Evolving Neurocontrollers for the Control of Information Diffusion in Social Networks

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

***Abstract*—Automated control of information diffusion in social networks is a difficult problem with potential application to marketing, security, and social organization. The θ-Consensus Avoidance Problem (θ-CAP) is a social network control problem modelling the specific case of maintaining a network within a homoeostatic range of states. It is an important problem because it represents social networks avoiding extreme views or situations that might lead to extreme behaviour.**

This paper presents a comparison of two Evolutionary Artificial Neural Network (EANN) variants acting in the behavioural component of an autonomous control system for instances of the θ-CAP. A novel variant of EANN is proposed by adopting characteristics of a well-performing heuristic into the structural bias of the neurocontroller. Information theoretic landscape measures are used to analyse the problem space as well as variants of the EANN.

The results obtained indicate that the two neurocontroller variants learn very distinct strategies for balancing the states of the social network, however, the newly proposed variant demonstrates improvements in both solution quality and execution time. A ramped-difficulty evolution scheme using constraint parameters is demonstrated to be effective at creating higher quality results as compared to the standard scheme for EANNs. A correlation between the proposed instance difficulty and identifiable landscape characteristics is discovered as well.

I. Introduction

Social networks are an important part of the world in which we find ourselves. Networked interactions of individual agents coalesce to form anything from biological entities (e.g., multi-cellular organisms) to complex social structures (e.g., gangs or political parties). Intentional use of these networks ranges in application from advertising to counter-terrorism, but consistently places a great deal of importance on their stability and predictability [2], [5]. Identifying network states of high risk, as well as guiding the network back to more desirables ones, are therefore essential components of our ability to use and control the social networks around us.

The Network Control Problem (NCP) describes a social network as a combination of structural and behavioural components, known as the network and the diffusion model respectively [30]. The NCP then defines a pair of dependant optimization problems in terms of discovering both the structural and behavioural components of a control system. The objective of the control system is to direct the state of the social network with a high degree of precision via interactions at the node-state level. The amount of direct influence a control system has on the social network is constrained by a budget placed on the size of the set of controlled nodes. Thus the control system is required to determine both an optimal configuration of connections to the social network, and an optimal state-to-signal mapping for the control system's ongoing output.

The NCP, in the general case, is a formalization of a set of related subproblems. In all subproblems the objective remains to control the state of a network, however the evaluation of specific states varies by problem. In the well-known case of the Influence Maximization Problem (IMP) [5], [28], [14], [15], [3], [17], [19], the goal is to maximize the spread of a single selected node state throughout the network. This implies that the control system's behaviour component for the IMP is rigidly defined (outputting a single state), and so the search is for an optimal structural configuration exclusively. Similarly, in [22], the problem of structural controllability is examined as a search for the minimum number of control sites and optimal placement thereof, without the associated search of an appropriate behavioural component for the control system. Solutions to these problems offer insight to the optimization of the configuration of the NCP, but not the behavioural component.

In [30], the θ-Consensus Avoidance Problem (θ-CAP) is defined as an NCP subproblem with both configuration and behavioural search requirements. In addition to considering the number and location of control sites on the network required to deliver an appropriate control signal, the θ-CAP considers the dependent search for a state-to-signal mapping capable of maintaining an equilibrium of node states within the network. This paper extends the previous work to consider the optimization of the behavioural mapping for the θ-CAP, with the control configurations determined randomly.

To optimize the θ-CAP behavioural component, two variants of the Evolutionary Artificial Neural Network (EANN) metaheuristic are applied. The input to output mapping structure of a feed forward neural network is a natural description for the state-to-signal mapping required for network control. Additionally, the universal function approximation characteristic of ANNs [9] is practical for exploring this unknown search space. In place of traditional gradient descent techniques for the training of the neural network weights, this paper applies Evolutionary Algorithms (EA). Evolution-based search techniques have been shown to effectively search difficult problem spaces, and are less prone to getting trapped at sub-optima in multi-modal search spaces, particularly when supervised training information is not available or costly to obtain [30], [44].

In addition to establishing new benchmark results for the

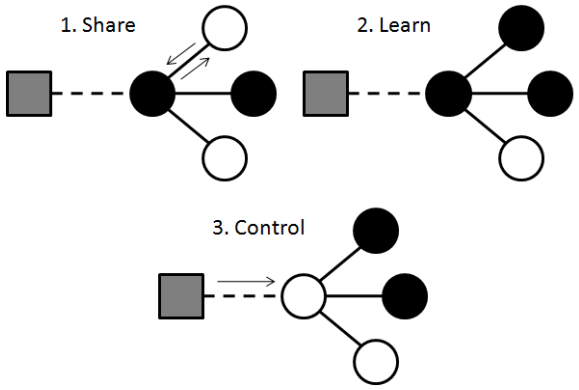

Fig. 1. Phases of the θ-CAP execution. 1. State information is shared between neighbouring nodes in the network. 2. Nodes adopt shared information into their own states. 3. The control system applies its control signals to the controlled node(s).

behavioural component of the θ-CAP, this paper compares the implemented neurocontrollers in terms of their relative fitness, speed, and their evolved strategies of control. Landscape analysis is applied to distinguish characteristic differences in the search spaces explored by either algorithm variant, as well as over varying ranges of difficulty parameters.

The remainder of this paper is structured as follows: Section II provides background on the θ-CAP, EANNs, and fitness landscape measures, Section III describes the experimental approach taken here, Section IV presents the results of this paper, and finally Section V concludes this paper, providing insight into future work.

II. Background

A. θ-Consensus Avoidance Problem

The θ-Consensus Avoidance Problem (θ-CAP) is a homeostasis-type NCP defined in [30]. It is based on the canonical pole-balancing problem, in which a state-to-action mapping is required to determine the appropriate velocity of a cart affixed with a hinged pole in order to maintain a vertical orientation in the pole for as long as possible [1], [10], [11], [20], [29]. It should be noted that the search objective for the pole-balancing problem is strictly for a behavioural component, as the control system configuration is specified in the problem itself.

The objective for the control system in the θ-CAP is to maintain the state of the social network within a bounded range of equilibrium. Given a graph $G = (V, E)$ of nodes V connected by edges E (including self loops), each node $v \in V$ contains a single discrete memoryless state $k \in \{0, ..., (K-1)\}$ (here $K = 2$) initialized randomly. A control system C is connected to G by selecting a set $V_C \subseteq V$ of nodes to control, subject to a budgetary constraint and an associated cost measure per node.:

$$\sum_{v \in V_C} cost(v) \leq B \tag{1}$$

where V_C is the set of nodes in the social network controlled by the control system C, B is the budgetary parameter for the given problem instance, and $cost(v)$ is a problem-specific node cost function that associates a cost to each node in the network. Here, as in [30], this cost function is equal to the node's degree.

Simulation then proceeds in three main phases as seen in Figure 1. In phase 1, each node shares information regarding its state along all outbound edges. In phase 2, each node adopts a new state based on the information it received in phase 1. In phase 3, the control system applies its control signal, directly setting the state of all nodes in the controlled set. Following this, the state of the social network is evaluated. This process repeats until the network exceeds a consensus threshold parameter, θ_G.

Formulated as a search problem, the goal of the θ-CAP is to find a time series of control signals, or equivalently a state-to-signal mapping, that will minimize the following network state utility function over an infinite time span:

$$U(t) = \begin{cases} 0, & if \ \frac{||V^1(t)| - |V^0(t)||}{|V|} \leq \theta_G \\ 1, & otherwise \end{cases} \tag{2}$$

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. The fitness, f, of a θ-CAP solution is the number of time-steps that the control system maintains the network within the first case of Equation (2).

Phases 1 and 2 of the θ-CAP correspond to the operation of a diffusion model, which is the behavioural component of the network used to govern the interactions between individual nodes. In order to both simplify and make explicit the challenge posed to the control system this paper considers the Voter Model (VM) [8], [6]. The VM is a simple diffusion model with a guarantee to converge to consensus (if left uncontrolled) in O(n^5) time [6]. Specifically, a node randomly adopts the state of one of its neighbouring nodes at each time step as follows:

$$Sh(v) := s(v, t) \tag{3}$$

$$L(v, I) := \begin{cases} 1, \ with \ probability \ \frac{|\{s=1|(w,s,u) \in I(v,t)\}|}{|I(v,t)|} \\ 0, \ with \ probability \ \frac{|\{s=0|(w,s,u) \in I(v,t)\}|}{|I(v,t)|} \end{cases} \tag{4}$$

where, $Sh(v)$ is the sharing strategy corresponding here to $s(v, t)$ the state of node v at time t, $L(v, I)$ is the learning strategy of node v based on the input information token I. $I(v, t)$ is the set of information tokens received at node v at time t paired with the weights of all edges along with the information travelled and the index of the source node, u. This paper considers all edges to be unweighted (i.e., $w_{i,j} = 1$).

Initial solutions to the θ-CAP *behavioural* search problem were studied in [30]. The configuration component was left unoptimized, using instead a random selection of nodes until the given budget was exhausted. No attempt was made to optimize parameters or algorithm design for the five control system heuristics described therein. The use of a neural network structure was found effective; however, the gradient descent technique was considered unsuccessful when compared to the

EANN and Anti-Majority (AM) control systems. The AM is a heuristic-based controller that outputs signals that correspond to the least-represented state in the controlled nodes' immediate neighbourhoods. It was found successful at low budget, but suffered poor performance when the problem instance budget gave the control system a majority in the social network. The EANN that was used consisted of a fully-connected feed-forward network using the mutate-nodes operation from [25]. Testing results found the AM and EANN control systems to be competitive over certain ranges of problem instance parameters, but the EANN suffered from a comparatively very high computational cost for training.

Algorithm 1 Pseudo-code of the EANN algorithm with a single individual elitism, used to evolve ANN weights.

Require: Pop is the initial population of weight arrays
1: **procedure** TRAINEANN($instance$, $maxGen$, Pop)
2: $gen \leftarrow 1$
3: **while** not Termination(gen, $maxGen$, Pop) **do**
4: Evaluate(Decode(Pop),$instance$)
5: $best \leftarrow$ maxFitness(Pop)
6: $Pop \leftarrow$ Select(Pop, $best$)
7: $Pop \leftarrow$ Mutate(Pop)
8: $gen \leftarrow gen + 1$
9: **return** Pop

B. Evolutionary Artificial Neural Networks

Artificial Neural Networks (ANNs) are a universal function approximation meta-heuristic modelled after the functioning of the brain. For a full description of this meta-heuristic see [31], [24]. In brief, a feed-forward ANN consists of an input-to-output mapping represented as a layered structure of nodes, called neurons, connected by directed and weighted edges, called axons. Values are set at the input layer of neurons, then fed-forward to a hidden layer. All values entering a given neuron are combined using a weighted sum and then a threshold function (such as a sigmoid function) is applied to generate the output fed forward to the next layer.

Training a neural network consists of adjusting the weights of each axon, typically in a supervised context, to reduce the observed error from an expected output. A common approach for this adjustment is the use of gradient descent techniques, most notably back-propagation [42]. Gradient descent techniques are limited to situations in which a differentiable error gradient exists, and are prone to convergence to local optima [44]. This makes these techniques less attractive in situations for which error information is not known or costly to compute, as is the case for the θ-CAP, since the optimal control signals are not known in advance. An attempt was made in [30] to estimate the error based on relative changes in output and fitness values, but the resulting ANN showed erratic learning.

Evolutionary Algorithms (EAs) are a potentially unsupervised, population-based, stochastic search meta-heuristic inspired by the concepts of biological evolution and survival of the fittest. For a full description of EAs see [31], [23]. In brief, a population of potential solutions is initialized randomly and each evaluated. Individuals are selected stochastically for the next generation based on their relative fitness among the current population, and are subsequently combined and/or mutated to form the next generation. This process repeats until a solution of sufficient quality has been found or until the population ceases to improve over time.

Evolutionary Artificial Neural Networks (EANNs) perform a search for an approximately optimal neural network without requiring gradient or supervisory information [25], [44], [45]. EANNs have been used to optimize weights, architecture, and learning rules, but this paper considers only the former. Here, a population of weight arrays is maintained and each is evaluated according to the resulting network's performance on some task (e.g., balancing the states in a social network). Representation of an ANN in the EA population as a list of real values has been demonstrated to be effective [25], [45]. This allows standard real-valued genetic operators to be used; however, [25] present several crossover and mutation operators that take ANN-specific features into account.

In comparison to standard back-propagation, EA is claimed to be a more robust search operator with improved scalability and decreased likelihood of being trapped in local optima [34], [44]. Experimental comparisons on various function approximation and classification test problems demonstrate consistently better performance in terms of both speed and accuracy using EA over BP [25], [33], [34], [45]. Successful applications of EANN to various control problems have been reported [18], [21], [26], including variations of the relevant pole balancing problem [7], [11], [13], [32], [43], [39].

Difficulty was noted in problems of fine-tuning very small numbers of weights, which is a task better suited to gradient descent techniques. The permutation problem, in which the change in location of a neuron in any given layer of a fully connected ANN does not alter its functionality, has been identified as challenge for training EANNs [25], [45], [16]. It was suggested that the use of mutation as the primary genetic operator could reduce this issue [45], [16]. That is the approach taken in this paper.

C. Fitness Landscape Analysis

The fitness space of a problem is the set of evaluations of all feasible solutions, ordered according to some neighbourhood function. Fitness landscape analysis considers samples of this space taken as random walks, in which a single solution is evaluated and then altered according to a search operator (e.g., a mutation operator) to obtain a neighbouring solution. This process is repeated until a walk of sufficient length is acquired.

Measurements are performed on the sampled walks to determine features of the landscape, with a goal of predicting the difficulty of a given problem or explaining the observed performance [27], [40]. In effect, fitness landscape analysis is used to discern whether the neighbourhood function of two compared samples differs, and to present a characteristic description of their differences that may offer insight to their relative difficulty.

A local optimum is a solution which has a greater fitness than all of its neighbours. The epistasis of a problem is the

level of inter-dependency of genes within the solution. A landscape with many optima or a high epistasis is said to be rugged, while one with fewer optima is relatively smooth [35], [37]. An attractive basin is a series of solutions in the region of a landscape surrounding a local optimum which form a progressively improving gradient of fitness as it approaches the optimum itself. Small attractive basins are considered isolated, while larger attractive basins are considered to make the overall landscape smoother and thus easier to search [12]. A plateau is a series of solutions which have a difference of fitness within some specified sensitivity parameter, ϵ. Here, $\epsilon = 0$ indicating the most sensitive measurement as in [40].

A number of information theoretic landscape measurements, as proposed in [38], are used in this paper. It is shown in [40] that these measures correlate with the performance of EA using various mutation and crossover operators. To allow the application of landscape analysis measures, the fitness space is commonly assumed to be statistically isotropic [41], [12], [38], this is the case in related work with EANNs as well [36], [37], [39]. For a detailed review of a variety of fitness landscape measures see [27]. The remainder of this section defines the specific measurements used in this paper.

The *Information Content* (IC) characterizes the variety and frequency of transitions along the random walk. A lower IC corresponds to more consistency among transitions which likely implies a less rugged landscape. The measure is calculated as:

$$IC(\epsilon) := -\sum_{p \neq q} P_{[pq]} \cdot \log_6(P_{[pq]}) \tag{5}$$

where $P_{[pq]}$ is the observed probability of transitioning from state p to state q in the random walk ensemble Ψ:

$$\Psi(i,\epsilon) = \begin{cases} \bar{1}, & \textit{if } f_i - f_{i-1} < -\epsilon \\ 0, & \textit{if } |f_i - f_{i-1}| \leq \epsilon \\ 1, & \textit{if } f_i - f_{i-1} > \epsilon \end{cases} \tag{6}$$

where f_i is the fitness observed in step i of the random walk, and ϵ is a sensitivity parameter used to classify steps in the random walk as increasing, decreasing, or plateaued.

The *Partial Information Content* (PIC) measures the modality of the landscape as the ratio of peaks and valleys in the walk to the length of the entire walk itself. Large values here indicate a highly modal landscape while smaller values suggest a flat or smooth landscape.

$$PIC(\epsilon) := \frac{\mu}{n} \tag{7}$$

where n is the length of the walk and μ is the result of the summarized walk, $\mu = \Phi(1,0,0)$ computed using the following recursion:

$$\Phi(i,j,k) = \begin{cases} k & \textit{if } i > n \\ \Phi(i+1,i,k+1) & \textit{if } j = 0,\ \Psi(i) \neq 0 \\ \Phi(i+1,i,k+1) & \textit{if } j > 0,\ \Psi(i) \neq 0, \\ & \quad \text{and } \Psi(i) \neq \Psi(j) \\ \Phi(i+1,j,k) & \textit{otherwise} \end{cases} \tag{8}$$

which has the effect of counting all of the transitions from a decreasing walk to an increasing walk and vice versa while excluding plateaus and smooth edges.

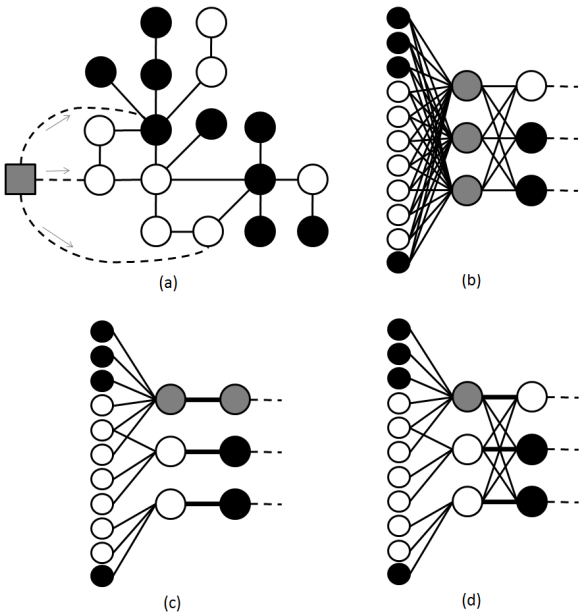

Fig. 2. Connectionist algorithms for control of a social network. a) A sample social network with 3 controlled nodes b) Standard EANN. c) CAM control system. d) ECAM control system.

Finally, the *Density Basin Information* (DBI) characterizes the smoothness of the landscape using an entropic measure of the regions of the ensemble which do not transition.

$$DBI(\epsilon) := -\sum_{p \in \{\bar{1},0,1\}} P_{[pp]} \cdot log_3(P_{[pp]}) \tag{9}$$

where $P_{[pp]}$ represents the observed probability of two consecutive elements in the ensemble Ψ having the same state. Landscapes with large basins of attraction or plateaus will have a higher DBI.

III. Algorithm Design

This section discusses the design and implementation details of two variants of EANN algorithm that are used in the experiments of Section IV. Note that this paper is concerned exclusively with the optimization of the behaviour component of the θ-CAP control system. The configuration component uses a seeded random selection in all considered experiments of this paper.

Results from [30] demonstrate that an EANN control system is capable of consistently learning network balancing behaviour over a variety of random control system configurations and network types. Additionally, the Anti-Majority (AM) control system showed results competitive to the EANN over select problem instances of lower budget.

The AM heuristic can be modelled as a connectionist architecture with fixed weights as shown in Figure 2(c). This form of the AM is termed the Connectionist Anti-Majority (CAM) heuristic. The CAM control system requires no training, and computes an identical input-to-signal mapping as the standard AM control system. The number of input nodes is set to the number of unique observable nodes in the social network ($|N_{V_C}|$). The number of hidden and output nodes

are both set to the number of controlled nodes ($|V_C|$). Axons between the input layer and hidden layer are added if and only if there exists an edge in the social network between the observed node and the controlled node. Inputs to the CAM control system are scaled and shifted to the range [-1,1] (from the VM-defined social network states as discrete values in [0,1]). Weights on axons between the input and hidden layer are fixed at 1, and the hidden layer neurons use a tangent sigmoid activation function to output a positive or negative value corresponding to which of the two states was more prevalent in the neuron's input. Thus the hidden layer neurons encode an observed majority feature for each controlled node. The hidden layer connects in a 1-to-1 manner to the output layer. Weights on the hidden-to-output axons are fixed at -1 to produce an anti-majority value at each output, which are then processed through a logistic sigmoid activation function to produce a value in the required [0,1] range. Values are rounded to produce a discrete value for the resulting control signal, with values of 0.5 (corresponding to an equal proportion of states as input) given a random signal value as in the original AM heuristic.

The CAM control system performs identically to the AM: fast, competitive results for low-budget instances, but severe decrease in performance with high budgets. The EANN control system on the other hand achieves consistently high-quality results at the cost of a slow training phase. This paper proposes the Evolutionary Connectionist Anti-Majority (ECAM) control system as a middle ground between the prior two approaches. This is achieved by reducing the number of axons in the EANN scheme which require settable weights based on the fixed weights in the CAM scheme, as seen in Figure 2(d). Axons between the input and hidden layers of the ECAM are configured as in the CAM structure, and weights are fixed at 1.0. The hidden-to-output layer is fully-connected and adjustable; however, it was found during experimental tuning that initializing the 'horizontal' axons (bold in Fig. 2(d)) to -1 as in the CAM scheme, but still allowing them to be adjusted through the EA training, improved the quality of results. All other axons are initialized randomly in the standard [-0.5,0.5) range.

Unless otherwise specified, the EA system used throughout experimentation consists of 100 individuals over 100 generations, with an elitism of a single individual. Due to variance in fitness for any given solution (individual), the elitism individual is tested once at the end of each generation. Early termination occurs if the population converges to a standard deviation of fitness less than 0.0001, or if the elitism individual achieves the same testing fitness for 5 consecutive generations. Selection is performed using a best-of-3 tournament to maintain relatively high diversity despite the small population size. The search operator used is the outbound variant of the *mutate-nodes* operator defined in [25], in which neurons are selected at random from the given network and all weights on axons originating from the selected neuron are given a small random shift (U[-0.5,0.5]). EANN uses a mutation rate of 0.1, while ECAM was found to improve with a higher rate of 0.4 (likely due to the bias of its initialization). Additional mutation operators were tested but changes in results were found to not be statistically significant. The number of hidden layer neurons for both EANN and ECAM were set to equal the number of output layer neurons, specifically $|V_C|$. Once the training is complete, the final elitism individual is tested over 100 independent simulations.

Algorithm 2 Pseudo-code of the Ramped evolution EANN for the θ-CAP.

Require: $instance$ is a tuple (θ_G, B, G)

procedure RAMPEDEANN($instance$, $maxGen$)
  $instance.\theta_G \leftarrow 0.9$
  $Pop \leftarrow$ InitPop($instance$)
  while $instance.\theta_G \geq 0$ **do**
    $Pop \leftarrow$ TrainEANN($instance$, $maxGen$, Pop)
    Report(Pop)
    $instance.\theta_G \leftarrow instance.\theta_G - 0.1$

In addition to the standard EA scheme, this paper presents a ramped difficulty evolution, in which the population begins by initially evolving to solve a problem instance with a high θ_G (=90%). The termination criteria of the standard EA are extended to include a single individual in the population reaching the simulation step limit (500000 steps). As with the standard evolution scheme an upper limit of 100 generations is used. Once the evolution is complete, the difficulty of the problem is increased by lowering the value of θ_G by 10% and the evolution is begun again. The population is evolved continuously through each level, so the population that reached the limit condition (step- or generation-) for one value of θ_G is the initial population for the next. This process is described in Algorithm 2.

The time required to evolve a solution to a given problem instance is proportional to the quality of the found solution; the longer a control system can balance the network, the longer the simulation will run. A number of early stopping criterion, described above, are aimed at reducing long series of simulations which routinely reach the simulation run limit. However, these only limit testing of instances which consistently do so. Instances which come close, but are not consistent will still require a considerable amount of execution time. Real world applications of this problem are likely to be interested only in control of a network using a minority of the nodes in that network, likely a very small minority at that. Thus, consideration of budget values that are greater than 50% require a great number of lengthy executions that are ultimately uninteresting and unnecessary. This compels the decision to narrow the focus of experimentation to a limited area of interesting problem instances. Throughout experimentation in Section IV, the main region of the parameter space considered is that which resides below the diffusionless threshold (see [30]), and uses a minority set of controlled nodes ($B < 50\%$).

One final design note, the fitness of repeated testing results of any single solution to the θ-CAP is observed to form a right-censored exponential distribution. While a corrected average can readily be calculated to present a more accurate view of the expected testing results for any given solution, it is less informative for purposes of understanding the problem-fitness space that each learning algorithm is faced with. So, for the

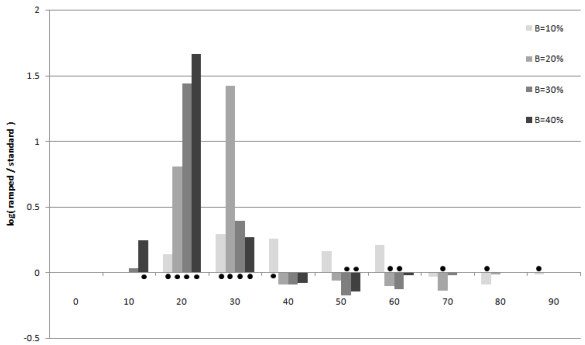

Fig. 3. Comparison of ramped evolution versus standard evolution using an EANN control system. Values presented are condensed logarithmically for scale. Comparisons with a statistically significant difference of means are marked with a dot.

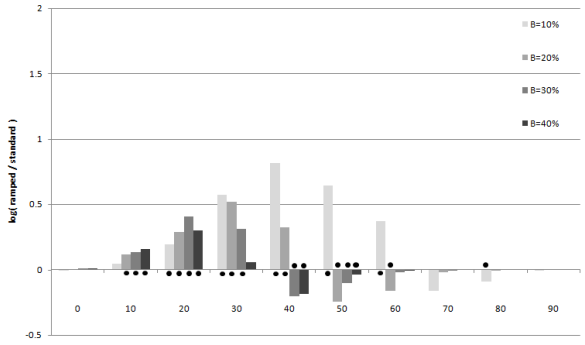

Fig. 4. Comparison of ramped evolution versus standard evolution using an ECAM control system. Values presented are condensed logarithmically for scale. Comparisons with a statistically significant difference of means are marked with a dot.

purposes of this comparative study, in which both considered algorithms are faced with the same censored information, there is no tangible benefit to reporting corrected averages (which may approach infinite simulation length).

IV. Experimental Results

The experimentation performed in this paper is aimed towards both comparing and understanding the execution quality of EANN algorithms acting in the behavioural component of an autonomous control system for the θ-CAP. In brief, the experiments considered here are as follows:

- comparison of a ramped evolution scheme for training of EANNs versus the standard fixed-parameter scheme,
- comparison of EANN and ECAM algorithms in terms of solution fitness, control signal characteristics, and execution speed, and
- analysis of fitness landscape characteristics relative to changes in both problem instance parameters and control system implementation.

A. Ramped Evolution

The EANN results of [30] are considered using independent evolutions for each combination of the problem instance parameters B and θ_G. There, poor results are observed for low consensus threshold values where the problem difficulty is greater. Two potential causes for this are: a) There is a limit to how narrow the constraints to the problem can be before the solution space is unexploitable, in that no solution is better than any other, or b) The execution time is too short to allow for the sufficient training of better solutions to these instances. The ramped evolutionary scheme is considered in order to distinguish between these two cases. Using this scheme, each evolved population will have been given considerably more training time prior to the final testing for the lower values of θ_G.

1) Does a ramped difficulty evolution improve the quality of the resulting neurocontroller?: The ramped evolution scheme is compared against the standard variant over budgets from 10% to 40% for both EANN and ECAM algorithms, and are presented in Figures 3 and 4 respectively. The best solution found per parameter set throughout either evolution is tested over 100 independent simulations. A Wilcoxon Signed-Rank test paired by control system configuration seeds with a 99% confidence interval, was used to determine if the observed improvements in test results were significant in terms of improved solution quality. Results showing statistically significant improvement over their counterpart are indicated as dots along the x-axis in Figures 3 and 4. Dots below or above the x-axis indicate instances for which the ramped evolution performs significantly better or worse respectively on average. There it can be readily seen that the ramped evolutionary solutions exhibit improved results as compared to their independently evolved counterparts for low values of θ_G. A lack of improvement over the $\theta_G = 0$ instances suggests that this level of constraint may be too severe to solve effectively with low budget values, however the improved performance over the 10-30 range suggests that the earlier hypothesis of additional time requirements is a feasible one. Notably, the ramped evolution fails to outperform the standard evolution at higher θ_G values. This trend is presumed due to the single-limit early stopping condition used in the ramped evolution, as compared to the early stopping occurring after five consecutive limits in the standard evolution. This was done in order to hasten the execution time of the ramped evolution over the set of all levels of θ_G, as well as to avoid the potential of overfitting the population prior to resuming evolution for an altered problem.

Overall the ramped evolution outperforms the standard evolution more consistently as measured by the frequency of instances for which statistically significant improved results are obtained. For the EANN algorithm ramped evolution outperforms standard at a ratio of 10:7:19 (ramped : standard : neither) over the 36 considered instances. For ECAM the ratio is 14:7:15. Many of the instances for which the ramped approach is outperformed could likely be improved if the more lengthy standard early stopping criteria are observed (at a loss of speed), since it is the only practical difference between the populations at high values of θ_G. The results of this comparison show the effectiveness of the ramped evolution in training control systems that are capable of preventing convergence for greater time-spans on average for low values

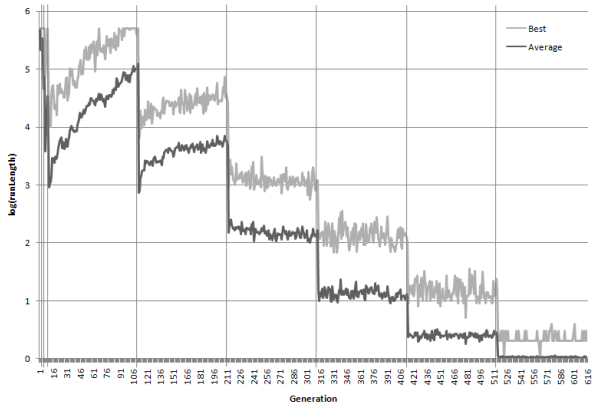

Fig. 5. Typical learning curve of a ramped evolution. Vertical lines indicate a decrease in θ_G.

of θ_G. The diminished performance at high levels of θ_G are seen as acceptable losses that do not prevent the development of higher quality solutions for the lower, more difficult values.

2) Why does the ramped approach outperform the standard evolution?: Interestingly, the weights of the final population for low consensus threshold perform poorly when tested on higher and thus easier values. For example, the best weights evolved for the $\theta_G = 20\%$ portion of a ramped evolution achieve much lower testing results on the $\theta_G = 70\%$ instance than the weights evolved for that instance specifically. This implies that the intuition which prompted the use of ramped evolution, namely that lowering θ_G simply creates a narrower constraint around the same optimal areas of the solution space, is unsupported. This is further corroborated by Figure 5, which shows a typical learning curve for a ramped evolution. The sharp decreases in fitness correspond to decreases in θ_G, followed by increases in fitness where possible, which indicates convergence towards new local optima. Despite this pattern, the quality of solutions is still improved for lower values of θ_G as compared to the standard evolutionary scheme.

If selection pressure is too high in independent standard executions of small θ_G values, leading to comparatively poor results (premature convergence to local sub-optima) then the benefit of ramped evolution may come from pre-conditioning the population into potentially fit regions along the low-pressure fitness surface within large θ_G instances prior to narrowing the constraint. This would explain both the shape of the convergence curve, as well as the improved testing results versus standard evolution, but not the ineptitude of the best solutions of narrow problems when applied to wider ones. It was considered that the improved testing results may have simply arose from the larger range of potential values for ANN weights given the additional generations of random shift diffusion from their initial [-0.5,0.5) range. However, testing of standard evolution with wider initial ranges demonstrated poor performance. If the highly fit regions of the solution space for larger values of θ_G are simply distinct from those of lower ones, then it remains unclear why the ramped evolution would provide superior results.

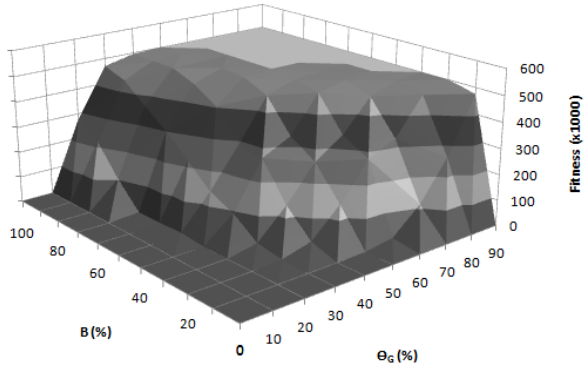

Fig. 6. Average testing fitness over the (θ_G,B)-parameter space using the EANN control system.

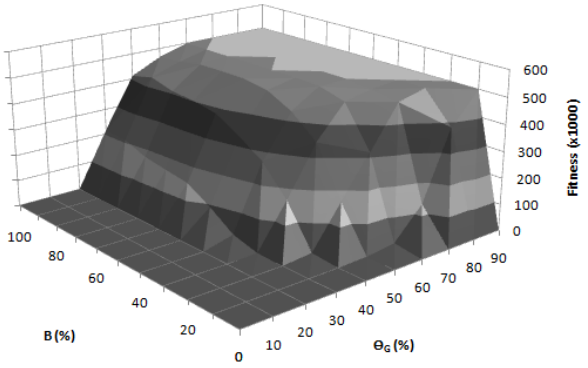

Fig. 7. Average testing fitness over the (θ_G,B)-parameter space using the ECAM control system.

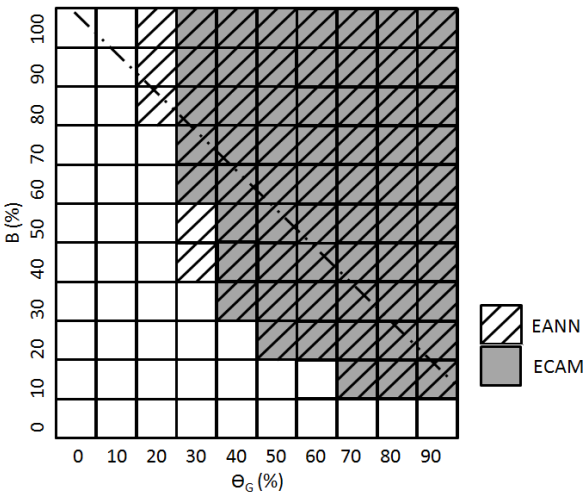

Fig. 8. Surface coverage of average testing fitnesses exceeding 10^5 with either algorithm. The dashed line represents the approximate threshold between diffusionless and diffusive instances.

B. Evolutionary Neural Network Comparison

1) Does the performance of the EANN and ECAM algorithms differ by problem instance?: To consider the quantitative differences in solution quality between the EANN and ECAM control systems, first observe the results of using either

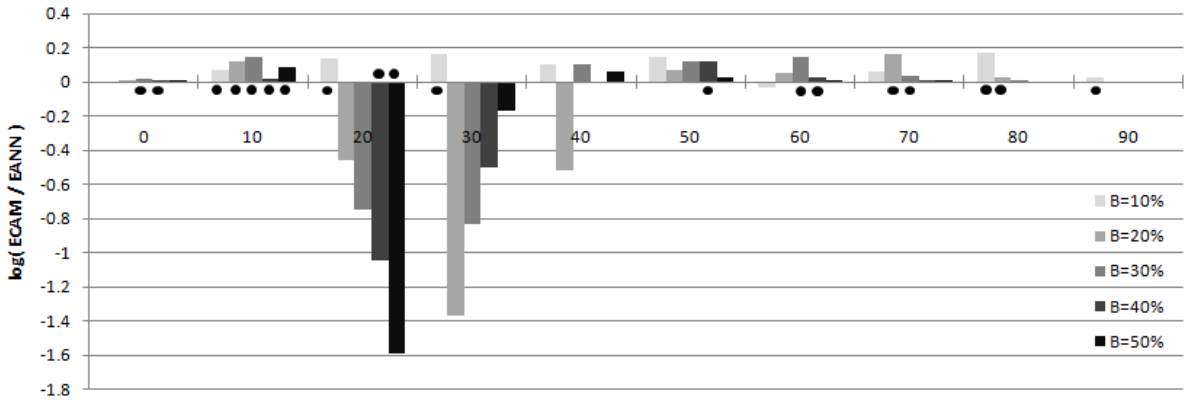

Fig. 9. Comparison of fitness of averaged best results of ECAM vs. EANN control systems over the (B,θ_G)-parameter space. Values presented are logarithmically scaled differences grouped by θ_G.

algorithm over the (B, θ_G)-parameter space. A visual representation of the surfaces of average test fitness are presented in Figures 6 and 7. Each displayed value is the average result of 30 runs of the control system in question with different random seeds (this implies 30 different random control system configurations). The result of a single run is determined as the average number of simulation steps prior to exceeding the θ-consensus criterion (see Eq. (2)) over 100 test simulations of the final best individual discovered in a given EA execution.

The surface plots illustrate the performance of either algorithm over the parameter space for a given network structure. The ridge of sharp increase in the order of magnitude of the average fitness shows the exponential growth of the ease and thus duration of control of the network as the constraints loosen. This ridge corresponds approximately to the line indicating diffusionless instances as defined in [30]. The algorithms here are capable of extending the high quality results ($f > 10^5$) to instances beyond that line, indicating their learned ability to cope with limited amounts of uncontrolled influence diffusion. For example, in the presented network, if $\theta_G = 70\%$ then the diffusionless threshold budget is $B^* = 186$ ($\approxeq 30\%$). However, the observed results indicate average control fitnesses in the 10^5 order of magnitude with a $B = 60$ (10%), and the ECAM algorithm in particular reaches the simulation run limit in approximately half of the considered configurations with $B = 120$ (20%).

Figure 8 illustrates the ridge of high quality results projected and overlaid for both algorithms. There it can be readily observed that both algorithms achieve high quality results over a majority of the parameter space. The location of this ridge is comparable between both algorithms, with EANN spreading slightly wider. From Figures 6-8, the EANN algorithm appears to begin the lower end of the ridge with lower θ_G values, while the ECAM algorithm appears to rise sharper bringing both algorithms to the simulation cap at approximately the same range of instances. By means of example, the average rate of increase over θ_G values with B fixed at 40% is 7100 ($\frac{\Delta f}{\Delta \theta_G}$) for ECAM and 6300 for EANN. All of this suggests that the ECAM algorithm is more consistent once the problem instance parameters become favourable, while the EANN algorithm may be more robust over greater ranges of parameters. Finally, it is observed that the ECAM algorithm does not suffer the inability to control large budget instances that its predecessor, the AM control system, suffered in [30].

2) Is the ECAM algorithm variant a statistically significant improvement over the standard EANN?: A more direct comparison of these two neurocontrollers is performed using the difference of their average solution quality. The results are presented as log-ratios to prevent the significantly differing magnitudes from diminishing the scale of more difficult problem instances. As above, a Wilcoxon Signed-Rank test with a 99% confidence interval is used to determine if the observed results differ significantly by algorithm alone. Indications of such are presented as dots along the horizontal axis in Figure 9.

The most glaring result from Fig. 9 is the performance of the EANN algorithm on the $\theta_G = 20, 30$ instances, in which its average results outperform the ECAM algorithm by up to nearly two orders of magnitude. These results, while interesting, are somewhat misleading. The high quality is contributed by a few very good results averaged over 30 random configurations. This causes many of the results for which EANN appears superior to not measure as statistically significant. Instances for which the ECAM algorithm appears superior are typically significant. Of the 95 considered θ-CAP instances with $B \leq 50\%$ over 4 network structures, the ECAM algorithm outperforms the EANN algorithm with significance at a ratio of 39:4:52 (ECAM:EANN:neither).

The increasing trend of the EANN's results for $\theta_G = 20\%$ and decreasing trend for $\theta_G = 30\%$ implies that parameters become more favourable for the ECAM algorithm as the budget increases, but still insufficient for the narrow balance required for the smallest values of θ_G. The ECAM algorithm, however, is particularly consistent in its superiority for instances near the ridge of high quality solutions. This sharper increase in solution quality relative to θ_G suggests that the ECAM algorithm is more sensitive to changes in this parameter than the EANN algorithm.

Figure 10 presents a comparison between these algorithms over the complete parameter space. The algorithms are com-

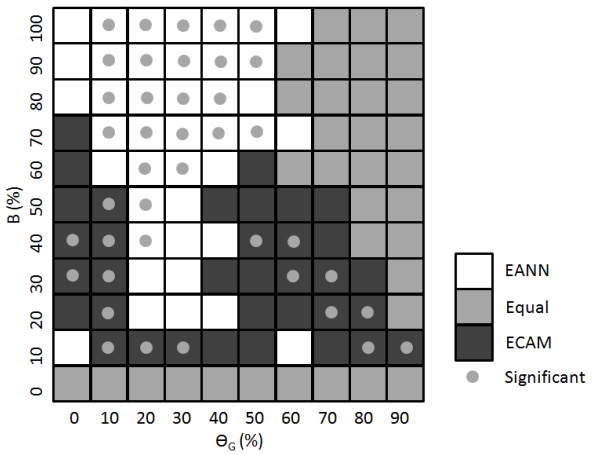

Fig. 10. Comparison of superior average testing fitnesses over the (B, θ_G)-parameter space.

pared in terms of which has a higher average testing fitness over the 2D range of instances in the (θ_G,B)-parameter space for a given network. The significance ratio when considering the full range of budget values is 17:23:60 and is nearly divided by the $B = 50\%$ line of instances. The ECAM algorithm consistently shows statistically significant improvements versus the EANN algorithm at both the low and high ends of θ_G values for instances with $B < 50\%$. However, the AM biased initialization of the ECAM algorithm clearly loses its advantage for instances with $B > 50\%$. Although the ECAM is capable of learning high quality results over large budget instances (see Figure 8), it does so from a disadvantaged initial condition, causing the standard EANN to show significantly better results for the same duration of training.

To state outright whether or not the ECAM algorithm is strictly an improvement over the standard EANN would overlook the consequence of the No Free Lunch theorem [4]. It is more accurate to discuss under which conditions the use of either algorithm is appropriate. From the above results there is a clear distinction among θ-CAP instances for which either algorithm outperforms the other. Given interest in the complete parameter space one could construct a hyper-heuristic for algorithm selection based primarily on the budget parameter. However, as discussed in Section III, the primary interest in this work is towards the more difficult instances (low budget, low consensus threshold). Over the restricted space of these instances the ECAM algorithm appears to be an improvement.

3) What qualitative differences exist between the EANN and ECAM algorithms that could account for the observed quantitative results?: The qualitative differences between the EANN and ECAM algorithms derive from the differences of their initialization and structures. The initial state of the ECAM algorithm is similar by definition to the CAM algorithm, while the initial state of the EANN algorithm can only be described as random. It is observed accordingly, that the population of the standard evolution ECAM algorithm consistently begins with a higher average fitness than that of the EANN for low budget instances ($B \leq 50\%$), and a lower average fitness for the high budget instances. The speed of convergence and relative final fitness, however, do not appear to be dependent on the initial conditions. These are instead determined to be a factor of the properties of the instance under consideration, as is explained below.

One advantage to the design of the ECAM algorithm is that the hidden layer neurons therein are given prescribed meanings prior to training. Specifically, the output from each hidden neuron corresponds to the ratio of states at its inputs. It follows that what the ECAM algorithm then learns by adjusting the hidden-to-output layer axon weights is the strength of connections in a fully connected network of reinforcing and oppositional relations between controlled nodes. In this way, the decision making is done on the level of setting signals the same or different from other controlled nodes (and their corresponding neighbourhoods). The same cannot be said of the EANN algorithm in which hidden nodes align with controlled nodes in quantity alone.

A visual assessment of the neurocontrollers in operation revealed the two algorithms evolved different forms of control. It was observed that when possible, the EANN algorithm evolved to maintain fixed output signals at different regions of the network, which creates converged regions of the network that were in opposition to each other, thereby creating a balanced network overall. On the other hand, the Anti-Majority foundation of the ECAM algorithm leads it to a rapid flickering behaviour that, when successful, maintains balance in the social network by injecting noise into its otherwise converging diffusion.

These behavioural differences between the EANN and ECAM algorithms can be measured in terms of their *intra-* and *inter*-neuron output standard deviations as presented in Figure 11. The intra-neuron output standard deviation is the observed standard deviation of all of the outputs of a single neuron in the output layer of the elitism ANN over the course of a simulation. This value is averaged over all output nodes, and then over 30 separate executions to produce a point in Fig. 11. The inter-neuron output standard deviation is calculated as the standard deviation over all neurons of the average output of each neuron in the same elitism ANN. As such, a control system which outputs a cleverly constructed fixed signal across the network will have a very low intra-neuron standard deviation (because each given node will have very little change), and a relatively high inter-neuron standard deviation (owing to the fact that different states will be fixed for different neurons). Conversely, an ANN control system which exhibits a flickering behaviour will have a large variance within each node, but all nodes will behave approximately the same (from this measure). This is precisely the observed behaviour in Fig. 11.

The foregoing results indicate a clear distinction in evolved strategy brought about by the bias (and lack thereof) in the structure and initial conditions of the two considered EANN algorithms. The fixed output strategy of the standard EANN is considerably more effective than the ECAM strategy when it is successful. However, discovering a successful combination of control system and ANN weight configurations appears to have a low likelihood. Even a single solution that suc-

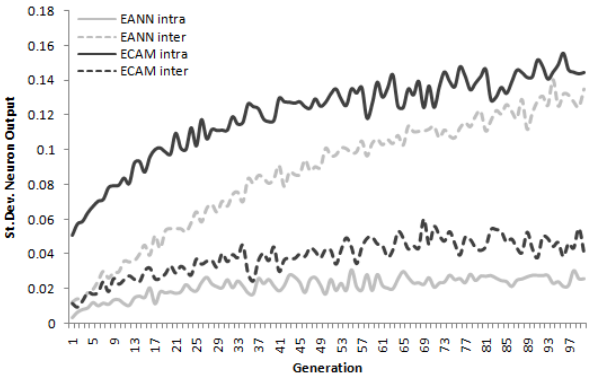

Fig. 11. Intra- and inter-neuron output standard deviations averaged over 30 executions of EANN and ECAM algorithms.

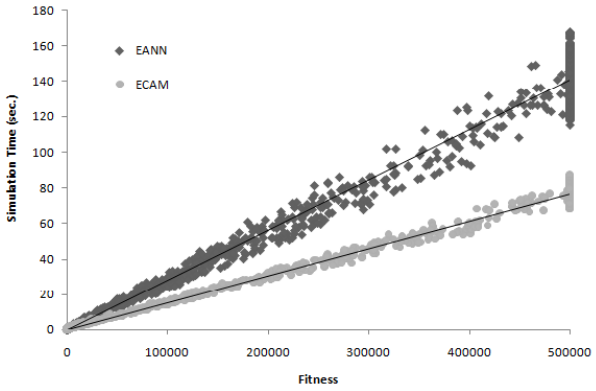

Fig. 12. Comparison of simulation execution time versus run length for a range of θ_G values using $B = 30\%$.

cessfully manages to stabilize the social network within the given threshold is inconsistent in its ability to do so. This inconsistency is likely a result of the reliance of the control system on the stochastic process of convergence among the uncontrolled local subsections in the network prior to reaching a steady state. Selection of appropriate choke-point nodes is expected to play a significant factor in the outcome of this strategy, however, the study of configuration strategies is the subject of ongoing investigation to be reported in future work.

4) What is the relative efficiency between the EANN and ECAM algorithms?: In addition to its superior results, the biased structure of the ECAM algorithm requires significantly less memory. Whereas the EANN algorithm uses two fully connected layers of axons requiring storage for O($V \cdot V_C + V_C^2$), the EANN must train only a single layer requiring only O(V_C^2). This smaller network structure consequently requires fewer calculations to determine a control signal from a set of inputs. This means that an ECAM control system requires less execution time to balance a network to the same simulation fitness as the standard EANN control system. A comparison of the execution times of the EANN and ECAM algorithms, over the range of $\theta_G = \{0, 10, ..., 90\}$, $B = 30\%$ problem instances, is presented in Figure 12. Simulation times were measured in nanoseconds from the beginning of simulation until termination, and do not include the set-up and configuration steps of the network and control systems. Simulations were carried out in serial on an Intel Core i7-3632QM 2.20GHz CPU with 8GB RAM. Implementation is available online[1]. From the figure there is a clear distinction in the linear trend lines supporting the above claim, with results for the simulation step limit requiring up to twice as much execution time for the EANN control system as for the ECAM for the selected problem size.

C. Landscape Analysis

The remaining experiments aim to explain the results observed in the previous experiments using information theoretic measurements of the fitness landscape. First, the measures are considered according to their variance over the (B,θ_G)-parameter surface to validate their use on θ-CAP instances

[1]http://sourceforge.net/p/influence/

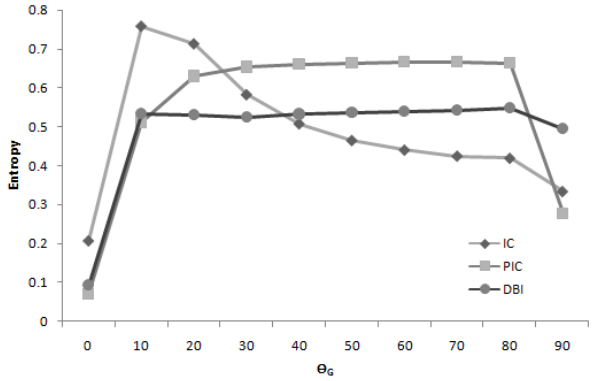

Fig. 13. Characteristic curves of the IC, PIC, and DBI measurements over the range of θ_G values for a fixed budget.

and observe the characteristic differences that are observable with known changes in difficulty. Next, the results of the landscape measures using EANN and ECAM representations for the random walk are compared with interest toward finding correlations with the previous algorithm fitness comparison.

The experiments performed consisted of 30 random walks, each of 1000 steps per random configuration. Thirty random configurations were used for each problem instance for a total of 900000 simulations per result. The landscape measures are calculated independently for each walk then averaged over all walks to present the results for each instance.

1) Do the information theoretic landscape measures correlate with changes in θ-CAP instance difficulty?: The landscape measurements taken over the (B,θ_G)-parameter space show consistent trends corroborated over all observed networks. The characteristic curves of the three landscape measures over a range of θ_G values for a fixed budget is shown in Figure 13. The specific curve shown is for $B = 10\%$, however the same curve is observed for each budget. As the budgets increase the transition phase of the curve adjusts, inline with the cliff observed in the surface plots (Figs. 6,7).

Random walks for instances with $\theta_G = 0$, as with evolutionary searches of these instances, typically have simulation run lengths of only 1. The average fitness for random walks with this value (for all considered networks/budgets) is $\approxeq 1.05$

with a standard deviation of $\approxeq 0.2$. This implies that the typical surface of such a walk consists primarily of long plateaus. As such the initial points of all three measures describe very little variety, and stand out as distinct from the rest of the curves.

The *information content* (IC) characterizes the frequency and variety of transitions in the random walk (where larger numbers refer to all transitions being more likely and smaller numbers refer to one or few transitions being more likely). The high initial IC value for $\theta_G = 10$ indicates a reasonable balance between all 6 transition types on average over the random walks of this instance. The IC is observed to decrease in line with the instance difficulty over both threshold and budget values. The curve downwards approaches a limit of 0.4 which corresponds to the equal likelihood of two (out of six) possible transitions with probability $\approxeq 0.3$. That is, the decrease in IC likely corresponds to the disappearance of plateaus from the random walks as the fitness space becomes less predictable. Note, that due to the exponential distribution of testing results for any given solution to a single instance of the θ-CAP, a greater variety of fitnesses with a random solution (corresponding to a more rugged landscape) counter-intuitively implies a more fit region and/or an easier to solve instance.

The *partial information content* (PIC) measures ratio of transitions in the random walk to the total length. Here, the PIC increases as the threshold and budget parameters increase (corresponding to decreased difficulty). This supports the findings of the IC measure as it indicates increased modality of the random walks. Interestingly, the PIC for all considered instances curves towards a value of $0.\dot{6}$, indicating that at most two thirds of any walk are spent transitioning between increasing and decreasing fitness, while the remaining third are so-called 'smooth' edges. There is no clear evidence to suggest why this limit exists.

The *density basin information* (DBI) measures the flat (non-transitioning) regions of the random walks. The DBI values remain relatively consistent throughout all considered instances between 0.52 and 0.54, accounting for the remaining one third as an approximately equal number of flat-increasing or flat-decreasing steps.

For all instances there is a sharp decrease in all three measurement values when the simulations of the random walks plateau at the simulation limit. However, certain budget levels show a slight increase in IC and DBI values prior to this owing to the increased variety from the presence occasional simulation limit plateaus in the walk.

2) Do the landscape measures differ between the EANN and ECAM algorithms?: A comparison of the landscape results from the EANN and ECAM algorithms offers very little distinguishing characteristics between the fitness space explored by the two algorithms. Although the algorithms have distinct structures including different dimensionality of solutions and initial search space biases, landscapes using either algorithm reveal the same trends relative to changes in the θ_G and B parameters as described above. This similarity implies that the differences between the two algorithms cannot immediately be attributed to an easier search space for one over the other. This is surprising given the evidence of significant differences in the fitness of the evolved solutions. Similar difficulty in landscape measurements of EANN fitness spaces was noted in [36], where any single solution had a non-uniform distribution of potential fitnesses.

EANN B=20%					
θ_G	Avg. Fit.	St.Dev.Fit.	PIC	IC	DBI
10	2.2	1.95	0.767	0.528	0.537
20	6.2	6.78	0.690	0.633	0.530
30	16.7	18.61	0.559	0.654	0.528
40	43.1	49.95	0.488	0.662	0.534
50	125.4	162.38	0.449	0.666	0.539
60	568.3	970.85	0.427	0.666	0.542
ECAM B=20%					
θ_G	Avg. Fit.	St.Dev.Fit.	PIC	IC	DBI
10	2.2	1.95	0.774	0.537	0.538
20	6.3	6.52	0.685	0.636	0.525
30	17.0	17.38	0.550	0.656	0.527
40	46.2	49.04	0.478	0.663	0.535
50	177.2	257.15	0.439	0.667	0.539
60	2487.9	7070.30	0.420	0.666	0.543

TABLE I
SAMPLE RESULTS COMPARING THE LANDSCAPE MEASURES OF EANN AND ECAM STRUCTURED RANDOM WALKS.

Table I presents sample landscape data for a cross-section of the parameter space fixed at $B = 20\%$ for the network instance discussed in the previous section. The results are similar in nature to all other diffusive instances considered. A slight decrease in IC measure for the majority of ECAM results suggests the EANN algorithm may be less consistent in its results. The IC measure of the ECAM algorithm is less than that for EANN in 30 of 40 considered instances by an average amount of 0.0068. More low-valued plateaus would contribute to a higher IC versus the rapid oscillating fitness of an IC with a greater success rate. The random walks using the ECAM algorithm tend to have higher average fitness (in 34 of 40 instances) and a lower average standard deviation (in 22 of 40 instances), supporting the above suggestion. The improved quality of the ECAM random walks may also be attributable to the fact that they traverse a smaller biased fitness space, as compared to the EANN which is double in dimensionality and uses strictly random initial locations. The very high quality results of the EANN algorithm depicted in Fig. 9 are not observed within any of the 900 random walks of the given problem instance, which is further indication of their outlier status in the solution space.

V. CONCLUSION

This paper demonstrates the ability of an evolutionary neurocontroller to successfully act as the behavioural component for the θ-CAP. The novel, problem specific, EANN variant, ECAM, consistently outperformed the standard EANN variant in a majority of considered problem instances. ECAM, which was designed as a middle ground between the EANN and CAM algorithms, provides benefits from both. It achieves better quality results in less time on the more difficult small budget instances. While the standard EANN attains consistently better results for instances with large budgets, the ECAM overcomes the weakness of the Anti-Majority heuristic to

demonstrate extended durations of control for these instances as well.

Two distinct strategies for balancing the network were observed in the best solutions from the two neurocontroller variants considered. The standard EANN evolved a fixed-output convergence to two or more opposed subsection of the network of size equal enough to prevent exceeding the θ_G threshold. The ECAM, on the other hand, evolved to inject noise into the network, sewing enough chaos to prevent convergence among the highly communicative agents of the social network. It is important to note that this was accomplished via limiting and biasing the search space of all possible EANN weights to only those available in the ECAM definition.

The landscape analysis comparison of the two algorithms revealed unexpected similarity in the nature of the fitness landscapes traversed. Slight changes in the observed values, and increased average fitness over the majority of random walks, suggests that both networks search on the same fitness space, with the ECAM simply biased to a promising region thereof.

A ramped evolution scheme was proposed and analysed with the intent of gradually training the neurocontrollers over increasing levels of difficulty. The ramped scheme was observed to improve the quality of the evolved solutions, however some uncertainty remains regarding the exact cause of the improvement.

Future Work

There are many potential directions for future research in the θ-CAP and more broadly the NCP areas of study. Most pressing is the need to optimize the control system configuration component in place of the random configuration used here. Results using random configurations show a distinct inequality in the ability of a given control system between various random seeds. Optimization of this component will likely improve overall testing quality at the cost of increased dimensionality of search.

It is likely that the inconsistency arising from the fixed-output control strategy of the standard EANN variant could be alleviated be more intentional selection of choke-point nodes and/or broader diffusion of the controlled set throughout the social network. Alternatively, it is well known that EAs are fast global optimizers, but poor local optimizers when compared to gradient-based techniques [44]. Thus, the addition of gradient descent or some other local search could aid in improving the consistency of the EANN neurocontroller.

Further study of the applicability of landscape analysis to the θ-CAP, as well as for EANN search in general is needed. Correlation between landscape values and instance difficulty suggests that some information could be gained by using the landscape measures prior to execution; however, the exponential distribution of testing results makes their use difficult. Future work should consider varying the sensitivity parameter according to the observed standard deviation of some sample testing results to potentially alleviate these negative effects and provide more accurate information. Alternatively, multiple evaluations of each step in the walk would give an averaged fitness per location, with the effect of smoothing the observations for the non-homogeneous fitness spaces observed here.

REFERENCES

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

[2] R. S. Burt. The network structure of social capital. *Research in organizational behavior*, 22:345–423, Jan. 2000.

[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] J. Culberson. On the futility of blind search: An algorithmic view of no free lunch. *Evolutionary Computation*, 6:109—-127, 1998.

[5] 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.

[6] 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.

[7] F. Gomez and R. Miikkulainen. Solving non-Markovian control tasks with neuroevolution. *IJCAI*, 99:1356–1361, 1999.

[8] 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.

[9] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[10] I. Jeong, S. Kim, and J. Loe. A New Hybrid Genetic Algorithm and Its Application to Control. In *Proceedings of International Conference on Neural Information Processing*, pages 1669–1674, 1994.

[11] I.-K. Jeong, C. Choi, J.-H. Shin, and J.-J. Lee. A modified genetic algorithm for neurocontrollers. *Proceedings of 1995 IEEE International Conference on Evolutionary Computation*, 1:306–311, 1995.

[12] T. Jones. *Evolutionary algorithms, fitness landscapes and search*. Phd thesis, University of New Mexico, 1995.

[13] N. Kaise and Y. Fujimoto. Applying the Evolutionary Neural Networks with Genetic Algorithms to Control. *Simulated Evolution and Learning*, pages 223–230, 1999.

[14] 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.

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

[16] P. Kim and P. Vadakkepat. Evolution of control systems for mobile robots. *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02*, 1:617–622, 2002.

[17] 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.

[18] N. Kohl and R. Miikkulainen. Evolving neural networks for strategic decision-making problems. *Neural networks : the official journal of the International Neural Network Society*, 22(3):326–37, Apr. 2009.

[19] 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.

[20] 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.

[21] S. Liu, Y. Wang, and Q. Zhu. An evolutionary neural network based tracking control of a human arm in the sagittal plane. *Advances in Computation and Intelligence*, pages 316–325, 2007.

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

[23] S. Luke. *Essentials of metaheuristics*. Lulu, second edition, 2013.

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

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

[26] R. Neruda, S. Slušný, and P. Vidnerová. Behavior Emergence in Autonomous Robot Control by Means of Evolutionary Neural Networks. *Advances in Computational Algorithms and Data Analysis*, pages 235–247, 2009.

[27] E. Pitzer and M. Affenzeller. A comprehensive survey on fitness landscape analysis. In *Recent Advances in Intelligent Engineering Systems*, pages 161–191. 2012.

[28] 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.

[29] M. Riedmiller. Learning to control dynamic systems. *Cybernetics and Systems Research*, pages 1055–1060, 1996.

[30] A. Runka and T. White. Towards Intelligent Control of Influence Diffusion in Social Networks. Technical Report TR-14-06, School of Computer Science, Carleton University, 2014.

[31] 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.

[32] N. Saravanan and D. Fogel. Evolving neural control systems. *IEEE Expert: Intelligent Systems and Their Applications*, 10(3):23–27, 1995.

[33] R. S. Sexton, R. E. Dorsey, and J. D. Johnson. Toward global optimization of neural networks: A comparison of the genetic algorithm and backpropagation. *Decision Support Systems*, 22(2):171–185, Feb. 1998.

[34] 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.

[35] T. Smith, P. Husbands, P. Layzell, and M. O'Shea. Fitness landscapes and evolvability. *Evolutionary computation*, 10(1):1–34, Jan. 2002.

[36] T. Smith, P. Husbands, and M. O'Shea. Not measuring evolvability: initial investigation of an evolutionary robotics search space. *Proceedings of the 2001 Congress on Evolutionary Computation*, 1:9–16, 2001.

[37] J. Teo and H. a. Abbass. An information-theoretic landscape analysis of neuro-controlled embodied organisms. *Neural Computing & Applications*, 13(1):80–89, Apr. 2004.

[38] V. K. Vassilev, T. C. Fogarty, and J. F. Miller. Information characteristics and the structure of landscapes. *Evolutionary computation*, 8(1):31–60, Jan. 2000.

[39] M. Ventresca and B. Ombuki. Search space analysis of recurrent spiking and continuous-time neural networks. *Neural Networks, 2006. IJCNN'06.*, 2006.

[40] M. Ventresca, B. Ombuki-Berman, and A. Runka. Predicting genetic algorithm performance on the vehicle routing problem using information theoretic landscape measures. In *Evolutionary Computation in Combinatorial Optimization*, pages 214–225, 2013.

[41] E. Weinberger. Correlated and uncorrelated fitness landscapes and how to tell the difference. *Biological cybernetics*, 336:325–336, 1990.

[42] P. J. Werbos. Beyond regression: new tools for prediction and analysis in the behavioral sciences. *Harvard University*, 1974.

[43] A. Wieland. Evolving neural network controllers for unstable systems. *Proceedings of International Joint Conference on Neural Networks, IJCNN'91*, 2:667–673, 1991.

[44] X. Yao. A review of evolutionary artificial neural networks. *International journal of intelligent systems*, 8(4):539–567, 1993.

[45] X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999.