidates.clear();
    candidates.add(oNode);
} else if (test_happiness == best_happiness)
{
    candidates.add(oNode);
}
}

//If candidates perform better then the current happiness, select one
//of the candidates and perform the flip
//Otherwise, do nothing

candidates.trimToSize();
int size = candidates.size();

    if (happiness == best_happiness)
    {
        //Do Nothing
    } else if (size == 1)
    {
VillageNode cNode = (VillageNode)candidates.get(0);
int clabel = cNode.getNodeNumber();

//I used hashsets as a clunky way to get to the edges.
//There should only be one edge in each hashset
HashSet changed_edges = this.getEdgesTo(cNode);
HashSet other_changed_edges = cNode.getEdgesTo(this);

//Change the Edge strength value for the edges
if (!changed_edges.isEmpty())
{
    Iterator edges1 = changed_edges.iterator();
    VillageEdge changedEdge = (VillageEdge)edges1.next();
    changedEdge.flipEdge(this.edge_strength);
}

if (!other_changed_edges.isEmpty())
{
    Iterator edges2 = other_changed_edges.iterator();
    VillageEdge otherChangedEdge = (VillageEdge)edges2.next();
    otherChangedEdge.flipEdge(this.edge_strength);
}

//Update the node's happiness and the adjacency matrix

```

```

this.happiness = best_happiness;
double test_link = exist_mat.get(h, clabel);
if (test_link == 1.0)
{
    exist_mat.set(h, clabel, this.edge_strength);
    exist_mat.set(clabel, h, this.edge_strength);
} else {
    exist_mat.set(h, clabel, 1.0);
    exist_mat.set(clabel, h, 1.0);
}
} else {
int change_list = Random.uniform.nextIntFromTo(0, size - 1);
VillageNode cNode = (VillageNode)candidates.get(change_list);
int clabel= cNode.getNodeNumber();
HashSet changed_edges = this.getEdgesTo(cNode);
HashSet other_changed_edges = this.getEdgesFrom(cNode);
if (!changed_edges.isEmpty())
{
    Iterator edges1 = changed_edges.iterator();
    VillageEdge changedEdge = (VillageEdge)edges1.next();
    changedEdge.flipEdge(this.edge_strength);
}

if (!other_changed_edges.isEmpty())
{
    Iterator edges2 = other_changed_edges.iterator();
    VillageEdge otherChangedEdge = (VillageEdge)edges2.next();
    otherChangedEdge.flipEdge(this.edge_strength);
}
this.happiness = best_happiness;
double test_link = exist_mat.get(h, clabel);
if (test_link == 1.0)
{
    exist_mat.set(h, clabel, this.edge_strength);
    exist_mat.set(clabel, h, this.edge_strength);
} else {
    exist_mat.set(h, clabel, 1.0);
    exist_mat.set(clabel, h, 1.0);
}
}
}

//This evaluates the net benefit experienced by node i
//in a given network configuration
private double EvalMatrix (DenseDoubleMatrix2D short_mat, int i,
                           double inter_val) {
DenseDoubleMatrix2D crow_mat = getCrow_mat() ;
double cost_dist = 0;
double temp_happiness = 0;
double total_happiness = 0;
for (int j =0; j< crow_mat.columns(); j++) {
    if (i != j) {
        if (short_mat.get(i, j) == this.edge_strength * crow_mat.get(i, j))
        {
            cost_dist = cost_dist + this.cost * 0.5 * crow_mat.get(i, j);
        }
    }
}
}

```

```
    }  
  }  
  for (int j =0; j< crow_mat.columns(); j++) {  
    temp_happiness = 0;  
    if (i != j){  
      temp_happiness= inter_val /(short_mat.get(i,j)*short_mat.get(i,j));  
    }  
    total_happiness = total_happiness + temp_happiness;  
  }  
  total_happiness = total_happiness - cost_dist;  
  return total_happiness;  
}
```

villageEdge.java:

```
//Change the edge strength (and color)  
public void flipEdge(double other_strength) {  
  if (this.getStrength() == 1.0)  
  {  
    this.setStrength(other_strength);  
    this.setColor(Color.black);  
  } else {  
    this.setStrength(1.0);  
    this.setColor(Color.white);  
  }  
}
```

Appendix B

Source Code for Genetic Algorithm

The genetic algorithm was developed using the PGAPack library, available for either C or FORTRAN. The C version was used. The program was compiled with gcc version 2.95 and run on a Sun Ultra 10 using Solaris 7. More information on PGAPack, including the libraries and user's manual, can be obtained at: http://www-fp.mcs.anl.gov/CCST/research/reports_pre1998/comp_bio/stalk/pgapack.html.

Only the main and evaluate functions are presented here.

net-finder.c:

```
#include <stdio.h>
#include <math.h>
#include <pgapack.h>

#define MAX_NODES 50
#define POOL_SIZE 100

int readFile(char *filename);
void makeCrowMatrix(int numnodes);
double evaluate(PGAContext *ctx, int p, int pop);

float node_coords[MAX_NODES][2];
double crow_matrix[MAX_NODES][MAX_NODES];
int numnodes;

float inter_val;
float short_value;
float cost_dist;

int main(int argc, char *argv[])
{
    int i, j, length;
    PGAContext *ctx;

    double temp_happiness = 0;
    double total_happiness = 0;

    short_value = 0.3;
    inter_val = 50000;
    cost_dist = 1;

    /* read in the file, make the Crow Matrix */
    numnodes = readFile(argv[1]);
    makeCrowMatrix(numnodes);
    printf("numnodes= %i\n\n", numnodes);

    /* calculate the string length (less than 0.5 of a matrix) */
```

```

length = 0;
for (i = 1; i < numnodes ; i++){
    length = i + length;
}

/* Set up the GA */
ctx = PGACreate (&argc, argv, PGA_DATATYPE_BINARY, length,
PGA_MAXIMIZE);
PGASetPopSize          (ctx, POOL_SIZE);
PGASetCrossoverType   (ctx, PGA_CROSSOVER_UNIFORM);
PGASetNumReplaceValue (ctx, POOL_SIZE - 10);
PGASetMutationAndCrossoverFlag (ctx, PGA_TRUE);
PGASetMaxGAIterValue  (ctx, 1000);

PGASetUp(ctx);

/* Run the GA */
PGARun(ctx, evaluate);

/* Results are reported as part of PGARun */

/* Stop running the GA */
PGADestroy(ctx);

return 0;
}

/* The evaluate function takes in the string, converts it to an
adjacency
* matrix, creates a shortest path matrix, and then calculates benefit
* recieved by all nodes and the cost of the shortcut links. The net
* benefit (benefit of all nodes - shortcut cost) is returned.
*/
double evaluate(PGAContext *ctx, int p, int pop)
{
    int i, j, k, m, n, nbits, stringlen;
    int adj_mat[MAX_NODES][MAX_NODES];
    float distance[MAX_NODES][MAX_NODES];
    double current, altpath;
    int bit;

    double temp_happiness=0;
    double total_happiness=0;
    double end_happiness = 0;
    double temp_cost = 0;
    double total_cost = 0;
    stringlen = PGAGetStringLength(ctx);

    /*Make Adjacency Matrix*/
    k = 0;

    for (i = 0; i < numnodes ; i++ ) {
        for (j = 0; j <= i; j++) {
            if (i == j) {
                adj_mat[i][j] = 0;
            } else {
                bit = PGAGetBinaryAllele(ctx, p, pop, k);

```

```

        adj_mat[i][j] = bit;
        adj_mat[j][i] = bit;
        k++;
    }
}

/* Make the technology modified distance matrix */
for (i = 0; i < numnodes ; i++) {
    for (j = 0; j < numnodes ; j++) {
        if (i == j) {
            distance[i][j] = 0;
        } else {
            distance[i][j] = adj_mat[i][j] == 1 ? short_value *
crow_matrix[i][j] :
crow_matrix[i][j];
        }
    }
}

/* Calculate the cost of the shortcut links */
for (m = 0; m < numnodes; m++) {
    for (n = 0; n < m; n++) {
        temp_cost = adj_mat[m][n] == 1 ? crow_matrix[m][n] : 0;
        total_cost = total_cost + temp_cost;
    }
}

/* now apply the Floyd-Warshall algorithm to build all shortest paths*/
i=0; j=0; k=0;
for (k=0; k < numnodes; k++)
    for (i=0; i < numnodes; i++)
        for (j=0; j < numnodes; j++) {
            current = distance[i][j] ;
            altpath = distance[i][k] + distance[k][j] ;
            distance[i][j] = (current <= altpath) ? current : altpath ;
        }

/* Now onto calculating happiness
   * We got the costs while we were filling out the distance matrix
   * Total Benefit is:
*/
for (i = 0; i < numnodes ; i++ ) {
    for (j = 0; j < numnodes; j++ ) {
        if (i != j) {
            temp_happiness = 0;
            temp_happiness = ( inter_val / (distance[i][j] * distance[i][j]) );
            total_happiness = total_happiness + temp_happiness;
        }
    }
}
end_happiness = total_happiness - total_cost;
return end_happiness;
}

```