

# Influencing Massive Multi-agent Systems via Viral Trait Spreading

Brian McLaughlan and Henry Hexmoor<sup>1</sup>

Department of Computer Science  
Southern Illinois University  
Carbondale, IL 62901  
brian@mclaughlandigital.com, hexmoor@cs.siu.edu

**Abstract**-This paper describes a method by which a massive multi-agent system can be influenced without resorting to micromanagement. This method could be utilized in the development of meta-reasoning components of individual agents. Agents in the system adopt the traits of their successful peers. The administrator guides this spread of traits through selectively injecting influential agents with modified traits. These key agents are identified via social network analysis techniques. Experimentation is described in which the system is tested for its ability to automatically adopt an acceptable configuration as well as testing the ease in which the administrator is able to guide the system to a better configuration.

## I. INTRODUCTION

Recent advances in miniaturization and computational power have led to increased interest in fields such as sensor networks and data fusion. However, the rapid pace in which operating environments change requires a system that can quickly and easily adapt to these challenges. Additionally, the potential scale of these systems precludes any form of micromanagement. As a further complication, many organizations prefer to maintain some nominal control over the agents within the system, disqualifying completely automated systems.

The Man-On-The-Loop (or MOTL) paradigm originated with the objective of developing computational protocols and methodologies for placing a human operator in charge of a massive and complex community of agent-based systems [1]. In such a system, the human administrator is removed from being a bottleneck *in* the loop and is instead placed *on* the loop, functioning as an observer that is able to assert control when necessary. These massive communities are useful for examining military as well as civilian applications that require simulation of large communities [2].

This paper describes a system framework compatible with MOTL goals that allows a single administrator to influence and guide the development of a massive network of intelligent agents. The framework is modeled on the intuitive understanding of survival of the fittest and viral spread. In summary, agents adopt the traits of their successful peers or can be artificially adjusted by the administrator.

The process of issuing commands for agent-based systems is not new. Two generic methods are commonly associated with human control of systems: brute force and policy-making.

Both of these methods will be quickly examined to determine where improvement is needed.

In a brute force-based system, we consider agents to possess predefined values for traits. These default values can be treated as global values uniformly used by all agents. This method has obvious benefits for the massive organization requirement.

Obviously, this approach does not scale well to complex systems. If all agents take the same values, then it is unlikely that all agents would approach optimal configuration. If groups of agents can be targeted for adjustment, this brings new difficulty. In particular, it is likely that the administrator would not be able to accurately determine which agents should be included in the adjustment group. Once that hurdle is overcome, it would be time consuming for the administrator to specify these agents that should be part of the group and would approach micromanagement.

Unlike brute force methods, a policy-based approach offers flexibility and an easy to extend evolutionary management solution that is applied in several disciplines, most notably in network management [3]. Policies are rules and conventions governing the choice in the behavior of the element that is managed. Policies can be used for controlling of autonomic systems or the guidance for decision-making.

Policies are disabled, updated, modified, or entirely substituted by new ones. A key property of a policy based management system is that changes to targets should be performed in a consistent fashion in order to avoid policy conflicts. This creates a need for policy revision and maintenance that is performed offline. This process is tedious and slow. This is a major shortcoming that we chose to avoid.

Network policies are typically stored in a repository. This repository can be used in an enforcement mechanism, proactively seeking and checking compliance in a “push” model of enforcement, but this method has an impractical number of enforcement points in massive systems. Alternatively, policies might be disseminated to subscribers via a “pull” model, but this would result in inconsistent subscriptions.

## II. APPROACH

In the methodology we are labeling Viral Trait Spreading, we specify components within the individual agents as well as an interface to those agents by the administrator. The methodology consists of three main components: Trait Adoption, Candidate

---

<sup>1</sup>This research is sponsored by the Air Force Research Laboratory (AFRL) under contract FA8750-06-C-0138.

Identification, and Trait Injection. Although all the components are executing concurrently, it is helpful to visualize the system as shown in Figure 1.

Several assumptions are made concerning the agents in the system. First, the agents are assumed to be heterogeneous. Second, the assumption is made that the agents will interact with each other. Finally, with all other things being equal, an agent that is more successful than another agent at some task is more likely to be better configured to accomplish it.

#### A. Framework Core: Trait Adoption and Injection

The core of the framework consists of the Trait Adoption and Trait Injection components. Thus, these components are discussed together. The primary external influence on the system is through agent enhancement via the Trait Injection component. When enhancing an agent, we say that *the agent administrator is injecting a trait that is tied to some context*.

**Definition 1:** *The agent administrator* is defined as the overseer of the multi-agent system. The administrator is not an end user and is generally not concerned with the specific tasks of the agents. However, the administrator is expected to be familiar with the overall goals of the system. The administrator is tasked with optimizing the behavior and capabilities of the agents as a group.

**Definition 2:** A *trait* is a variable characteristic or ability of an agent that can be parameterized or compartmentalized. Traits can be expressed in virtually any form, and can represent a wide variety of concepts such as policies, procedures, data, configurations, goals, or even complete methodologies.

**Definition 3:** A *context* is defined as the setting in which an interaction between agents occurs. A context can include physical location, commonality of task or situation, or shared personal traits.

**Definition 4:** An *injection* is a process by which the agent administrator introduces one or more viral traits to an agent. The injection includes a context to contain the spread of the carried trait to appropriate candidates.

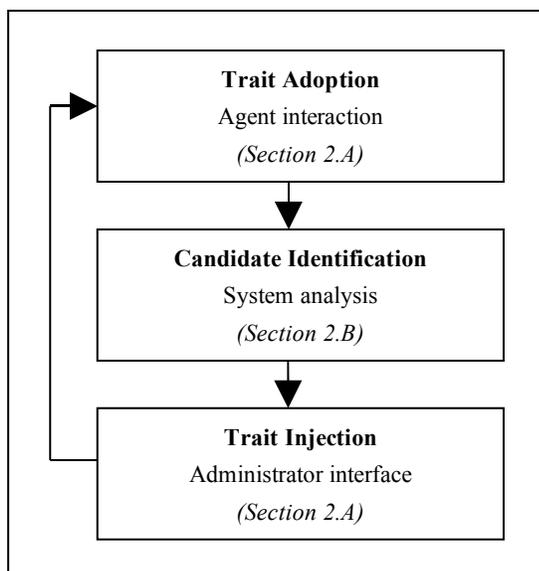


Fig. 1. Framework components

Traits and contexts can be as simple or as complex as the system goals demand or as the capabilities of the agents limit. In a simple distributed sensor network, one trait may be “message transmission timing” while the context may be a single universal state. For systems in which the agents are able to perform complex reasoning about their orientation and situation, context can be a subtly layered and powerful system if needed

Agent interaction occurs in the Trait Adoption component. When agents interact within some context, the more successful agent will pass its appropriate traits to the less successful agent. The greater the relative success, the more strongly the trait is adopted or the more likely the trait will be adopted. The exact method of adoption can be probabilistic or fuzzy.

For a simple example of the Trait Adoption and Trait Injection components in action, consider a system of agents that must interact with each other and various intelligence agency sources around the world to locate various types of weapons of mass destruction. In a complex system such as this, the exact best strategy may not be known for every situation. Generalized rules could exist. Perhaps Procedure X is the best strategy to utilize in Asia when locating a particular type of WMD, while Procedure Y is best in the Middle East when locating that same type. It would therefore be reasonable to expect the agent administrator to be able to identify some agent that should utilize Procedure X or Procedure Y. However, the boundary between Asia and the Middle East is vague, leaving agents that are working in border regions difficult to categorize. Thus, requiring the administrator to explicitly determine which agents should operate using Procedure X or Procedure Y is not reasonable.

Suppose the agent administrator knows that Procedure X is the best strategy in Asia, but observes that the agents are utilizing less effective methods. The administrator can locate an agent that is clearly operating in Asia and inject “Use procedure X when locating that particular WMD” into this agent. “Use Procedure X” would be the trait with “When locating that particular WMD” as the context. This agent would then become more successful at locating these WMDs than its peers and would pass the trait to those with which it interacts. Eventually, the trait would pass to regions of the world that are harder to define as being part of Asia, as shown in Figure 2. In such a case, the trait would continue to transmit as long as Procedure X is still more valid than the procedure in use by the local agents. Eventually, the “Use Procedure X” trait may spread near enough to the Middle East that it is not as effective as Procedure Y. When this happens, agents that utilize Procedure Y will be more successful than those using Procedure X, and an equilibrium boundary will emerge as the Procedure X trait ceases to spread. This precludes the need for the administrator to exactly define where Procedures X and Y are most useful.

Now suppose the administrator observes that some procedure is being successfully utilized in Europe and has spread to agents throughout the region as the best known strategy. However, the administrator also observes some flaws in this strategy and can see how to make the strategy even better. By tweaking the trait of some influential agent and injecting the modified trait back into the agent, the administrator can guide the agents to achieve more success. In this case, the current traits of successful agents

suggest to the administrator an even better trait. In these scenarios, the agent administrator's primary function is to utilize his intuition and expertise to overcome local maxima.

*B. Network Analysis: Automated Candidate Identification*

The job of the Candidate Identification component is to determine which agents are most influential to the system and are therefore ideal candidates for modification via injection. By focusing on the modification of these agents, the administrator can limit the time spent managing traits. To accomplish this goal, the component utilizes social network analysis (SNA) techniques. These techniques include formulas for determining agents that are *prominent* [4] or *central* [5] in the network due to degree, closeness, betweenness, and amount of information that might pass through the agent.

*Degree centrality* is the simplest in both concept and equation [6]. It simply measures the number of connections that an agent possesses and assumes that an agent with many direct connections to other agents is more prominent in the network than an agent with few connections. In general, the agent-level degree centrality index  $C_D(n_i)$  is defined as

$$C_D(n_i) = d(n_i) = \sum x_{ij} = \sum x_{ji} \tag{1}$$

where  $x_{ij}$  is the value of the tie between agents  $i$  and  $j$ . For this experiment, no connection is more important than another, so the values of all ties are identical. Thus, the formula reduces to simple counting.

Another centrality concept explored is *closeness*, or how near an agent is to all other agents in the network [7][8]. The formula used to calculate this index is

$$C_c(n_i) = [\sum d(n_i, n_j)]^{-1} \tag{2}$$

where  $d(n_i, n_j)$  is the number of connections in the geodesic linking agents  $i$  and  $j$ .

*Betweenness* is the third index examined [9]. This index measures the number of other agents that utilize the particular agent as an intermediary for communicating with others. This index is quantified for an agent by counting the number of geodesics on which it lies and is defined as

$$C_b(n_i) = [\sum g_{jk}(n_i)] g_{jk}^{-1} \tag{3}$$

for  $i$  distinct from  $j$  and  $k$ , with  $j < k$ , and where  $g_{jk}$  is the number of geodesics linking agents  $j$  and  $k$  and where  $g_{jk}(n_i)$  is the number of geodesics linking those two agents that contain agent  $i$ .

*Information centrality*, the fourth index examined, is a modification of the betweenness concept [10]. Where betweenness assumes a message between two agents will always take the shortest path, information centrality gives some probability to longer routes. Thus, the weight given to a particular path is inversely proportional to the length of the path. The equation for this index is defined as

$$C_i(n_i) = [c_i + (T - 2R) / g]^{-1} \tag{4}$$

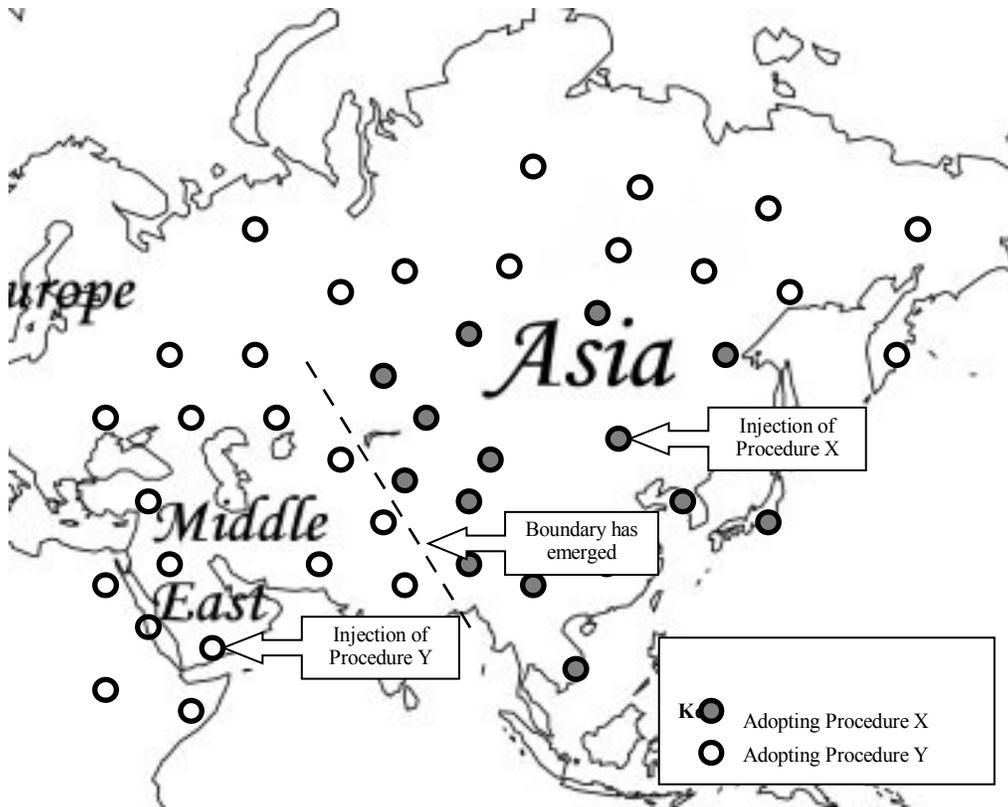


Fig. 2. Results of injecting identified agents

where  $C$  is an inverted adjacency matrix,  $T$  is the sum of diagonal entries of  $C$ ,  $R$  is the sum of any row of  $C$ , and  $g$  is the number of geodesics.

Using the equations described above, the administrator can view a mathematical representation of the agents under his control to determine the correct course of action. The most influential agent or group of agents can be identified, thus reducing the number agents that must be directly modified. Experimentation was used to determine which of these formulas provide useful information.

### III. SIMULATION

To investigate the viability of the proposed framework, a group of related simulation testbeds was developed. The simulations explore three basic agent structures – a grid-like network with each agent connected to its four neighbors, a scale-free network with a minority of agents having the majority of connections, and a randomly distributed network. The links between agent nodes represent the communication links between friendly agents. These types of structures are representative of the types of networks found in natural and artificial systems. Although each network may not be useful in every situation, the exploration of each demonstrates the system’s ability to handle a wide gamut of situations. Each network was populated with 2500 individual agents to get a good feel for the system’s ability to handle large populations. 2500 agents was found to be the largest size population the test computer could safely handle without running out of memory.

The trait examined in this simulation is *trust inertia*. For this experiment, the concept of trust assumes agents are benevolent and does not involve information security [11]. Rather, it is a measure of the perceived reliability of information from a particular agent. Inertia, in general science terms, is the resistance an object has to a change in its state of motion. Combined, trust inertia is a parameter that describes an agent’s resistance to changing its trust in another agent in light of successes or failures. For instance, if a peer agent is caught in a mistake, an agent with low trust inertia will be quick to discount the peer’s future accomplishments as unreliable while an agent with high trust inertia will be more forgiving of the mistake.

#### A. Trait Adoption

To examine the framework’s ability to distribute ideal traits, agents within the network were given random traits and competence levels while the simulation recorded agent configurations and success rates. During each time iteration, every agent would complete a task. An agent was probabilistically successful at the task based on its competence level. After completing the task, the agent reported the results of its task to each of its neighbors (*e.g.*, each other agent that is connected to the agent). This simulated the sharing of information with its peers. These neighboring agents would then utilize the information if they trusted the broadcasting agent or would discard the information if they distrusted it. If the agent successfully accomplished the task and broadcasted good information, the neighboring agents would increase their trust in the agent based on their individual trust inertia levels. Agents with higher trust inertia would be less likely to increase their trust than agents with low trust inertia. Similarly, faulty information would result in a decrease in trust, with agents

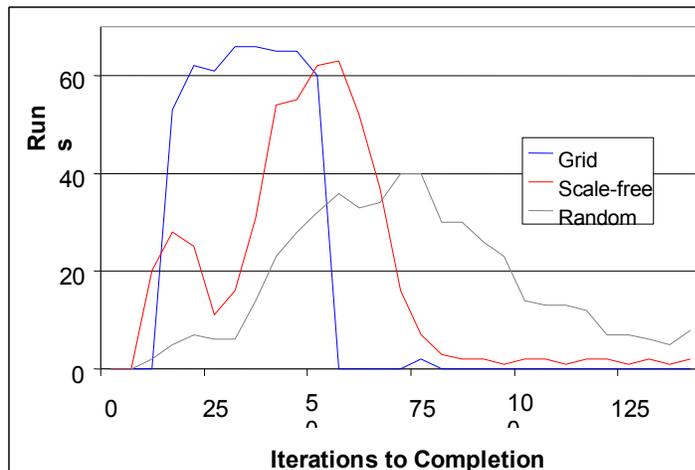


Fig. 3. Range of stabilization time

having low trust inertia being more likely to lower their trust than agents with high trust inertia. An agent’s success score was based on the amount of good information that it incorporated from its neighbors and was punished by the amount of bad information it utilized.

To make the experiment more interesting, different regions of the network were given different ideal trust inertia levels. Thus, agents in some regions would benefit from having a more stable trust while others would do better with more volatile trust. To simulate these regions, reward and punishment modifiers were added to areas of the network. These rewards and punishments could be varied to simulate a range of real-world environments.

The simulation was executed 500 times for each type of network, the results of which can be seen in Figure 3. The framework’s ability to stabilize at an acceptable state is quite good. The random network had the most variance in stabilization time and accuracy, but averaged to reasonable levels. The grid’s performance was as expected. When the “best fit” agent was centrally located, it took approximately half the distance across the network to stabilize, while “best fit” agents in the corners caused propagation to take much longer. The scale-free network had more variation than initially anticipated since the distance across a scale-free network is quite short, but this variation can be attributed to the location of the ideal traits. If the “best fit” agent is also one of the network hubs, the system distributes the trait very rapidly. If the ideal trait is located in a peripheral agent, the transfer of the trait is slow until it hits a major communication artery.

#### B. Candidate Identification

The experimentation done in the Candidate Identification component is different than what was done in the previous component. Rather than prove that the entire component is viable, experimentation was done to determine which methods of social network analysis are most appropriate and best determine the candidates for injection. That is, the goal is to determine what type of prominence is most appropriate for the framework’s needs.

To determine the usefulness of various social network analysis algorithms, a variation on the first experiment was

performed. A network of agents with random traits and capabilities was generated. Then, the SNA algorithms were allowed to determine the key agents in the system. The results of these formulas would be interpreted as predictions of which agents would be ideal candidates for injection. Next, the trait adoption component was executed, and the network was allowed to stabilize at some best traits. At this point, the systematic exploration began. Each agent was chosen in turn to be the “ideal candidate.” This agent would be injected with a configuration known to be ideal, and the system would be allowed to re-stabilize. The influence of that agent on the network (e.g., the number of agents influenced) was recorded, the state of the network was rolled back, and a different agent was chosen as the test subject. Each agent was similarly tested to determine how accurately the particular formula predicted the location of ideal injection candidates. The results of this experiment are shown in Figure 4.

The results of these experiments are preliminary and may change as more data is processed. In addition, speedup caused by candidate selection has not been analyzed but will be in the future. However, with the data at hand, it appears Closeness has the best balance of useful information and ease of implementation in a real-world system.

### C. Trait Injection

The experimentation performed on this component examined the ability of the system to quickly and easily respond to the administrator’s changes as it attempts to achieve an acceptable state. A system that can achieve perfect configuration but requires the administrator to perform tweak after tweak ad nauseam is generally not as desirable as a system that achieves a *nearly* perfect configuration with only a couple of simple clicks of the mouse.

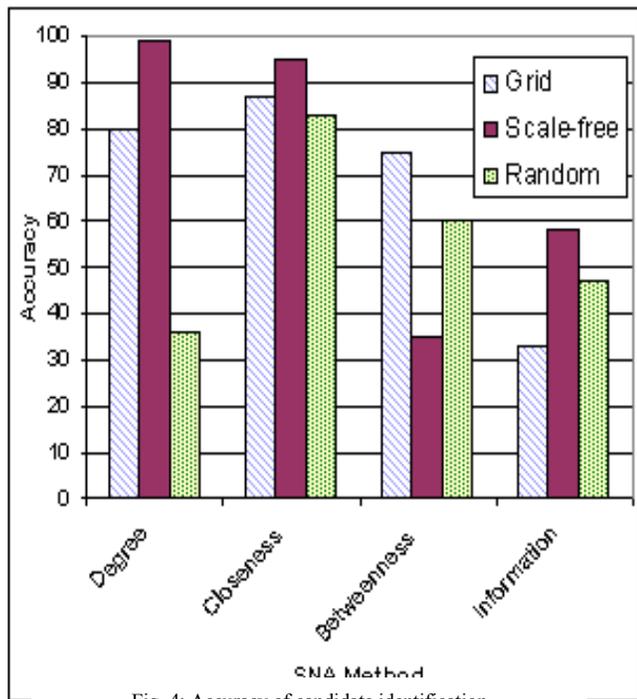


Fig. 4: Accuracy of candidate identification

To test the system’s ability in this regard, a simulation involving a human component was performed. A network of agents was generated and allowed to stabilize as in previous experimentations. Then, the agent administrator (the human subject) was allowed to make some number of injections while the system attempted to re-stabilize. When the system achieved the desired level of configuration, the experiment ended and the number of injections was recorded. The number of injections was compared to the ideal (and unrealistic) system in which the administrator would only be required to make a single injection for each unique reward/punishment region in which the agents operated. For example, if two regions existed in the experiment, each with different ideal traits, the administrator would only be required to make two injections in an ideal system.

This experiment gives a good indication of the framework’s real-world performance as shown in Figure 5 which shows an average configuration curve over time. The framework’s ability to approach the unrealistic ideal only falls short in cases where a single region is partitioned, either by one or more other regions or by lack of communication lines between agents. Such a region could be considered to be multiple, separate regions. As such, the framework typically requires one injection for each region (natural or artificially partitioned). However, barring operator error, the system often requires an average of less than one injection per region as some regions are able to achieve near ideal states without any intervention. In fact, the configuration goal had to be increased to 98% of ideal in order to achieve interesting data due to too many regions achieving the lower but acceptable standard without any human involvement.

## IV. CONCLUSIONS AND FUTURE WORK

The Viral Trait-Spreading methodology presented here is expected to greatly simplify the agent administrator’s task of managing massive systems of agents while maintaining robust capabilities. By utilizing a series of interrelated tools, agents will obtain efficient configurations without requiring the administrator to resort to micromanagement. This project is ongoing, but initial research has shown the framework to perform as predicted.

This methodology can be used as a tool for the development of the meta-reasoning component of multi-agent systems. In [12], a model for meta-reasoning was proposed that allowed the agent to modify its own reasoning process. If elements of the meta-reasoning process are parameterized as traits, the work presented here can determine when to trigger an agent’s need to examine its current reasoning methodology to potentially adopt a more successful methodology and would present potentially better configurations to the agent. With proper utilization of context, an agent can observe and learn what reasoning methodologies are appropriate for various circumstances.

Future plans include examination of other methods of modeling or mathematically defining the network for better prediction of network behavior. Plans have been set in motion to explore partially observable Markov decision processes as well as game theory in the context of controlling the massive viral spread.

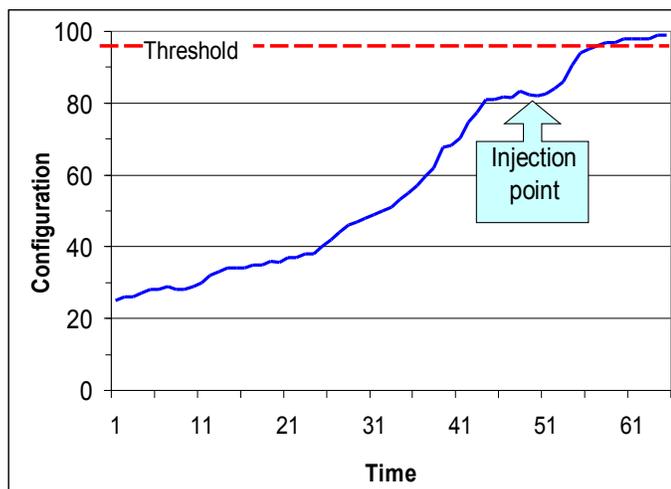


Fig. 5. Typical configuration curve

Conceptualizing and developing methods for quantification of network stability and resilience is an emerging topic in social network analysis research. Applications of these methods are of interest to the US Department of Defense. The work done to determine key agents can be utilized to explore mechanisms for intentionally (de-)stabilizing a network.

This work can be expanded in other ways as well. In particular, traits could potentially be used to create self-organizing structures such as hierarchies, teams, and so forth. In addition to automated formation of desired organizations, this methodology could be developed further to automatically initiate transitions between organizations as conditions change.

Another track is to explore limits of the administrator's depth and breadth of network control. Defining and quantifying the administrator's power of network influence will require exploration of leadership qualities and cognitive processes for understanding intent and effective methods of transferring intent to the network.

Additionally, work can be done in automating the administrator's job. Although the requirements for this research specified that a human maintain nominal control over the system, the administrator does not necessarily need to be human. Related research could be done exploring the viability of various automation techniques.

## REFERENCES

- [1] Hexmoor, H., McLaughlan, B., Tuli, G., "Natural human role in supervising complex control systems." *Journal of Experimental & Theoretical Artificial Intelligence*. Vol. 21, No. 1, March 2009, 59-77.
- [2] Ishida, T., Gasser, L., Nakashima, H., eds., *The First International Workshop on Massively Multi-Agent Systems, MMAS 2004*.
- [3] Sloman, M., Lupu, E., "Policy specification for programmable networks", *Proceedings of the 1st International Conference on Active Networks*, Berlin, Germany, ed. S. Covaci, Springer Verlag, June 1999.
- [4] Knoke, D., and Burt, R.S., "Prominence." In Burt, R.S., and Minor, M.J. (eds.), *Applied Network Analysis*, pages 195-222. Newbury Park, CA: Sage, 1983.
- [5] Bavalas, A., "A mathematical model for group structure." *Human Organizations*. 7:16-30, 1948.
- [6] Proctor, C.H., and Loomis, C.P., "Analysis of sociometric data." In Jahoda, M., Deutsch, M., and Cook, S.W. (eds.), *Research Methods in Social Relations*, pages 561-586. New York: Dryden Press, 1951.

- [7] Bavalas, A., "Communication patterns in task-oriented groups." *Journal of the Acoustical Society of America*. 22:271-282, 1950.
- [8] Sabidussi, G., "The centrality index of a graph." *Psychometrika*. 31:581-603, 1966.
- [9] Freeman, L.C., "A set of measures of centrality based on betweenness." *Sociometry*. 40:35-41, 1977.
- [10] Stephenson, K., and Zelen, M., "Rethinking centrality: methods and applications." *Social Networks*. 11:1-37, 1989.
- [11] Castelfranchi, C. *Trust and Deception in Virtual Societies*. Springer, 2001.
- [12] Cox, M., and Raja, A. "Metareasoning: A Manifesto", in *Proceedings of AAAI 2008 Workshop on Metareasoning: Thinking about Thinking*, pp 1-4, Chicago, IL. July 2008.