Learning-based control of the consensus value in unknown graphs

Florin Gogianu[1], Lucian Buşoniu[1], Irinel-Constantin Morărescu[1,2]

***Abstract*— We consider the problem of optimal budget allocation for consensus reaching in unknown networks. The network is represented by a directed graph whose vertices corresponds to agents influencing each other. At discrete instants, agents are influenced by an external entity that intends to sway the consensus to a target value. Between two external influence instants, the states of the agents evolve continuously due to the network dynamics. Prior results establish that, in known networks, a water-filling strategy that targets the most influential agents first is optimal. In our approach to the unknown-network setup, the marketer uses the evolution between two influence instants to update a learned model of the graph.Then, the control allocation at the next marketing instant is done according to the water-filling strategy applied to the current model of the graph. Our main analytical contribution states that the sub-optimality of the budget allocation induced by the approximation of the graph is related to the error of the learning algorithm. Extensive numerical analysis illustrates the performance of our method and suggests a regularization term that improves it further.**

***Index Terms*— Control of networks, machine learning, opinion dynamics**

I. Introduction

MULTI-AGENT systems are used in many applications such as robotic teams, energy and telecommunication networks, opinion dynamics in social networks, analysis of biological networks, etc. The desired behaviour of the agents is often described in terms of consensus or agreement. The analysis and design of control strategies to reach consensus in multi-agent systems received significant attention in the literature [1]–[3]. Many works focus on finding sufficient conditions ensuring the consensus in various types of networks: directed or undirected, fixed or time-varying.

An interesting application of consensus algorithms is opinion dynamics over social networks [4]–[8]. The contribution of each agent to the final consensus value of a directed network (or the agent influence power) has been investigated in both fixed [9] and time-varying graphs [10]. Finding the influence power so as to prioritize major influencers in the network is a key tool in the design of targeted marketing strategies [11]–[13]. For opinion dynamics driven by consensus algorithms, detecting the influencers corresponds to measuring the centrality [9] of the individuals in the social network.

This work was supported by project DECIDE, no. 57/14.11.2022 funded under the PNRR I8 scheme by the Romanian Ministry of Research, Innovation, and Digitisation, and by project "Romanian Hub for Artificial Intelligence–HRIA", Smart Growth, Digitization and Financial Instruments Program, MySMIS no. 334906. (1)Automation Department, Technical University of Cluj-Napoca, 400114 Cluj-Napoca, Romania. (2)Université de Lorraine, CNRS, CRAN, 54000 Nancy, France. (emails: florin.gogianu@gmail.com, lucian@busoniu.net, constantin.morarescu@univ-lorraine.fr)

In most existing works, analysis and the control design are based on precise knowledge of the interaction graph between the agents. Nevertheless, this information is unknown in many practical applications, such as viral marketing over social networks. In [14], the authors approximate from data the influence power in a fully connected social network. In contrast, our goal in this paper is to use the known form of the agents' dynamics in order to learn the interaction graph (that may be sparse), before applying a budget allocation strategy inspired from [12] to control the consensus value.

An important line of work is concerned with system identification of network topologies [15], [16] and their identifiability [17], [18]. Our graph learning component shares with the Wiener filters used in [15] the same mean squared error objective, but is not limited to recovering graph topology (whether two nodes are connected) and allows for additional constraints on the learned coefficients. Furthermore, our method does not assume independent white noise sources on each node that drive identification. Other lines of work, such as [16], [17], assume stationary policies, which we do not. The methods above resort to orthogonalization procedures that can be numerically unstable in the low-data regime that we require, which we avoid with our gradient based learning.

Several works formulate the control of consensus as a reinforcement learning (RL) problem, e.g. [19]–[21], usually also in the unknown-graph setting. To avoid the large sample complexity of RL methods, resulting from the need to directly learn control policies, we focus instead on estimating the graph from data and using it for control.

The works of [22], [23] solve supervised learning problems on graph-structured data using neural networks optimized by gradient descent. We concern ourselves with the generative problem [24]: given observations from a process that has an underlying graph structure, we aim to recover the underlying graph. Similarly to [25], we parametrize the weights between graph nodes. However, instead of using an attention mechanism for inferring the relations between nodes, we simply learn the weights from data, taking advantage of the linear interactions between agents and the sparsity of the graph.

Our main contributions are the following. 1) We develop an algorithm to learn the interaction graph from state transition observations, and combine it with a technique to allocate marketing budget to the agents across multiple campaigns so as to sway the opinion in a desired direction. 2) We analyze the optimality loss induced in the opinion by the learned graph approximation, showing how this loss decreases with the errors made in the centrality of the nodes. 3) We provide an extensive numerical analysis on randomly generated Barabási-Albert graphs that: confirms the relationship from our analysis; shows that for most graphs, performance with the learned

graph is close to the known-graph optimum; and for the edge-case graphs that are difficult to learn, proposes an extra regularization term to improve performance.

II. Problem formulation

Consider a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with the vertex set $\mathcal{V} := \{1, \dots, N\}$ and edge set $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$. The state of each vertex $i \in \mathcal{V}$ at time t is denoted $x_i(t)$. In this work we only consider scalar states, however our method generalizes to higher dimensions as well. In the sequel we normalize the states of the agents such that $x_i(0) \in [0, 1], \forall i \in \mathcal{V}$. Since we are using standard consensus dynamics, it follows that $x_i(t) \in [0, 1], \forall i \in \mathcal{V}, t \geq 0$. Specifically, the states of the agents evolve according to the following linear consensus dynamics:

$$\dot{x}_i(t) = \sum_{j \in \mathcal{N}_i} a_{ij} \big(x_j(t) - x_i(t)\big), \tag{1}$$

where $\mathcal{N}_i$ is the set of neighbors influencing i and $a_{ij} > 0$ is an entry in a weighted adjacency matrix $A \in \mathbb{R}^{N \times N}$. Note that $a_{ij} = 0$ iff node i is not influenced by the node j. A is row-stochastic, i.e. $\sum_j a_{ij} = 1, \forall i$ and it's diagonal is zero.

Assumption 1: The graph $\mathcal{G} = (\mathcal{V}, L)$ is weakly connected (sometimes referred to as quasi-strongly connected) i.e., it contains at least one directed spanning tree.

The linear dynamics in (1) lead to a state trajectory $x_i(t)$ for each node in the graph. We observe, with some sampling period s, samples $x_l^i = x_i(t^l)$ at discrete time indices $l \geq 0$, such that $t^l = sl$.[1] For convenience, we stack all the node states at time index l, resulting in a sequence of vectors $x_0, ..., x_L$, of size N. We further shorthand sequence $[x_0, ..., x_L]$ to $X \in \mathbb{R}^{L \times N}$. "Slicing" this object is possible, so we refer to observations $[x_l, ..., x_{l+p}]$ as $X_{l:l+p}$.

Given observations generated by the unknown graph $\mathcal{G}$, we aim to find an optimal allocation of interventions on the individual states $x_i(t)$ such that the distance to some desired target value ω is minimized. We consider a plausible scenario where an agent external to the graph computes and applies a (sparse) action on nodes in the graph, at certain time intervals referred to as campaigns. The number of campaigns $M + 1$ is finite and generally small, with $k \in \{0, ..., M\}$, and campaigns are distributed in time by $h = t_{k+1} - t_k$. At time t_k, the agent computes sparse actions $u_i(t_k)$, with $u^i \in \{0, \bar{u}\}$ and $\bar{u} \in [0, 1]$, such that the cumulative actions over all campaigns and agents stays within the total budget B: $\sum_{k=0}^{M} \sum_{i=0}^{N} u_i(t_k) \leq B$. The action at campaign time k is applied as follows:

$$x_i(t_k) = u_i(t_k)\omega + [1 - u_i(t_k)]\, x_i(t_k^-), \tag{2}$$

where $x_i(t_k^-)$ is the state of the network just before the campaign. In contrast to previous work [12], we do not have access to the adjacency matrix of $\mathcal{G}$, which increases the difficulty of the problem. Consequently, in our setup the time t_0 of the first campaign is non-zero and usually chosen to be h, so that we give time for the graph-learning component of the method to find an initial approximation of the graph.

[1]Note that for x we move the agent index i to the superscript so we can conveniently refer to sample index l in the subscript (but t^l has l in superscript to avoid confusion with the campaign times t_k).

III. Methods

When $\mathcal{G}$ is known, [12] provides algorithms for optimal discrete intervention on the belief state of the network, under a given budget. Usually however, one does not have access to the structure of the network but may only observe the evolution of the states of each node in time.

In what follows, we propose a method that exploits the knowledge about the structure of the transition dynamics in (1), our light assumptions regarding the adjacency matrix, and observations X, in order to approximate the graph, and design budget allocations based on this approximation.

A. Graph learning from observations

The state of the graph at the $l+1$ observation instant can be written in terms of the state at the previous time, x_l and the adjacency matrix using (1). We approximate these dynamics with Euler integration between observation instants:

$$x_{l+1}^i = f\left(x_l^i, A\right) = x_l^i + s \sum_{j \in \mathcal{N}_i} a_{ij}(x_l^j - x_l^i) \tag{3}$$

Next, we propose treating the adjacency matrix as trainable weights and learn them from data. Therefore, we formulate the following learning objective:

$$\mathcal{L}_{\text{dyn}}(\theta) = \sum_{i}^{N} \left(x_{l+1}^i - f\left(x_l^i, A_\theta\right)\right)^2, \tag{4}$$

where A_θ is an approximation of the real adjacency matrix A that generated the observations, parameterized by $A_\theta^{ij} = \exp(\Theta_{ij}) / \sum_j \exp(\Theta_{ij})$, with $\Theta \in \mathcal{R}^{N \times N}$. This softmax parameterization encodes the row-stochasticity assumption we made about A. Objective (4) encodes the goal of learning weights Θ such that the dynamics of the data-generating process are closely matched.

To encode the lack of self-connections in the graphs we consider, we add a regularization term to the objective driving the diagonal of A_θ to be zero, ending up with:

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{dyn}}(\theta) + \sum_{i}^{N} \Theta_{ii} \tag{5}$$

We optimize this objective using stochastic gradient descent (SGD). At each iteration of the algorithm, we sample a small batch of pairs of observations at consecutive time-steps and use them to compute the gradient of the objective with respect to the parameters Θ. To update parameters we use Adam [26].

Algorithm 1 Graph Learning from Observations (GLO)

procedure GLO(X)
  $A_\theta \in \mathcal{R}^{N \times N} \backsim \mathcal{U}$ ▷ Kaiming uniform init. [27]
  while $\mathcal{L}(\theta) < \varepsilon$ **do**
    $X_l, X_{l+1} \backsim X$ ▷ sample consequtive states
    $\widehat{X_{l+1}} \leftarrow f(X_l, A_\theta^i)$ ▷ run dyn. with A_θ^i
    $\mathcal{L}(\theta) = \left[\widehat{X_{l+1}} - X_{l+1}\right]^2$
    $A_\theta^{i+1} \leftarrow \text{SGD}(\mathcal{L}(\theta_i))$ ▷ update the weights
  end while
  return A_θ
end procedure

The full procedure is described in Alg. 1. To avoid a heavier notation, in this subsection we write the approximate dynamics (3), loss functions (4)–(5), as well as pseudocode instructions (e.g., sampling state pairs in line 4) for individual data samples, leaving implicit the fact that in reality we apply them to mini-batches of data. Our method acquires only a few new observations (4 in the experiments) between consecutive campaigns. This data scarcity, relative to the large number of estimated coefficients, would be problematic to most constrained optimization solutions due to numerical instabilities.

B. Optimal control in known graphs

We adapt prior work for finding the optimal allocation under a budget constraint for *known* graphs. A complete description and analysis of the algorithm can be found in [12]. We provide a brief description here, after which we extend the method to the case of *learned* graphs in Sec. III-C, and analyze the extension in Sec. IV.

For the sake of simplicity and without loss of generality we consider $\omega = 0$. The dynamics of the network becomes:

$$\begin{cases} \dot{x}(t) = -Lx(t), \quad t \in [t_k, t_{k+1}) \\ x(t_k) = \big(I - U(t_k)\big)x(t_k^-), \quad \forall k \in \{0, \dots, M\} \end{cases} \tag{6}$$

where $U(t_k) = \mathrm{diag}(u(t_k))$. We denote the normalized left eigenvector of L associated with the 0 eigenvalue by v, meaning that $v^\top L = v^\top$ and $v^\top \mathbf{1} = 1$. Further denote the budget at each campaign k with β_k. The objective we are interested in minimizing at each campaign is the distance between the desired state and the agent states once consensus would be reached after an allocation at time k, which are all equal to scalar $x_k^\infty = v^\top x(t_k)$. This objective writes $J_k^\infty(u(t_k)) = x_k^\infty$ (recalling that $\omega = 0$ and $x_k^\infty \geq 0$). The optimal allocation in this case is then characterized as follows.

Proposition 1: Define the influence power of Agent i as $p_i^k = v_i|\omega - x_i(t_k^-)| = v_i x_i(t_k^-)$. Denote by $\pi_k : \mathcal{V} \to \mathcal{V}$ a bijection which sorts the agents decreasingly by p_i^k, i.e., $p_{\pi_k(1)}^k \geq p_{\pi_k(2)}^k \geq \cdots \geq p_{\pi_k(N)}^k$. Under Assumption 1 the cost $J_k^\infty(u(t_k))$ is minimized by the investment profile:

$$u^*_{\pi(i)}(k) = \begin{cases} \bar{u} & \text{if} \quad i \leq \lfloor \beta_k/\bar{u} \rfloor \\ \beta_k - \bar{u}\,\lfloor \beta_k/\bar{u} \rfloor & \text{if} \quad i = \lfloor \beta_k/\bar{u} \rfloor + 1 \\ 0 & \text{else} \end{cases}$$

Consequently, the trajectory of system (6) is described by:

$$x^\star(t_{k+1}^-) = e^{-Lh}\big(I - U^\star(t_k)\big)x^\star(t_k^-) \tag{7}$$

with $u^\star(t_k)$ designed according to Prop. 1 and using "$\star$" to signify the allocation with perfect knowledge of the graph.

However, when the interaction network is unknown and approximated by the learning algorithm, one uses a Laplacian $\tilde{L}^k$ with a centrality vector $\tilde{v}^k$ to design the budget allocation in Prop. 1. This new budget allocation at time t_k is denoted $u(t_k)$, in contrast to $u^\star(t_k)$. Applying the dynamics of the network (6) with these allocations leads to the trajectory:

$$x(t_{k+1}^-) = e^{Lh}\big(I - U(t_k)\big)x(t_k^-) \tag{8}$$

To find the budget allocation for each campaign β_k, we apply brute-force search from [12]. We denote the overall procedure by $u(t_k) = \mathrm{PLAN}(x(t_k^-), A_\theta, B, k)$ where B is the budget to apply for the remaining horizon of $M-k$ campaigns, and we have already applied Prop. 1 to A_θ to obtain the per-agent allocation.

C. Near-optimal control in unknown graphs

We next combine the algorithms described in the previous two subsections. The main intuition is that we alternate the graph learning and allocation procedures. In-between allocation steps (marketing campaigns) we observe the evolution of the states of the unknown graph and use these observations to approximate it; while at the time of a new allocation, we execute the planning procedure on the approximated graph.

Our method starts with a random initialisation of the parameters Θ, receives observations of the node states for the time steps preceding the first marketing campaign, and uses these observations to approximate the adjacency matrix using Alg. 1 in Sec. III-A. Having obtained an initial approximation of the adjacency matrix, we compute the budget allocation of the first marketing campaign by employing the method described in Sec. III-B. We apply the computed actions to the states of the graph and we continue expanding the dataset of observations up to the next marketing campaign, at which point we find a new approximation of the adjacency matrix using stochastic gradient descent. The algorithm, summarized in Alg. 2, continues in this way until the final campaign.

Algorithm 2 Control in Unknown Graphs

1: $X = [\,]$ ▷ all the observations so far
2: **for** $k \leftarrow 0, M$ **do**
3: $X_{\text{new}} \leftarrow \mathcal{G}$ ▷ observe states for the time interval between two campaigns
4: $X \leftarrow [X|X_{\text{new}}]$ ▷ concatenate observations
5: $A_\theta \leftarrow \mathrm{GLO}\,(X)$ ▷ approximate A
6: $u \leftarrow \mathrm{PLAN}\,(X_{-1}, A_\theta, B, k)$ ▷ plan with A_θ from the last observed state X_{-1}
7: apply action u to the state of $\mathcal{G}$ using (2)
8: $B \leftarrow B - \sum_i u^i$ ▷ remaining budget
9: **end for**

Due to the limited amount of observations, we use very small batch sizes, both aspects adding up to the error when estimating the gradient of the objective. Consequently, this affects the quality of the graph approximation and therefore the accuracy of the planning. Nevertheless, we find that we get a sufficiently good approximation for planning.

IV. Near-optimality analysis

In this section we aim to quantify the optimality loss induced by the learned approximation of the interaction graph. Note that marketing campaigns happen with a sufficiently high frequency that the opinions do not necessarily have time to reach consensus in-between.

The objective is to evaluate the difference between the final opinion $x_M^{\infty\star} = v^\top x^\star(t_M)$ of the network when the marketer knows the graph and the final opinion $x_M^\infty = v^\top x(t_M)$ when the marketer learns the graph. We require that the learning algorithm performs well, in the following sense:

Assumption 2: There exist finite $\epsilon_k > 0$ such that $|v_i - \tilde{v}_i^k| \leq \epsilon_k, \ \forall i \in \{1, \dots, N\}$.

The budget allocation does not depend on the precise value of the centrality vector v but only on the ordering of the influence powers of the agents. In other words, if the re-ordering defined by π_k is the same when replacing v with $\tilde{v}^k$, then $u^\star(t_k) = u(t_k)$, meaning that we obtain exactly the same trajectory for the real opinion dynamics and the ideal one (perfectly known social network). Let $\mathbf{u}^\star$ be the sequence $(u^\star(t_0), u^\star(t_1), \dots, u^\star(t_M))$ of budget allocations over the M stages obtained by applying Prop. 1 to the real graph. Similarly, $\mathbf{u}$ is the sequence $(u(t_0), u(t_1), \dots, u(t_M))$ of allocations over the M stages obtained by applying Prop. 1 to the sequence of learned graphs. Recall that $u_i(t_k) \in \{0, \bar{u}\}$. We will also overload the objective notation J (which earlier only depended on the current-stage allocation $u(t_k)$, "hiding" the dependence on $x(t_k)$ which in turn was generated by all the allocations so far), to now explicitly depend on the full sequence of allocations: $J_M(\mathbf{u}^\star)$ or $J_M(\mathbf{u})$. Note we are interested in this quantity at the final campaign.

We define the errors induced by learning as $e(t) = x^\star(t) - x(t)$ and $e_u(t) = u^\star(t) - u(t)$ and notice that the components of e_u belong to the set $\{-\bar{u}, 0, \bar{u}\}$. To simplify the presentation we disregard the edge case in which one agent receives a fraction of $\bar{u}$. In that case, the corresponding induced error $e_u(t)$ is upper-bounded by $u^\star(t) - u(t)$ and the analysis becomes slightly conservative. Furthermore, if agent i is selected by both v and $\tilde{v}^k$ to (not) be influenced at time t_k, then $u_i^\star(t_k) = u_i(t_k) = \bar{u}$ (or 0) so the i-th component of e_u is zero. A component of e_u is thus nonzero only if the corresponding agent is selected to be influenced by v but not by $\tilde{v}^k$ or vice-versa, in which case its value is either $\bar{u} - 0 = \bar{u}$ or $0 - \bar{u} = -\bar{u}$. Moreover, $\sum_{i=1}^{N} u_i^\star(t_k) = \sum_{i=1}^{N} u_i(t_k)$, yielding $\sum_{i=1}^{N} (e_u)_i(t_k) = 0$. Denote by S^+ and S^- the set of indices for which the components of e_u are $\bar{u}$ and $-\bar{u}$, respectively. Then, the cardinality c of S^+ is equal to the cardinality of S^- to guarantee that $\sum_{i=1}^{N} (e_u)_i(t_k) = 0$. Introduce also $S = S^+ \cup S^-$ and define $\gamma(x^\star(t_k^-)) := 2c\bar{u} \max_{i \in S} x_i(t_k^-)$.

The next result quantifies the loss induced by learning. The main intuition is that this loss is roughly on the order of the learning precision ϵ_k, although it is important to note that $\gamma(x^\star(t_k^-))$ indirectly depends on the learned graph via e.g. S.

Theorem 1: Under Assumption 1 the cost $J_M^\infty(\mathbf{u})$ is a near-optimal approximation of $J_M^\infty(\mathbf{u}^\star)$, in the following sense:

$$|J_M^\infty(\mathbf{u}^\star) - J_M^\infty(\mathbf{u})| \leq \sum_{k=0}^{M} \epsilon_k \gamma(x^\star(t_k^-)) \tag{9}$$

Proof: By definition the cost $J_M^\infty(\mathbf{u}^\star) = v^\top x_M^{\infty\star} = v^\top x^\star(t_M)$ and $J_M^\infty(\mathbf{u}) = v^\top x_M^\infty = v^\top x(t_M)$. Consequently we will evaluate an upper-bound for $|v^\top e(t_M)|$. From (7) and (8) one deduces that

$$\begin{aligned} e(t_{k+1}^-) &= e^{-Lh}\Big[\big(I - U^\star(t_k)\big)x^\star(t_k^-) - \big(I - U(t_k)\big)x(t_k^-)\Big] \\ &= e^{-Lh}\Big[\big(I - U^\star(t_k)\big)e(t_k^-) + e_U(t_k)x(t_k^-)\Big], \end{aligned}$$

where $e_U(t_k) = U^\star(t_k) - U(t_k)$. Thus,

$$\begin{aligned} |v^\top e(t_{k+1}^-)| &= \Big|v^\top e^{-Lh}\Big[\big(I - U^\star(t_k)\big)e(t_k^-) + e_U(t_k)x(t_k^-)\Big]\Big| \\ &= \big|v^\top \big(I - U^\star(t_k)\big)e(t_k^-) + v^\top e_U(t_k)x(t_k^-)\big| \\ &\leq \big|v^\top \big(I - U^\star(t_k)\big)e(t_k^-)\big| + \big|v^\top e_U(t_k)x(t_k^-)\big| \\ &\leq |v^\top e(t_k^-)| + \big|v^\top e_U(t_k)x(t_k^-)\big| \\ &\leq |v^\top e(t_k^-)| + \Big|\sum_{i \in S^+} \bar{u} v_i x_i(t_k^-) - \sum_{i \in S^-} \bar{u} v_i x_i(t_k^-)\Big| \\ &\leq |v^\top e(t_k^-)| + \bar{u} \max_{i \in S} x_i(t_k^-) \Big|\sum_{i \in S^+} v_i - \sum_{i \in S^-} v_i\Big| \end{aligned} \tag{10}$$

For any $i \in S^+, \exists\, j \in S^-$ such that $v_i > v_j$ and $\tilde{v}_i^k < \tilde{v}_j^k$. We also recall that $|v_i - \tilde{v}_i^k| \leq \epsilon_k,\ \forall i \in \{1, \dots, N\}$. Thus,

$$\begin{aligned} v_i - v_j &= v_i - \tilde{v}_i^k + \tilde{v}_i^k - \tilde{v}_j^k + \tilde{v}_j^k - v_j \\ &\leq v_i - \tilde{v}_i^k + \tilde{v}_j^k - v_j \leq 2\epsilon_k. \end{aligned}$$

Consequently, (10) yields

$$\begin{aligned} |v^\top e(t_{k+1}^-)| &\leq |v^\top e(t_k^-)| + 2\epsilon_k \bar{u} c \max_{i \in S} x_i(t_k^-) \\ &= |v^\top e(t_k^-)| + \epsilon_k \gamma(x^\star(t_k^-)). \end{aligned} \tag{11}$$

Applying (11) iteratively one obtains

$$|v^\top e(t_M)| \leq |v^\top e(t_0^-)| + \sum_{k=0}^{M} \epsilon_k \gamma(x^\star(t_k^-))$$

and since $e(t_0^-) = 0$, (9) holds. ∎

Remark 1: If the learning algorithm performs well, the approximation of L improves from one iteration to the next, meaning that $\epsilon_{k+1} \leq \epsilon_k,\ \forall k \in \{0, \dots, M\}$.

V. Empirical results

We evaluate Alg. 2 using synthetic data since this allows for controlling the difficulty of the problem. A classical choice in the literature for studying the kind of scale-free, small-world graphs is the Barabási-Albert model [28]. This has been used extensively to model natural and artificial networks, including social networks [29] and opinion dynamics [30]. In all experiments, we generate fully-connected Barabási-Albert graphs of size 15 and the attachment parameter (AP) set to the (default) value of 2. Increasing the AP value leads to denser graphs which are easier to approximate by our method. In a social network a node is influenced by a relatively small number of neighbours without necessarily influencing them back, making them sparse and directed. Sparsity is controlled by the AP value and we make the graph directed by pruning randomly up to half of the edges. We further enforce the graphs are quasi-strongly connected, by checking the Laplacian has exactly one eigenvalue equal to 0. The states are initialized with a random permutation of evenly spaced values in $[0, 1]$. We track several metrics of interest: the error (difference between the desired and final opinion) $\hat{\delta}$ of the allocation proposed by the approximated algorithm, executed on the true graph the difference between $\hat{\delta}$ and the error of the optimal allocation (when knowing the graph), $\hat{\delta} - \delta^\star$, to which we refer as the sub-optimality; and the largest distance encountered at

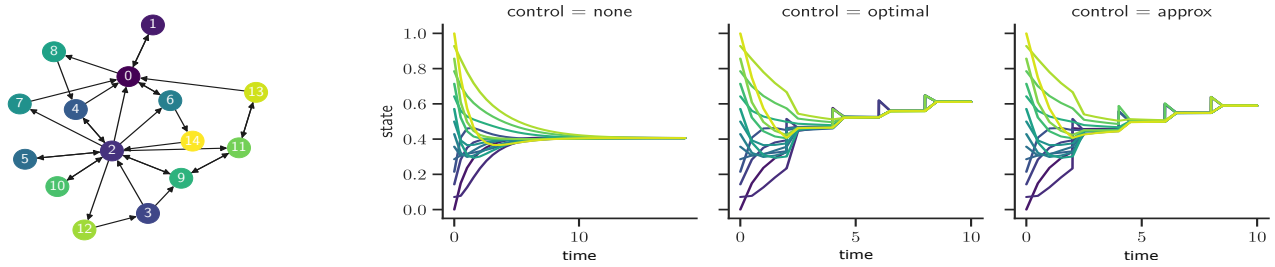

Fig. 1: Near-optimal allocation in a learned graph. Colours map to node indices. Detail of experiment ○ in Fig. 2.

any campaign between the true centrality vector v and the approximated one, $\max_k \|\hat{v}_k - v\|$. Hyperparameters we used are: $N = 15$ agents, $M + 1 = 5$ campaigns, inter-campaign time $h = 2.0$, desired opinion $\omega = 1.0$, per-agent budget $\bar{u} = 0.2$, observation sampling period $s = 0.5$, and learning rate $\eta = 0.001$. At every graph learning phase, we perform up to 50,000 SGD steps or until the mean absolute error between the predicted and the target state reaches a threshold of 0.001.

a) Near-optimal allocation in unknown graphs: we begin with a single graph of 15 nodes randomly generated by the procedure described above. The second panel in Fig. 1 shows the dynamics of the graph without control. The final two panels show the state evolution under opinion control, with the third panel illustrating the baseline operating on the known graph, and the fourth panel showing the dynamics induced by our method, which learns the graph from observations. Since $s = 0.5$, graph learning only receives three observations before the first planning stage and just 12 samples in total by the time of the last campaign. Despite sample scarcity, our method is able to identify the graph and use it for near-optimal allocation of the budget. Our method achieves $\hat{\delta} = 0.341$, compared to the optimal allocation with $\delta^\star = 0.315$.

b) Relation between approximation error and sub-optimality: having shown our method can recover close-to-optimal control in unknown graphs, we now investigate how its sub-optimality scales with the distance between the true centrality vector v and its learned approximation, connected to the analysis in Sec. IV. We generate 500 directed graphs with $N = 15$ using the same procedure as before. For each graph we conduct 10 runs of the algorithm starting from different permutations of the equally-spaced initial states, and different initializations of A_θ. While the algorithm is not overly sensitive to initial conditions, we are interested in marginalizing over the noise inherent in any stochastic optimization process, to more accurately evaluate our method.

Fig. 2 plots the correlation between the sub-optimality $\hat{\delta} - \delta^\star$ and the approximation error of the centrality vector $\max_k \|\hat{v}_k - v\|$. Each data-point is the average of the 10 different initializations. The marginal density estimates along the top and right sides of Fig. 2 show that for most graphs our algorithm is able to learn a good approximation of the underlying adjacency matrix as measured by the error with respect to the centrality vector, leading, in turn, to a small sub-optimality error. Crucially, Fig. 2 supports the conclusions of our analysis in Sec. IV, in that the sub-optimality error is mainly driven by how far the centrality vector of the learned graph deviates from the ground-truth. Indeed, we found no cases of graphs that are very well approximated but for which

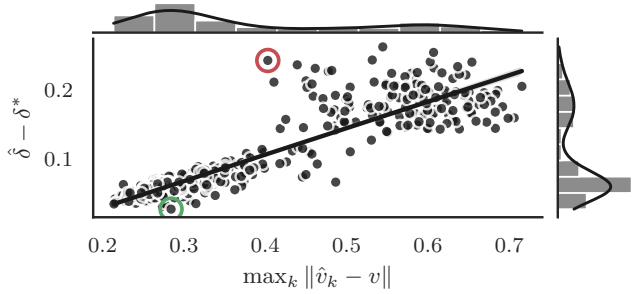

Fig. 2: The sub-optimality (y-axis) decreases with the approximation error of the centrality vector (x-axis). Only for a few graphs the sub-optimality is large and the approximation error is medium, eg.: trial ○. The experiment closest to optimal allocation is marked ○.

allocation fails. We now direct our attention to the few cases of average learning errors driving a larger sub-optimality.

c) The case of sparse centrality vectors: closely looking at the data points in the top region of Fig. 2 reveals that many have very sparse centrality vectors, containing only one or two large, non-zero values. When learning with Alg. 1, in these cases the approximate centrality vectors often contain small values instead of the zero elements in the ground truth. We hypothesize these small non-zero value are an artefact of our gradient-based method that leads to sub-optimal allocations. We therefore run experiments with an additional regularization of our objective that drives A_θ to be sparse, by adding the term $\lambda \sum_{ij}^N \Theta_{ij}^2$ to the objective in (5). Preliminary results suggested slightly larger errors with L1 penalty, so we settled for L2 regularization. The results in Tbl. I suggest that increasing the value of λ drives a decrease in the average sub-optimality error, as well as — more importantly — a reduction in the upper quartile of errors, Q3, associated with the more difficult cases. Fig. 3 further illustrates the importance of sparsity-inducing regularization when learning the graph. For $\lambda = 0.5$, many graphs that had medium approximation error but relatively large sub-optimality now correlate strongly with the approximation error.

TABLE I: Effect of L2 regularization on sub-optimality

λ	min	max	mean	median	Q3
0.01	0.030	0.258	0.098	0.069	0.153
0.10	0.025	0.253	0.098	0.068	0.152
0.50	0.026	0.248	0.089	0.065	0.113

d) Random policy and noisy observations: a control law that optimises for consensus can induce a data distribution unconducive to graph identification. We set up an experiment in which the control is performed with the random policy.

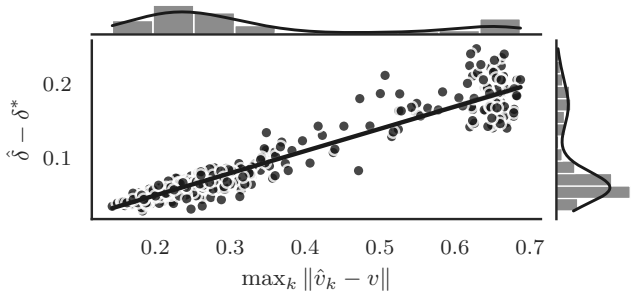

Fig. 3: Sub-optimality and approximation error correlation, $\lambda = 0.5$

The aim is to check whether a random control is able to correct the data distribution to such extent that the graphs are better approximated. Fig. 4 shows no indication this is the case. Further experiments with noisy observations (which also indirectly make the learning-based inputs more informative) fail to show meaningful improvements.

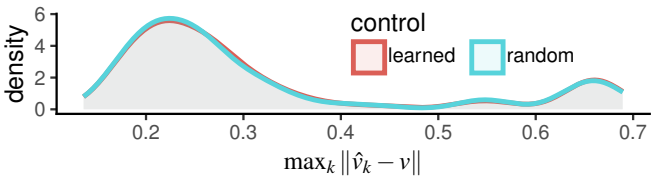

Fig. 4: Approximation error with learned and random control.

e) Scalability: the graph learning component of our method takes advantage of the good scalability properties of backpropagation and SGD. A comparison on graphs with a doubling number of nodes, $N \in \{15, 30, 60, 120, 240\}$, reveals only a linear increase in the time required to identify the graph: $13.1, 13.1, 16.3, 35.3$ and 95.6 seconds.

VI. Conclusion and Future Work

We proposed and characterized an algorithm to influence opinion dynamics that works on observations from an unknown graph. We proved that the sub-optimality of our method is proportional to the approximation error of the learning algorithm. We confirmed these findings empirically and introduced an extra regularization term that improves performance. We focused here on graphs with linear dynamics. Future investigation could characterize the conditions for identifiability and convergence of our method, but also extend this work to certain classes of non-linear interactions between the agents.

References

[1] F. Bullo, J. Cortés, and S. Martinez, *Distributed Control of Robotic Networks. A Mathematical Approach to Motion Coordination Algorithms*. Princeton University Press, 2009.

[2] M. Mesbahi and M. Egerstedt, *Graph theoretic methods in multiagent networks*. Princeton University Press, Princeton, NY, 2010.

[3] A. V. Proskurnikov and M. Cao, *Consensus in Multi-Agent Systems*. John Wiley & Sons, 2016, pp. 1–16.

[4] M. H. DeGroot, "Reaching a consensus," *Journal of the American Statistical Association*, vol. 69, no. 345, pp. 118–121, 1974.

[5] N. E. Friedkin and E. C. Johnsen, "Social influence and opinions," *Journal of Mathematical Sociology*, vol. 15, pp. 193–206, 1990.

[6] R. Hegselmann and U. Krause, "Opinion dynamics and bounded confidence models, analysis, and simulation," *Journal of Artificial Societies and Social Simulation*, vol. 5, no. 3, 2002.

[7] I.-C. Morărescu and A. Girard, "Opinion dynamics with decaying confidence: Application to community detection in graphs," *IEEE Trans. on Automatic Control*, vol. 56, no. 8, pp. 1862 – 1873, 2011.

[8] C. Altafini, "Consensus problems on networks with antagonistic interactions," *IEEE Trans. on Automatic Control*, vol. 58, pp. 935–946, 2013.

[9] P. Bonacich and P. Lloyd, "Eigenvector-like measures of centrality for asymmetric relations," *Social Networks*, vol. 23, pp. 191–201, 2001.

[10] S. Martin, I.-C. Morărescu, and D. Nešić, "Consensus and influence power approximation in time-varying and directed networks subject to perturbations," *International Journal of Robust and Nonlinear Control*, vol. 29, no. 11, pp. 3485 – 3501, 2019.

[11] N. Booth and J. A. Matic, "Mapping and leveraging influencers in social media to shape corporate brand perceptions," *Corporate Communications: An International Journal*, vol. 16, no. 3, pp. 184–191, 2011.

[12] I.-C. Morărescu, V. Varma, L. Buşoniu, and S. Lasaulce, "Space-time budget allocation policy design for viral marketing," *Nonlinear Analysis: Hybrid Systems*, vol. 37, p. 100899, 2020.

[13] D. R. Alkhorshid, E. S. Tognetti, and I.-C. Morărescu, "Saturated control of consensus value under energy and state constraints in multi-agent systems," *Automatica*, vol. 169, 2024.

[14] C. Bernardo, L. Wang, M. Fridahl, and C. Altafini, "Quantifying leadership in climate negotiations: A social power game," *PNAS*, 2023.

[15] D. Materassi and M. V. Salapaka, "On the problem of reconstructing an unknown topology via locality properties of the wiener filter," *IEEE Trans. Autom. Control.*, vol. 57, no. 7, pp. 1765–1777, 2012.

[16] P. M. J. V. den Hof, A. G. Dankers, P. S. C. Heuberger, and X. Bombois, "Identification of dynamic models in complex networks with prediction error methods - basic methods for consistent module estimates," *Autom.*, vol. 49, no. 10, pp. 2994–3006, 2013.

[17] H. H. Weerts, A. G. Dankers, and P. P. van den Hof, "Identifiability in dynamic network identification," *IFAC-PapersOnLine*, vol. 48, 2015.

[18] M. Gevers and A. S. Bazanella, "Identification in dynamic networks: Identifiability and experiment design issues," in *54th IEEE Conference on Decision and Control*, 2015, pp. 4005–4010.

[19] S. Guo, H. Xu, G. Xie, D. Wen, Y. Huang, and P. Peng, *Reinforcement Learning-Based Consensus Reaching in Large-Scale Social Networks*.

[20] D. L. Mingwei Wang and Z. Xu, "Consensus achievement strategy of opinion dynamics based on deep reinforcement learning with time constraint," *Journal of the Operational Research Society*, vol. 73, no. 12, pp. 2741–2755, 2022.

[21] V. S. Borkar and A. Reiffers-Masson, "Opinion shaping in social networks using reinforcement learning," *IEEE Trans. on Control of Network Systems*, vol. 9, pp. 1305 – 1316, 2022.

[22] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *2nd Int. Conf. on Learning Representations*, 2014.

[23] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Int. Conf. on Learning Representations*.

[24] W. L. Hamilton, "Graph representation learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 14, 2020.

[25] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *6th Int. Conf. on Learning Representations, ICLR, April 30 - May 3*, 2018.

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Int. Conf. on Learning Representations*, 2015.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *IEEE Int. Conf. on Computer Vision*, 2015, pp. 1026–1034.

[28] A. Ĺaszló Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286 5439, pp. 509–12, 1999.

[29] A. Barabási, H. Jeong, Z. Néda, E. Ravasz, A. Schubert, and T. Vicsek, "Evolution of the social network of scientific collaborations," *Physica A: Statistical Mechanics and its Applications*, vol. 311, no. 3, 2002.

[30] D. S. M. Alencar and et al, "Opinion dynamics systems on Barabási-Albert networks: Biswas-Chatterjee-Sen model," *Entropy*, vol. 25, no. 2, p. 183, 2023.