Towards Intelligent Control of Influence Diffusion in Social Networks

Andrew Runka, Tony White
School of Computer Science, Carleton University
{arunka, arpwhite}@scs.carleton.ca

***Abstract*—Control of the flow of information in large-scale non-deterministic social networks is a complex problem requiring both a search for the optimal connection of a control system to the network, and a means of determining the required control signals. This paper formalizes the Network Control Problem (NCP) as a means of relating the field of diverse social network control subproblems. Additionally, this paper defines a novel NCP subproblem, the θ-Consensus Avoidance Problem (θ-CAP), as a next step towards solving the general NCP. Benchmark results for the θ-CAP using Artificial Neural Networks, Evolutionary Neural Networks, and heuristic methods are presented, and interesting areas of the problem space are identified.**

***Index Terms*—nfluence Optimization Control Social network-snfluence Optimization Control Social networksI**

I. Introduction

Opinion, and the ability to influence opinion, are powerful forces in the information age. From advertisers seeking accurate and cost-effective means of using viral marketing, to politicians and activist groups attempting to sway public opinion with propaganda and grass roots advocacy, to counter-terrorism efforts seeking to identify and disrupt terrorist cells, the imposition of external influence on large-scale social networks is an important problem that could benefit from accurate automation of control.

A social network can be described using a graph representation of the interactions between agents. The state of an agent (e.g., its opinion, infection status, etc.) is determined largely through interactions with neighbouring agents in the network, who were, in turn, affected through prior interactions. Furthermore, it can be reasonably asserted that the state of an agent dictates its behaviour (e.g., purchasing or voting behaviour is based on opinions regarding a product or candidate). So the state of an influential agent affects the state, and thus the behaviour, of those around it. Influence diffusion is the process by which this state transmission occurs.

Liu et al. [18] studied the controllability of complex networks. The goal of their analysis was to find the minimum number and location of required control sites in order to direct the state of the social network. Absent from this problem is an exploration of an efficient means of determining the control signals required to direct the state of the network. Similarly, Kempe et al. [12], [13] studied the selection of influential nodes in social networks with the goal of maximizing the spread of a given state. In both of these works, the *controllability* of a network's state is examined, but not its actual *control*. That is, these works perform a search for what to control, but not how to control it.

This paper introduces the Network Control Problem (NCP) as the combined search for both the structure and behaviour required to achieve generalized control of a social network. The goal of the NCP is to dictate the state of the network with a high degree of precision, using the processes of that network (i.e., influence diffusion), and quantifiable amounts of external input. A number of existing problems including the well-known Influence Maximization Problem are cast in the form of NCP subproblems.

The θ-Consensus Avoidance Problem (θ-CAP) is introduced in this paper in order to address an absence in the literature of benchmarks for network controller behaviour optimization. The θ-CAP is inspired by the well-studied pole-balancing benchmark problem [1] from control theory literature. The objective of the θ-CAP is to balance the state of a social network in order to prevent the terminal state of any θ-sized proportion of nodes from converging to consensus for as long as possible, given a network dynamic that tends towards consensus. The ability to identify network states of high risk, as well as to guide the network back to more desirable ones, are essential components of our ability to maintain diversity within a network. So, an effective understanding of the θ-CAP has wide reaching implications. As examples, a lack of diversity in genetic expressions could leave a population vulnerable to disease, or lack of choice in a product or political market could lead to monopolistic prices or dictatorship respectively.

II. Network Control

The Network Control Problem (NCP) formalizes the general case of all related subproblems by means of the following components:

- A *social network* of nodes each containing some state information,
- a *diffusion model* governing the communication of states between nodes,
- a *control system* that manipulates the network via signals,
- a *configuration* specifying the connection of the control system to the network, and
- an *objective function* that quantitatively evaluates any given solution.

The objective function specifies the goal of the specific problem at hand and is given no required structure. The remaining components are each discussed below.

A. The Social Network

The social network is modelled as a graph $G = (V, E)$, where V is the set of nodes and E is the set of directed edges including self loops. Each node $v \in V$ represents an agent, and contains an instantaneous state value $s(v, t)$ at each time step t. The set of all node states cumulatively define the network state $s(V, t) = \{s(v, t)|\ v \in V\}$. The domain of node states is problem dependent. Each edge $e \in E$ is a directed relation between two nodes (e.g., $e_{u,v}$ is the edge from node u to node v), and is associated with a weight value $w(e, t)$. NCPs are not limited to any specific network structure; however, typical real world social networks have been shown to exhibit scale free characteristics [4].

B. The Diffusion Model

The Diffusion Model (DM) defines the behaviours of nodes as communicating agents in the social network. Here, a framework is presented that formalizes the existing definitions.

A DM can be defined as the pair of behaviours $DM(v) = \{Sh(v), L(v, I)\}$. The two parts of the DM correspond to two phases of communication in the social network. First, each node shares information regarding its current state along each outbound edge in its neighbourhood according to its sharing strategy $Sh(v)$. Next, each node adjusts its state according to its learning strategy given the inbound state information, $L(v, I)$. An optional third phase (not considered in this paper) may contain operations that modify the neighbourhood or connected edge weights according to new information. This process is repeated for each time step until some termination condition is met.

The process of state transition for a given node can now be described formally as the following:

$$s(v, t + 1) = L(v, I(v, t)) \tag{1}$$

$$I(v, t) = \{(w(e_{u,v}, t), Sh(u)) \mid u \in N_i(v, t)\} \tag{2}$$

Here $N_i(v, t) = \{u \mid e_{u,v} \in E\}$ is the inbound neighbourhood of v at time t, and $I(v, t)$ is the set of information tokens transmitted from all such neighbouring nodes. An information token is the output of a node's sharing strategy (e.g., the node's state) paired with the weight of the edge along which the token travelled.

A *progressive* DM is one in which the state of a node can only be changed a single time. Non-progressive models, are those for which node states may change any number of times during simulation. Node states may be *memoryless* in that they model only a single state at a time, or they may contain a history of states. Nodes may use trust, similarity, or other measures to alter the weights on their connected edges. The exact design choices in any of the strategies are not constrained in the general definition presented here.

C. The Control System

The control system, C, is a set of one or more controllers that connect to the social network and alter its state via interactions at the node-state level. There are two parts to the control system corresponding to two dependent optimization problems. The first part of the control system deals with configuration. The second part deals with behaviour. These are described below.

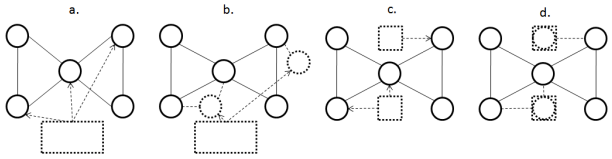

Fig. 1. Types of controller configuration. Solid lines are the social network, dashed lines are the control system components. **a** - Direct control. **b** - Indirect control. **c** - Distributed Direct control. **d** - Distributed Indirect control.

Controller Configuration: A controller configuration defines the locations in the network to which the controller will apply its signals, and from which it observes the state of the network.

The set of nodes connected to a controller $c \in C$ are known as controlled nodes, and are defined as $V_c = \{v \in V \cup P \mid \exists e_{c,v}\}$, where P is the set of proxy nodes (described below). The set of all controlled nodes is $V_C = \{V_c \mid c \in C\}$. The state of each controlled node is directly set to the value of the control signal it receives, that is $s(v, t) = \sigma(c, v, t)$ *iff* $v \in V_c$, where $\sigma(c, v, t)$ is the component of the control signal from controller c for node v at time t. Therefore, the structure of the control signals are confined to a single value for each controlled node.

Immediately prior to determining a control signal, the control system probes the visible state of the network. Network visibility need not be constrained in the general NCP definition, relying instead on problem-specific context. Throughout the remainder of this paper, however, the information regarding the network that is visible to the control system, $S(C, t) = s(N(V_C, t'))$, is the state of all neighbouring nodes at the time of command. The parameter t' is used here to distinguish between information passed between nodes during the sharing phase.

The control system may control the state of the social network directly or indirectly. Direct control (see Figure 1a,c) indicates that a controller will directly set the states of controlled nodes at each time step. Indirect control (see Figure 1b,d) indicates that a controller is setting the states of nodes outside of the network, which then communicate with nodes in the network via the diffusion model. These external 'proxy' nodes have state and connections like a typical node, but do not alter their own state as a result of influence diffusion, nor does their state contribute to the network state. The distinction between indirect and direct control is that in the direct control scheme $V_C \subseteq V$, whereas in the indirect $V_C = P$. This paper considers only the former, while the latter is the subject of ongoing research.

Each node $v \in V$ is associated with a cost, $cost(v)$, and a controller is given a finite budget, B, with which to select nodes for the controlled set:

$$\sum_{v \in V_C} cost(v) \leq B \tag{3}$$

The node cost formula varies by implementation in the literature from unit costs per node to problem specific cost

distributions. This paper considers the cost of connecting a controll to a node to be equal to the number of outbound edges originating from the node.

Controller Behaviour: The behavioural component of a controller c is responsible for determining a mapping between the inputs drawn from the social network and optimal outputs in the form of control signals, $\{I(V_c,t)\} \rightarrow \{\sigma(v,t) \mid v \in V_c\}$. The input information regarding the network that is visible to the control system, $I(V_C,t)$, is problem specific corresponding to the control system's available sensors. The output values specified are restricted to the domain of the controlled nodes' allowable states. The mapping is executed once per time-step, thereby generating an input-dependent time series of heterogeneous values for each controlled node.

The means used to generate these values are not restricted in the NCP definition. There may be one or many individual controllers independently generating signals. Multiple controllers may communicate or operate in isolation, they may be homogeneous or heterogeneous with respect to their design and their decisions.

NCP Subproblems: NCP subproblems arise through specification of the above components, and through the narrowing of constraints and objectives. Three main classes of NCP subproblem are identified here: *Targeted*, *Trajectory*, and *Homeostatic*. In targeted NCPs, the goal is to drive the network towards a specific goal state. A trajectory following NCP is an asynchronous targeted NCP, in which the goal is to have each node pass through a specific set or sequence of states. Finally, a homeostatic NCP has a goal of maintaining the network within a narrow range of states when the natural diffusion of the network tends away from stasis.

III. Related Work

The Network Control Problem and Diffusion Model framework defined in the previous section are derived from commonalities among existing problems and models. They are intended to present unifying descriptions of various implementations from their respective bodies of literature, as shown below.

A. Influence Maximization Problem

The Influence Maximization Problem (IMP) is a targeted NCP first presented as an optimization problem by Richardson and Domingos [7], [23] and later formalized by Kempe et al. [12], [13]. The objective of the IMP is to find a set of nodes in the network that maximizes the spread of a given state.

Objective Function: Formally, given a social network graph with node states $\{0,1\}$ the goal is to find:

$$argmax_{V_C \subseteq V} R(V_C) \tag{4}$$

given that $s(v,0) = 1 \ \forall v \in V_C$ and $s(v,0) = 0 \ \forall v \notin V_C$, subject to the given influence diffusion rules and the budgetary constraint:

$$\sum_{v \in V_C} cost(v) \leq B \tag{5}$$

where $R(V_C) = E[|V^1(t^*)|]$ is the expected return in terms of the number of nodes with state 1 at the time of quiescence t^* given the initial set V_C, $s(v,t)$ is the state of node v at time t, $cost(v)$ is the cost of adding node v to the initial (controlled) subset $V_C \subseteq V$, and B is the upper bound on incurred costs.

Network Structure: The IMP is commonly studied in a 2-state variant, where nodes are initially all inactive, and become active via the diffusion model. Variants of the IMP with greater than two states exist, in which each additional state represents a competing source of influence (e.g., rival product) in the network [2], [3].

The IMP has been studied over a variety of real-world network structures including co-authorship graphs [12], [3], [15], blog links [14], and recommender networks [23], [7].

Diffusion Model: The IMP has been studied under a variety of DMs. Prominent examples include the progressive variants of the Linear Threshold (LT) and Independent Cascade (IC) models, presented in [12], [13]. In the LT, a node becomes activated if there is sufficient pressure from activated neighbouring nodes. Formally,

$$Sh(v) := s(v,t) \tag{6}$$

$$L(v,I) := \begin{cases} 1, \ \textit{iff} \ \sum_{(w,s) \in I(v,t)} s \cdot w \geq \theta_v \\ 0, \ otherwise \end{cases} \tag{7}$$

where θ_v is the threshold of activation for node v.

In the progressive IC, once a node is activated, it has a one-time chance of activating each neighbouring node on the subsequent time-step. Formally,

$$Sh(v) := \begin{cases} s(v,t), \ \textit{iff} \ s(v,t-1) \neq s(v,t) \\ 0, \ otherwise \end{cases} \tag{8}$$

$$L(v,I) := \begin{cases} 1, \ with \ P(\sum_{(w,s) \in I(v,t)} s \cdot w) \\ s(v,t), \ otherwise \end{cases} \tag{9}$$

Even-dar and Shapira [8] consider the problem of influence maximization in the context of the Voter Model (VM). This non-progressive process is shown to converge to consensus in O($|V|^5$) time. In the VM, as introduced in [11], a node changes its own state by adopting that of a random neighbouring node. Formally,

$$Sh(v) := s(v,t) \tag{10}$$

$$L(v,I) := \begin{cases} 1, \ with \ probability \ \frac{|\{s=1 | (w,s) \in I(v,t)\}|}{|I(v,t)|} \\ 0, \ with \ probability \ \frac{|\{s=0 | (w,s) \in I(v,t)\}|}{|I(v,t)|} \end{cases} \tag{11}$$

Control System Configuration: The influence maximization problem corresponds to the search for an optimal configuration with which to initiate an influence cascade. It is shown to be NP-hard regardless of the diffusion model used [3], [12]. As such, many techniques have been suggested for addressing the problem effectively and efficiently. A greedy hill-climbing approach to node selection is described in [12], [13], and experimentally demonstrated to out-perform high-degree and centrality heuristics on the IC and LT models. The greedy algorithm iteratively adds to the initial set, the node with the highest marginal gain in influence diffusion across the network, as estimated through extensive Monte Carlo simulations. It is shown that the greedy algorithm approximates the optimal

set to within a factor of $(1-1/e)$. Leskovec et al. [16] consider a proportional version of the greedy algorithm, which factors in the node cost. That is, a node v is selected such that:

$$argmax_{v \in V - V_{C_{m-1}}} \left(\frac{R(V_{C_{m-1}} \cup \{v\}) - R(V_{C_{m-1}})}{cost(v)} \right) \quad (12)$$

where V_{C_m} is the set of selected nodes of size m, and $R(V_{C_m})$ is the return function given the initial set V_{C_m}. The bounds are proven to be within a factor of $\frac{1}{2}(1-1/e)$ of the optimal solution. Leskovec et al. also suggest the use of *lazy evaluation* to take advantage of the sub-modular property of the problem, and gain a significant speed up in place of evaluating the marginal gain of every node at every step in the greedy algorithm.

An alternative means of increasing the efficiency of evaluation is the use of approximation heuristics. Gui-sheng et al. [10] compare the performance of Genetic Algorithms, Differential Evolution, and Particle Swarm Optimization with the preferential attachment heuristic. Their findings show that the highest quality performance is achieved using genetic algorithms. However, the results are neither compared in terms of run time nor against the greedy heuristic for quality. Nevertheless, approximation heuristics are a promising approach to this NP-hard optimization problem.

Control System Behaviour: The objective for the IMP assumes the optimal control behaviour, thus eliminating the need to optimize this NCP component. The control system outputs a single value (active) to every controlled node. Formally,

$$\sigma(c, v, t) = 1, \ \forall v \in V_C \quad (13)$$

Discussion: The IMP is a prominent combinatorial optimization problem. It represents a simplified version of the general NCP by removing the search for an optimal behavioural component, allowing a great deal of research effort to focus on solely the configuration component. However, disregarding variety in the behavioural component, as well as overlooking the dependence of the two search problems, limits the utility of the IMP in terms of leading to a complete solution for the NCP.

B. Structural Control

The structural controllability of large scale networks was studied in [18] and [27] for both linear and non-linear network processes respectively. This is another example of a targeted NCP. The objective is to select the minimum required set of controlled nodes in order to deliver independent signals to every node in the network, given knowledge of the exact transmission process.

Objective Function: The goal of structural controllability is to find the minimum set of controlled nodes which maintain Kalman's controllability rank condition:

$$argmin_{V_C \subseteq V}(rank(M) = |V|) \quad (14)$$

$$M = (E_C, EE_C, E^2E_C, ..., E^{|V|-1}E_C) \quad (15)$$

where E_C is the matrix of edges connecting the set of controllers C to the nodes V, and E^2 is the square of the edge matrix E.

Network Structure: The controllability of a large number of real world networks as well as instances of the Erdös-Rényi and scale-free network models was studied in [18].

Diffusion Model: The problem is defined using linear time-invariant dynamics. In terms of the DM framework:

$$Sh(v) := s(v, t) \quad (16)$$

$$L(v, I) := \sum_{(w,s) \in I(v,t)} s \cdot w \quad (17)$$

Non-linear network dynamics are estimated using successive linear steps in [27].

Control System Configuration: In [19] it is shown that the minimum number of required control nodes is 1 if a perfect matching exists on the graph. Otherwise, the minimum number and location of the control sites can be determined efficiently by computing a maximum matching on the graph, and then applying control to any unmatched nodes.

Control System Behaviour: The determination of specific signals to be passed into the network is not considered as part of the structural controllability problem. Given state transitions defined as linear time-invariant dynamics, solutions to the specific control problem can be determined analytically by solving the linear equations of the system. This is prohibitive, however, as the problem size and complexity increase.

Discussion: The structural controllability problem combines the fields of network analysis and control theory and, as such, was a significant motivation for the development of the NCP. The lack of behavioural effects in this problem may put the derived results at odds with other NCPs. For example, the strong correlation found between a network's degree distribution and its controllability in [18], may be at odds with the findings of [12] that a greedy selection of controlled nodes w.r.t. marginal influence gain outperforms a degree distribution heuristic given the DMs used therein.

Like the IMP, the structural controllability problem omits the search for appropriate control system behaviour. However, by leaving this component unspecified (as opposed to rigidly defined), there is an opportunity to extend the work to consider scalable solutions.

C. Immunization Problem

The Immunization Problem (IP) studied in [5], [22], is an homeostatic NCP. The IP is defined as selecting an optimal set of nodes from the network to immunize in order to prevent the spread of an infectious process.

Objective Function: Formally,

$$argmin_{V_C \subseteq V}(\sum_{u \in V} s(u, t^*) | s(v, t) = 0 \ \forall v \in V_C, \forall t) \quad (18)$$

Network Structure: The IP is considered on a graph of nodes with two discrete states {0,1}, such that an individual with state 0 represents an uninfected (susceptible) individual, and state 1 represents an infected individual. Initial conditions, including the introduction of the infection, are considered dependent on the specific instance under consideration.

The IP is analyzed over instances of the scale-free graph model in [5], [22] due to the commonality of that structure in

real-world social networks. Edges between nodes are typically undirected and unweighted due to the use of model-specific global transmission rates in the standard infection DMs.

Diffusion Model: The IP is studied under the Susceptible-Infected-Susceptible (SIS) DM in [22]. Healthy nodes become infected at a rate, ν if they are connected to infected nodes, and infected nodes are cured at rate δ becoming susceptible to infection once again. Formally,

$$Sh(v) := s(v,t) \tag{19}$$

$$L(v,I) := \begin{cases} 1,\ with\ probability\ \nu,\ if\ |N^1(v,t)| \geq 1 \\ 0,\ with\ probability\ \delta,\ if\ s(v,t) = 1 \\ s(v,t),\ otherwise \end{cases} \tag{20}$$

where $N^k(v,t) = \{u \mid s(u,t) = k \in I(v,t)\}$ is the set of nodes states equal to k in the information token I received by node v at time t.

Another DM from the standard infection literature is studied in [5], the Susceptible-Infected-Removed (SIR) model. The SIR works like the SIS, except infected nodes do not become susceptible again, and are instead removed from the network.

$$Sh(v) := s(v,t) \tag{21}$$

$$L(v,I) := \begin{cases} 1,\ with\ probability\ \nu,\ if\ |N^1(v,t)| \geq 1 \\ -1,\ with\ probability\ \delta,\ if\ s(v,t) = 1 \\ s(v,t),\ otherwise \end{cases} \tag{22}$$

Following the state transition phases of the DM framework, the SIR invokes the optional network adjustment phase to extract the removed nodes from the network:

$$V = V - \{v | s(v,t) = -1\} \tag{23}$$

Control System Configuration: Targeted immunization strategies for scale-free networks are presented in [22] and [5]. In the former, nodes are selected for immunization proportionally according to their degree. In the latter, the immunized node is determined by first selecting a node at random, and then immunizing one of its neighbouring nodes. These strategies have similar effects of selecting the most highly connected individuals, with the latter being a stochastic estimate of such based on local information.

Control System Behaviour: In the infection literature, immunized nodes are removed from the network, thereby preventing infection spread over their associated links. This can be modelled as an direct control NCP as follows,

$$\sigma(c,v,t) = 0,\ \forall c \in C,\ \forall v \in V_c,\ \forall t \tag{24}$$

This effectively prevents the infection state from spreading through any controlled nodes.

Discussion: The IP is limited to a search for the optimal configuration. The control system behaviour is again dictated explicitly by the problem definition.

The IP contrasts with the IMP, in that the immunization state does not spread as the active state of the IMP does. So, while both non-dynamic control systems directly output the state to be maximized to a set of desirable nodes, the means by which this alters the diffusion of information in the network differs sharply. One creates the root of a diffusion tree, while the other breaks links in the network.

D. Summary

The foregoing discussion brings together some of the existing problems that can be considered to fit in the NCP framework. It should be noted that this is not intended to be an exhaustive list, nor a complete exposition of all of the research relating to each problem. The intent here is to illustrate the way in which a number of diverse problems relate when placed in the scope of the NCP framework.

There exists a gap in research of problems relating to the NCP. To this author's knowledge, no studies have been conducted regarding the search for the optimal control system behaviour component. The next section will address this shortcoming by introducing a novel subproblem of the NCP.

IV. θ-Consensus Avoidance Problem

The θ-Consensus Avoidance Problem (θ-CAP) is a novel homeostatic variant of NCP, in which the goal is to prevent any θ-sized proportion of the network from converging to consensus. This problem is derived from the failure avoidance class of problems from the adaptive control literature [1], [17].

Objective Function: The goal of the θ-CAP is to find a time series of control signals that will minimize the following state utility function over an infinite time span:

$$U(t) = \begin{cases} 0, & if\ \frac{||V^1(t)| - |V^0(t)||}{|V|} \leq \theta_G \\ 1, & otherwise \end{cases} \tag{25}$$

where $V^k(t) \subseteq V$ is the set of nodes at time t with state set to k, θ_G is a threshold parameter representing the upper bound of allowable disparity in the network, and $0 \leq \theta_G < 1$. A terminal failure state occurs when the proportion of converged node states in the network exceeds θ_G.

Network Structure: The problem is not limited to any specific structure. The domain of node states is constrained to discrete values. Variants may contain K possible states where $K \geq 2$; however, for brevity of exposition this paper will be limited to the $K = 2$ variant.

Diffusion Model: The θ-CAP may be instantiated with any non-progressive diffusion model. It is intended that the θ-CAP be instantiated with a DM which would fall to consensus within the entire network if left unopposed. A DM such as the Voter Model [8] guarantees this property. While solutions to the θ-CAP may exist on DMs that do not exhibit this quality, special consideration must be given in order to distinguish between successful control and natural convergence to non-consensus states.

Control System Configuration:

A number of potential solutions for the configuration search exist in the literature discussed in Section III. Variations of approaches such as structural controllability [18], clustering [6], greedy w.r.t. influence gain [12], [16] or node degree [8], or an evolutionary search [10] may all be considered appropriate. Note however, while these techniques have been effective for related problems such as the IMP, the goal of the

θ-CAP is not maximization and so the optimal control set is likely to differ as well.

The control system configuration is governed directly by the budget parameter B, and indirectly by the consensus parameter θ_G. Larger values of B allow a greater number of nodes to be controlled, and larger values of θ_G refer to larger allowable ranges of imbalance. Therefore, intuitively, as the parameters θ_G and B increase, the difficulty of the problem decreases. This provides a two-dimensional gradient of difficulty over instances of the θ-CAP which affects the need for intelligent decision making by the configuration component.

Theorem 1. *The control system in a direct control scheme is capable of sustaining the network from reaching consensus according to θ_G, regardless of the states of uncontrolled nodes, if the following holds:*

$$\theta_G + \frac{|V_C|}{|V|} \geq \theta_C + 1 \tag{26}$$

where $\theta_C \geq \frac{1}{|V|}$ is the upper bound of allowable difference of opinions among the subset of controlled nodes.

Proof. Let $V_U = V - V_C$ be the set of uncontrolled nodes in the network, and let $\theta'_G = \lfloor \theta_G * |V| \rfloor$ be the upper bound on allowed imbalance without consensus in terms of number of nodes. A locally optimal controller is one that maintains a minimal amount of imbalance amongst its controlled nodes. Given that the number of controlled nodes may be even or odd, a locally optimal controller can at best achieve an imbalance of 1 node in the general case.

If $|V_C| = |V| - \theta'_G + 1$ then $|V_U| = \theta'_G - 1$. A consensus among the set V_U would leave the network two nodes short of a failure state, thus requiring an imbalance within V_C of at least two nodes before failure could occur. A locally optimal controller will provide at most an imbalance of one. Therefore, a locally optimal controller with direct control to a set V_C of sufficient size can prevent consensus within the entire social network without considering the states of uncontrolled nodes if the following inequality is satisfied:

$$|V_C| \geq |V| - \theta'_G + 1 \tag{27}$$

Furthermore, the amount which $|V_C|$ exceeds $|V| - \theta'_G$ defines the amount of allowable imbalance within V_C. Therefore the controller can sustain the network from reaching θ'_G-consensus if the problem instance satisfies:

$$\theta'_G + |V_C| \geq \theta'_C + |V| \tag{28}$$

where $\theta'_C \geq 1$ is the upper bound on imbalance within the set of controlled nodes exclusively. Dividing this equation by $|V|$ yields Thm. 1. □

For instances in which Eq. (26) holds true, the controller can prevent θ-consensus using a *non-specific set of nodes*, even if every non-controlled node has converged to a single state. This defines the class of direct-control θ-CAP instances designated here as *diffusionless* instances.

Whereas the diffusionless instances do not necessarily require an intelligent configuration component, the remaining instances for which Thm. 1 is not satisfied an NP-hard [12] combinatorial search problem of selecting the optimal set of controlled nodes. The criteria for optimality are not known, and is only evaluated indirectly via the ability of the control system's behaviour component.

Control System Behaviour: The control system behaviour component is required to determine an optimal mapping between input node states and control signals. The visible input state of the network for the controller at each step is equal to the neighbourhood of all controlled nodes, $I(V_C, t) = \{s(u,t) | u \in N(v,t), v \in V_C\}$. An optimal mapping for the θ-CAP is one that minimizes the objective function (Equation (25)) over an infinite time span, or put another way, one that avoids θ-consensus in the network for as long as possible.

Formulation of Thm. 1 using θ_C defines a simpler objective for the diffusionless class of instances. The goal of these is to minimize the following state utility function:

$$U_C(t) = \begin{cases} 0, & if \ \frac{||V_C^1(t)| - |V_C^0(t)||}{|V|} \leq \theta_C \\ 1, & otherwise \end{cases} \tag{29}$$

where $|V_C^k(t)|$ is the number of controlled nodes with state equal to k at time t and $|V_C^0(t)| + |V_C^1(t)| = |V_C|$. This is simpler than Eq. (25) in that it considers only those nodes whose states are directly settable by the controller, neither needing to account for the other $|V| - |V_C|$ nodes nor the process of diffusion among them. Note, that while this view of the problem objective limits the amount of uncertainty that a controller would be faced with, it also narrows the threshold of θ-consensus requiring a more effective ability to balance states.

The means of determining an optimal mapping is the subject of experimentation in the next section and in future work.

V. Experimental Results

The experimental results presented in this paper are intended to highlight the utility of the θ-CAP as an intelligent algorithms testbed, as a difficult problem within the space of intelligent control of influence diffusion in social networks. It should be noted that the aim here is not to spend effort in obtaining the *optimal* solutions to the problem, but instead to lay out an exploratory set of benchmark results that will facilitate future research. Problem instances, source code, and further results are available online.[1]

A. Methodology

1) Controller Algorithms: In order to study the θ-CAP, the following algorithms were implemented as the control system's behavioural component: *Null*, *Random*, *Anti-Majority*, *Artificial Neural Network* (ANN), and *Evolutionary ANN* (EANN). Each of these algorithms are described briefly in turn below. Learning algorithms are applied with no optimization or tuning for the problem at hand, in order to demonstrate the nature of the problem itself. Additional algorithms, experimental tuning, and optimization are the subject of ongoing investigation and will be discuss in future works. Note, that in this preliminary study of the θ-CAP, all control systems were

[1] http://people.scs.carleton.ca/∼arunka/cap

configured by selecting nodes for the controlled set uniformly at random from the social network until the exhaustion of a given budget.

a) Null Controller: The Null controller is a place-holder for a control system. It does not output control signals and therefore has no effect on the simulation of the social network's convergence. This is used as one of two experimental controls for comparison in the results that follow.

b) Random Controller: The Random controller determines a control signal by generating a uniform random output for each controlled node. This is the second of two experimental controls used in this study.

c) Anti-Majority Controller: The Anti-Majority controller uses a heuristic method that reads the state of all visible nodes around a given controlled node (i.e., their neighbourhood including self-loop), and sets the state of the controlled node to the least common state observed. In the case of a tie the state is set uniformly at random.

d) Artificial Neural Network: An Artificial Neural Network (ANN) is an interconnected layered network of linear separating units capable of acting as a universal function approximator. It is appropriate for use here given that the exact form of the state to signal mapping required of the control system is not known. For a more in-depth description of simple feed-forward ANNs see [9], [20], [25].

In an unsupervised training scenario, such as the present work, there is no direct feedback regarding the quality of the control signals generated. One method of estimating this information is to use successive evaluations to determine the change in fitness with respect to changes in controller outputs. This is the method used by Anderson [1] in his application of an ANN to the pole-balancing control problem. The gradient can be estimated as:

$$\frac{\delta f}{\delta \sigma(c,v)} \approx \frac{f(t) - f(t-1)}{\sigma(c,v,t) - \sigma(c,v,t-1)} \tag{30}$$

where $f(t)$ is some evaluation of the state of the social network at time t (see Section V-A3 below for details on the metrics used here), and $\sigma(c,v,t)$ is the control signal passed from a controller c to a controlled node v at time t.

The imprecision of the error estimate is mitigated by the use of the RPROP algorithm [24]. Here, only the sign of the error is used in the weight adjustment rules. The absence of parameter tuning requirement for RPROP is an additional benefit for its use in these preliminary results.

The neural networks used consist of three layers. The input layer is given sufficient inputs for all information received at the controlled nodes (i.e., $|S(V_C,t)|$ less overlap). The output layer contains an output node for each controlled node in the network (i.e., $|V_C|$). The hidden layer was set to 10 hidden nodes. This value was not experimentally tuned, and was kept small to minimize run-times.

Training of the neural network consists of 100 independent simulations. Weight updates were performed in batch-mode after each complete simulation. Early stopping of training occurred if the fitness of the training simulation remained unchanged for 5 successive runs.

e) Evolutionary Artificial Neural Network: The Evolutionary Artificial Neural Network (EANN) controller uses the same structure as the ANN controller to handle the state to signal mappings. However, in place of a gradient descent technique for learning, it uses an Evolutionary Algorithm to search for optimal weights in the weight space.

Specifically, the "Mutate-nodes" operator of [21] was used throughout this study. This operator applies a random shift to the weights of all edges originating at a single node in the neural network. A population of 100 such networks are used as the EA search set. An elitism of the single best performing individual of each generation is kept aside, and retested once at the end of each generation. Evolution stops after 100 generations or once the elite individual's fitness remains unchanged for 5 successive generations.

2) Instances: The instances of the θ-CAP studied here are a combination of the structure of the social network and the instance parameters θ_G and B.

The network structure is determined by the number of nodes, the instance type, a degree parameter, and a random generator seed. The number of nodes is kept constant at 100 for simplicity of exposition throughout this paper. The instance type specifies the generator used to construct the network. Here, two generators are considered: Random Graph (RG) and Preferential Attachment (PA).

The RG-type networks are constructed using a nearest neighbour style random generator, in which nodes are repeatedly selected at random and connected to the nearest node in the network that is not already attached to the selected node. The degree parameter, d, corresponds the number of connections added to the graph as $d * |V|$. Note that the use of bidirectional edges means that two edges are considered to exist between each pair of connected nodes, so only $\lfloor d*|V|/2 \rfloor$ edges are added randomly. Additionally, all nodes contain self-loops, therefore the average node degree is $d+1$, and the total number of edges is $|E| = d \cdot |V| + |V|$.

The PA-type networks are constructed using preferential attachment, given a fully connected seeding graph of $d+1$ nodes. The degree parameter in PA networks corresponds to the number of new edges with which each node is attached when added. Each new node added to the graph connects to d existing nodes stochastically biased towards nodes with high degree. Again, self-loops and bidirectional edges are used, and so $|E| = 2d \cdot |V| + |V|$.

Once the social network is generated, the parameters θ_G and B define a θ-CAP instance. As described in Section IV, with θ_G limiting the allowable range if disparity in the network, and B limiting the number of controlled nodes, these parameters control the difficulty of the problem for any given network structure.

Instances are described by an abbreviation of all of the above information. For example, the instance rg.n100.d5.b100.t30.i2 corresponds to a network structure that was generated using the RG-type network generator with 100 nodes, degree parameter 5, and instance seed 2. The budget parameter is set to 100 meaning a configuration can select 100 edges worth of nodes, and the θ_G parameter is set to 30 meaning a difference of greater than 30% of the nodes in the

network constitutes a failure (e.g. $|V^1| = 66$ nodes).

3) Metrics: The presented results are each derived from 30 runs on a given network structure, using differing seeds for the control system where appropriate (including the random configuration). Each run consists of an optional training phase followed by 100 test runs. The presented results are averaged over all 30 runs, which are in turn averaged over 100 tests.

The quality of a control system is primarily measured as the duration with which it can maintain the social network within the bounds of θ_G. This *run-length* metric is capped at 500000 steps of the social network simulation. Run times were measured independently on an Intel Core i7-3632QM processor with 8GB of RAM.

B. Results

The varying difficulty of the θ-CAP is most readily depicted in the gradient surface plots of Figure 2. By tuning values of the θ_G and B parameters, problem instances vary in difficulty, as measured by simulation run lengths, from immediate failure to reaching the simulation run cap.

The Null controller surface (Figure 2a) shows changes in the problem difficulty with respect to θ_G, as well as the speed of convergence of the network in the absence of a control system's intervention. The gradual incline of the run length in correlation with increased threshold values demonstrates that the convergence process under the VM is a rapid one. Average and standard deviations of run lengths over 10 different network structures using the null controller with a 90% θ_G threshold are presented in Table I. Faster convergence in the PA-type networks corresponds to shorter average path lengths.

TABLE I
AVERAGE AND STANDARD DEVIATION OF RUN LENGTHS GIVEN θ_G=90% USING THE NULL CONTROLLER OVER 10 NETWORK STRUCTURES OF EACH TYPE.

Type	Average	St.Dev.
RG	135.74	43.28
PA	35.66	3.04

The second baseline of comparison is derived from the ability of a random generator to act as the control system's behaviour component. The surface of results achieved by the Random controller (Figure 2b), demonstrates changes in the difficulty of the problem space over combinations of θ_G *and* B. It can readily be seen that for easier instances (corresponding to larger values of these two parameters), the Random controller provides sufficient input to the social network to keep it from θ-consensus until the simulation run limit. Intuitively, random noise is reasonable choice of approach when opposing the uniformity of consensus.

The sharp increase in ease of control corresponds to Theorem 1. The budget value required to create a diffusionless instance with a given θ_G can be calculated using Equation (26). For example, given θ_G = 0.7, $B^* = |E| \cdot ((1/|V| + 1 - \theta_G)/V) = 186$. Note, this budget assumes the average cost for every node ($|E|/|V|$). Over 30 successive random configurations the actual number of nodes controlled may vary to either side of the required $|V_C|$. These boundary budget values, B^*, are found along the lower edge of the diagonal ridge of the Random controller surface. Sample results on and around the B^* boundary for a fixed θ_G are presented in Tables II and III. The values in these tables present the span of roughly two squares in the surface graph. An exponential growth is observed over this span in Table II for the Random controller. In Table III, however, this growth is not observed given the tight θ_G=0.3 constraint. A similar lack in growth of results is observed over θ_G values below this range in Figure 2b.

The Random controller is *not* considered a locally optimal one. Its outputs are drawn from a uniform distribution, meaning the combined state of the controlled nodes is governed by a binomial distribution. This is most clearly seen in Figure 2b given a full-control budget, $B = |E| = 600$ ($|V_C| = |V|$). Here the output distribution has a mean of 50 (meaning it outputs a given state to 50 nodes and the opposing state to the remaining $|V| - 50 = 50$ nodes) and a standard deviation of $+/-5$. Note, that $+5$ of one state is also -5 of the other, resulting in a network state of 10. Therefore, a $\theta_G = 10\% = 10$ nodes, corresponds to $\approx 32\%$ chance of failing on the first step. The expected run length of the simulation is thus approximated by a geometric distribution where θ_G denotes the critical range. Given this information, the expected duration of a simulation with $\theta_G = 40\%$, is approximately 31000 steps, which matches the observed average of 30903 shown in Figure 2b and Table IV. A locally optimal controller which controls every node in the network, on the other hand, should be able to balance the network indefinitely within any $\theta_G > 0\%$. Given that the Random controller is not locally optimal, the trade-off of uncertainty versus the precision required to use the narrower θ_C threshold does not appear to be a beneficial one.

TABLE II
SAMPLE RESULTS AROUND THE REQUIRED BUDGET FOR A GIVEN INSTANCE TO BECOME DIFFUSIONLESS. THE INSTANCE USED HERE IS RG.N100.D5.I1, $\theta_G = 0.5$, $B^* = 306$, $5\% = 30 \approx 5$ NODES.

	Budget B^*-10%	B^*-5%	B^*	B^*+5%	B^*+10%
Null	29	29	29	29	29
Rand.	2952	7584	21574	56019	128068
Anti	478077	475092	445030	350479	90768
ANN	170225	382270	360314	377192	400428
EANN	442508	472207	483721	492958	494277

TABLE III
SAMPLE RESULTS AROUND THE REQUIRED BUDGET FOR A GIVEN INSTANCE TO BECOME DIFFUSIONLESS. THE INSTANCE USED HERE IS PA.N100.D2.I3, $\theta_G = 0.3$, $B^* = 355$, $5\% = 25$.

	Budget B^*-10%	B^*-5%	B^*	B^*+5%	B^*+10%
Null	5	5	5	5	5
Rand.	71	105	153	225	313
Anti	20	12	8	6	5
ANN	26997	46995	219890	275770	383371
EANN	442508	472207	483721	492958	494276

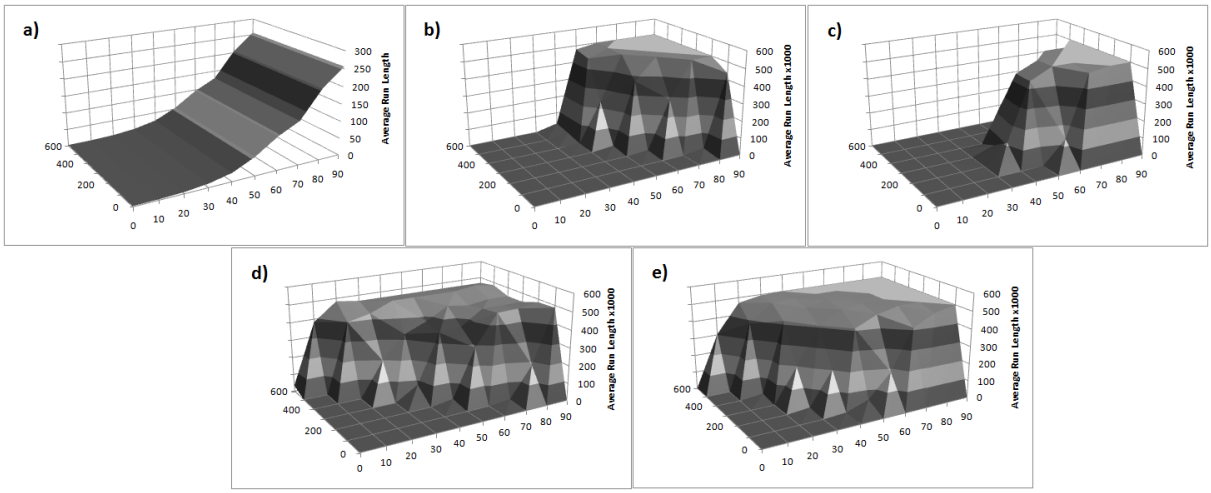

Fig. 2. Surface plots of run lengths achieved over varying θ_G and B arguments for a single network structure (rg.n100.d5.i0). a) Null controller b) Random controller c) Anti-majority controller d) ANN controller e) EANN controller

TABLE IV

CROSS-SECTION OF RESULTS OVER A RANGE OF BUDGET VALUES ON 5 NETWORK STRUCTURES WITH θ_G HELD CONSTANT AT 0.4. BUDGETS ARE NORMALIZED AS PERCENTAGE OF TOTAL COST TO CONTROL THE ENTIRE NETWORK. VALUES ARE AVERAGED OVER 30 RANDOM CONFIGURATIONS AND 100 TESTS PER CONFIGURATION, AND ROUNDED TO THE NEAREST INTEGER.

	Budget									
rg.n100.d5.i0	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Null	24	24	24	24	24	24	24	24	24	24
Random	26	39	71	179	548	1915	5991	14812	25692	30903
Anti	42	388	42635	129351	44317	1140	86	41	33	133
ANN	24	12	13	114	55176	347754	359378	466668	500000	500000
EANN	361	21039	152017	352653	409719	470303	490741	491696	499614	500000
rg.n100.d5.i1										
Null	17	17	17	17	17	17	17	17	17	17
Random	23	37	79	205	653	2199	6699	15969	26764	30903
Anti	42	591	99239	246592	78971	1074	70	31	22	32
ANN	9	16	4	358	84788	386584	386694	416668	450000	483334
EANN	38	432	11954	277602	428488	475693	489003	496535	499428	500000
pa.n100.d2.i0										
Null	8	8	8	8	8	8	8	8	8	8
Random	11	18	31	74	230	994	3581	12126	24544	30903
Anti	18	139	5109	53132	6499	253	22	8	4	3
ANN	10	9	7	11	4	199803	366668	483334	466668	483334
EANN	12	22	90	4890	211226	443076	481259	499879	498400	499851
pa.n100.d2.i1										
Null	7	7	7	7	7	7	7	7	7	7
Random	11	16	29	64	212	933	3137	10893	24783	30903
Anti	18	101	9602	35104	6188	203	21	8	5	4
ANN	9	7	4	4	15713	225462	400000	433334	450000	500000
EANN	11	20	149	3143	189580	432330	488528	494630	496568	500000
pa.n100.d1.i0										
Null	18	18	18	18	18	18	18	18	18	18
Random	23	33	58	126	384	1350	4390	12554	24820	30903
Anti	32	93	1393	7512	5464	558	102	36	20	16
ANN	21	17	25	44	32777	305173	425342	416668	466668	483334
EANN	58	301	20773	81204	401615	446979	491919	498428	499052	500000

The remaining intelligent controllers each outperform the Random and Null controllers overall. Table IV presents some sampled results for a number of different instances across successive budget values using a constant $\theta_G = 40\%$. This table exemplifies that these intelligent controllers (with some exceptions discussed below) each improve the quality of results for given instances. This same trend is observable in Figures 2c, d, an e, where the ridge of higher quality solutions is pushed successively towards the origin (θ_G=0,B=0) when compared to the Random controller. This can be said to be the goal of study of the θ-CAP, that is, to find control system implementations which further this trend.

The Anti-majority controller achieves high quality results on instances of lower budget. Despite its lack of adaptation through any learning mechanism, it is capable of leveraging minimal amounts of control to statistically significant increases in performance as compared to the Random controller. A sample Mann-Whitney UTest using a 5% level of significance on the first set of results from Table IV over budgets 10%, 20%, and 30% yield z-scores of 6.6, 6.7, and 6.7, respectively; P<0.0001 in all cases. The Anti-majority controller experiences difficulty, however, as the budget grows past 50% control of the network. Intuitively this makes sense, as the controlled set (V_C) becomes the majority, the observed node states that the controller is opposing in its decision making, are more likely to be from its own output. Controlled cliques of initially diverse node states will all be set to the same minority state in the next time step, which then becomes the majority both within the clique, and its nearby neighbours. This tail-chasing behaviour leads to results that decline in performance above the B=40% mark for all instances. Real world applications of automated control, however, are not likely to have such large amounts of dictation over the network state, and so the Anti-majority controller may be a suitable heuristic alternative to the more complicated learning controllers. One final point to the credit of the Anti-majority controller is its fast runtime as seen in Table V. In addition to fast individual runtimes, the lack of any training requirement decreases the overall runtime substantially.

TABLE V
COMPARISON OF CONTROLLERS' AVERAGE RUN TIME ON THE RG.N100.D5.B300.T90.I0 INSTANCE, FOR WHICH ALL BUT THE NULL CONTROLLER REACHED SIMULATION CAP ON ALL RUNS. IGNORES COSTS ASSOCIATED WITH RESETTING THE NETWORK BETWEEN SIMULATIONS AND WEIGHT ADJUSTMENTS IN THE TRAINED CONTROLLERS. ASSUMES NO EARLY TERMINATION OF TRAINING.

	Single	Aggregated
Null	1.09ms	x * 100 = 109ms
Rand.	2s853ms	x * 100 = 4m45s
Anti	2s513ms	x * 100 = 4m11s
ANN	8s450ms	x * 200 = 28m10s
EANN	8s101ms	x * 100^2 + 200 = 22h57m

The ANN controller shows the opposite results over varying budgets as compared to the Anti-majority controller. Where the Anti-majority controller does comparatively well using lower budgets, the ANN controller shows very poor results. Using a large budget, however, results in reasonably good solutions using ANN. Table III shows that the quality of the ANN controller's solutions improve dramatically near and above the B^* diffusionless budget level. However, Table II illustrates that the ANN controller succeeds in pushing the higher quality solutions into the more difficult instances. Table IV allows for a slightly broader perspective on this issue, where B^* corresponds approximately to the 60% column. Here the exponential increase in run length can be seen to begin as early as 10% before B^*.

Note, that in Figure 2, the ANN controller appears to fail to reach the simulation cap on instances for which the Random controller does so. The results presented are averaged over 30 separate control system configurations. Several of these configurations produce near-immediate failure for high budget and/or θ_G values resulting in a slight drop in quality when averaged with a large number of capped-off instances. This inconsistency does not appear as significantly in any of the other controllers, suggesting that it is a result of the use of gradient descent to train the network. The uncertainty in the state of the uncontrolled portion of the social network is likely the cause of gradient descent's difficulty. Gradient descent does best in the presence of a smooth error gradient that slopes towards the global optimum. Susceptibility to poor local optima is a known weakness of gradient descent learning, particularly when posed with highly irregular fitness landscapes[26].

The difficulty of training a neural network for the θ-CAP using gradient descent is clearly illustrated by the erratic training curve displayed in Figure 3. This presents the ANN controller training scores for an instance on which it achieved an average run length of ≈18000 in testing. A slight improvement is seen over the course of training, however, the erratic convergence is indicative of the unsuitability of gradient descent as applied to this problem. This difficulty was the original motivation for the use of evolutionary algorithms to train the ANN weights. Future work will also consider the use of an adaptive critic network for this purpose.

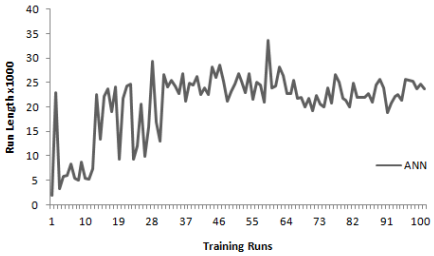

Fig. 3. Averaged learning curve of the ANN controller for the instance rg.n100.d5.b200.t50.i0.

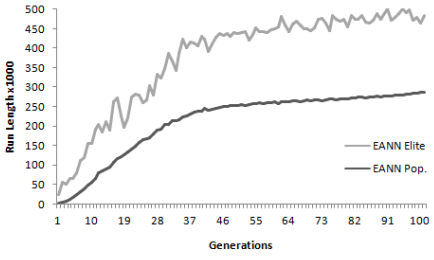

Fig. 4. Averaged population and elite fitness curves for the EANN controller on the instance rg.n100.d5.b200.t50.i0.

The use of a neural network representation for the controller behaviour component is validated by the results of the EANN controller. Although the ANN controller is capable of exhibiting high quality (albeit inconsistent) results for instances near and above the B^* line, the EANN consistently achieves comparatively high quality results over all instances considered.

On a number of instances with shorter path lengths (as estimated via the null driver) and small budgets ($B \leq 40\%$), the Anti-majority controller achieves results with greater average run lengths than the EANN. This is believed to be a result of the unoptimized EANN's inability to generate enough meaningful data from which to learn due to the short simulation times. Further experimentation is needed with optimized heuristics to state this definitively. The use of a single population over gradually decreasing θ_G values could also be a means of evolving an effective controller.

Additionally, the EANN is occasionally outperformed in instances for which the averaged values contain a majority of tests that reached the simulation cap. The simulation cap causes an omittance of information as it overrides the potential for greater solutions, making it difficult to draw conclusions using these instances alone. A more descriptive comparison is to consider a region throughout which one controller fails to reach the simulation cap, but the other successfully does so. This is the case throughout Table IV when comparing the EANN and Anti-majority controllers.

The high quality results of the EANN controller, however, come at the cost of very high run times. This is primarily due to the large population and generation limits used, requiring 10000 simulations in the training phase (ignoring the early stoping criteria).

The training phase of the EANN provides a much smoother learning curve as depicted in Figure 4. It should be noted that the elitism curve shown therein contains abnormal depressions in fitness over the course of training due to the practice of testing the elite individual at the end of every generation. An individual with inconsistent performance that is selected as the elite individual of a generation may report poor performance during the elitism test leading to the observed erratic curve during the training phase. These inconsistent individuals are expected to be out-performed in successive generations, leaving consistently high performing results obtained via the EA trained neural network representation.

To conclude the results, it should be remembered that the presented control systems were run without an performing an exhaustive optimization tuning for this problem. Despite this, each of the intelligent controllers outperformed the Random controller in some respect. The Anti-majority controller achieves high quality results on low budget instances with a fast run time. The ANN controller shows moderate improvements over the Random controller in or near the class of diffusionless instances at the cost of a slower run time and training phase. Finally, the EANN requires significant training time to converge to comparatively high quality results over the entire set of instances.

VI. Conclusion

The study of automated control of large-scale social networks is an important problem with wide-ranging applications. However, the present state of the literature is incomplete. This paper presents a formalization of the Network Control Problem and highlights the properties of several subproblems worthy of study, including the novel θ-Consensus Avoidance Problem. It is our intention to motivate research activity in this area.

This paper introduced the θ-Consensus Avoidance Problem; an NCP subproblem requiring a search for both a configuration and a behavioural mapping in the development of an optimal control system. Several algorithms were implemented as the control system's behavioural component, and the Evolutionary Artificial Neural Network controller substantially outperformed the other algorithms considered. However, it did so at the cost of substantial training time. Ongoing and future work will look at the potential to optimize the EANN's performance in terms of both quality and speed. The anti-majority heuristic will be considered as a means of improving the EANN by limiting the size of the ANN structure. This is expected to decrease runtimes while improving reliability of the resulting solutions.

A fitness landscape analysis for the θ-CAP will be used to explore the hypothesis that the ANN's poor learning performance is due to a highly modal landscape. An Adaptive critic implementation will be considered for its potential to ease this difficulty.

Additionally, a comparative study of configuration techniques will be performed. Existing heuristics, such as the greedy marginal gain selection of [12], [16], the optimal distribution maximum matching of [18], and the degree-proportional selection of [5], will each be considered for their applicability to the θ-CAP. Additionally, clustering [6] and evolutionary algorithms [10] will be considered.

Finally, the extensions to the θ-CAP may be investigated, such as indirect control, multiple controllers, or variations of the diffusion model used (e.g., linear threshold, probabilistic voter model, trust, similarity, and/or evidence based communication). Interesting questions may be address such as whether a node that is less likely to listen to a neighbour once they know that neighbour is being controlled is harder to influence, how such knowledge of control could arise in a network, or what steps can a population take to protect their opinions from external biased influence.

References

[1] C. Anderson. Learning to control an inverted pendulum using neural networks. *Control Systems Magazine, IEEE*, 9(3):31–37, 1989.

[2] S. Bharathi, D. Kempe, and M. Salek. Competitive influence maximization in social networks. *Lecture Notes in Computer Science*, 4858(Internet and Network Economics):306–311, 2007.

[3] T. Carnes, C. Nagarajan, S. Wild, and A. van Zuylen. Maximizing influence in a competitive social network: a follower's perspective. *Proceedings of the ninth international conference on Electronic commerce, ACM*, pages 351–360, 2007.

[4] A. Clauset, C. Shalizi, and M. Newman. Power-law distributions in empirical data. *SIAM review*, 51(4):661–703, 2009.

[5] R. Cohen, S. Havlin, and D. Ben-Avraham. Efficient Immunization Strategies for Computer Networks and Populations. *Physical review letters*, 91(24):1–5, 2003.

[6] F. Comellas and A. Miralles. A fast and efficient algorithm to identify clusters in networks. *Applied Mathematics and Computation*, 217(5):2007–2014, Nov. 2010.

[7] P. Domingos and M. Richardson. Mining the network value of customers. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 57–66, 2001.

[8] E. Even-Dar and A. Shapira. A note on maximizing the spread of influence in social networks. *Information Processing Letters*, 111(4):184–187, Jan. 2011.

[9] L. Fausett. *Fundamentals of Neural Networks*. Prentice-Hall Inc, 1994.

[10] Y. Gui-sheng, W. Ji-jie, D. Hong-bin, and L. Jia. Intelligent Viral Marketing Algorithm over Online Social Network. In *2011 Second International Conference on Networking and Distributed Computing (ICNDC)*, pages 319–323. Ieee, Sept. 2011.

[11] R. Holley and T. Liggett. Ergodic theorems for weakly interacting infinite systems and the voter model. *The annals of probability*, 3(4):643–663, 1975.

[12] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146, 2003.

[13] D. Kempe, J. Kleinberg, and E. Tardos. Influential nodes in a diffusion model for social networks. *Automata, languages, and programming*, pages 1127–1138, 2005.

[14] M. Kimura, K. Saito, R. Nakano, and H. Motoda. Extracting influential nodes on a social network for information diffusion. *Data Mining and Knowledge Discovery*, 20(1):70–97, Oct. 2010.

[15] T. Lappas, E. Terzi, D. Gunopulos, and H. Mannila. Finding effectors in social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '10*, pages 1059–1068, New York, New York, USA, 2010. ACM Press.

[16] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '07*, pages 420–429, New York, New York, USA, 2007. ACM Press.

[17] D. Liu. Adaptive critic designs for problems with known analytical form of cost function. In *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02*, pages 1808–1813. IEEE, 2002.

[18] Y. Liu, J. Slotine, and A. Barabási. Controllability of complex networks. *Nature*, pages 1–7, 2011.

[19] Y.-Y. Liu, J.-J. Slotine, and A.-L. Barabási. Controllability of complex networks. (Supplementary). *Nature*, 473(7346):167–73, May 2011.

[20] K. Mehrotra, C. Mohan, and S. Ranka. *Elements of artificial neural networks*. The MIT Press, 1996.

[21] D. Montana and L. Davis. Training Feedforward Neural Networks Using Genetic Algorithms. *IJCAI*, pages 762–767, 1989.

[22] R. Pastor-Satorras and A. Vespignani. Immunization of complex networks. *Physical Review E*, 65(3):1–9, 2002.

[23] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 61–70. ACM, 2002.

[24] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *IEEE International Conferance on Neural Networks*, pages 586–591. IEEE, 1993.

[25] S. J. Russell, P. Norvig, J. F. Canny, J. M. Malik, and D. D. Edwards. *Artificial intelligence: a modern approach*, volume 74. Prentice hall Englewood Cliffs, 1995.

[26] M. Siddique and M. Tokhi. Training neural networks: backpropagation vs. genetic algorithms. *Proceedings of International Joint Conference on Neural Networks, IJCNN'01*, 4:2673–2678, 2001.

[27] W. Wang, Y. Lai, and J. Ren. Controllability of complex networks with nonlinear dynamics. 2011.