NetGist: Learning to generate task-based network summaries

Sorour E. Amiri Bijaya Adhikari Aditya Bharadwaj B. Aditya Prakash
Department of Computer Science, Virginia Tech
Email: [esorour, bijaya, adb, badityap]@cs.vt.edu

***Abstract*—Given a network, can we visualize it for any given task, highlighting the important characteristics? Networks are widespread, and hence summarizing and visualizing them is of primary interest for many applications such as viral marketing, extracting communities and immunization. Summaries can help in solving new problems in visualization, sense-making, and in many other goals. However, most prior work focuses on generic structural summarization techniques or on developing specific algorithms for specific tasks. This is both tedious and challenging. As a result, for several popular tasks, there do not exist readymade summarization methods.**

In this paper, we explore a promising alternative approach instead. We propose NetGist, a framework which automatically learns *how* to generate a summary for a given task on a given network. In addition to generating the required summary, this also allows us to *reuse* the learned process on other similar networks. We formulate a novel task-based graph summarization problem and leverage reinforcement learning to design a flexible framework for our solution. Via extensive experiments, we show that NetGist robustly and effectively learns meaningful summaries, and helps solve challenging problems, and aids in complex task-based sense-making of networks.

I. Introduction

Networks are common and occur in many different domains such as social networks, protein-protein interaction, communication networks and so on. Understanding these networks is essential for many tasks, such as viral marketing, community detection, anomaly identification and finding different patterns. One approach is to construct *network summaries* by merging (grouping) nodes and edges into super-nodes and super-edges.

Moreover, *task-based* summaries can help in solving various problems. For instance, identifying epidemiologically relevant anomalies in dynamic human-mobility networks [12]. There are ways to identify anomalies on a network, but in a dynamic network, it is not straightforward how to use the information of previous snapshots to detect anomalies effectively and efficiently. On the other hand, if we learn a function that summarizes each snapshot of the network and highlights the anomalies on them, it would make it easier to detect anomalies on the dynamic graph. So, given a task-based summary for the task, we can potentially discover epidemiologically relevant anomalies in dynamic human-mobility networks. Fig. 1 shows the summary networks using a learned summarization function for three different snapshots of such networks in the context of interactions between different departments of a company; the anomalous nodes are shown in black; as discussed in experiments later, the summaries consistently identify anomalies correctly matching the given ground-truth.

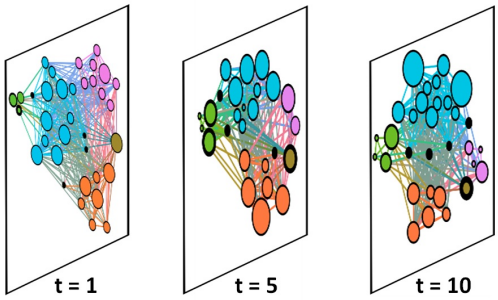


Fig. 1: Summary graph of three snapshots of Work-Place data. Black nodes are anomalies.

Similarly, task-based summaries can help in sense-making and visualizing hidden patterns in networks, e.g., for identifying bridges in different types of networks. Finally, task-based summaries can also help in getting a better quality solution for tasks with known algorithms [17]. Usually, past work has looked into constructing summaries preserving important characteristics of the network (e.g., structural characteristics).

Based on previous work, we make three observations. (1) Different tasks give rise to different network summaries. For example, the desired summary for the community detection task (which tries to find cohesive regions) is very different from the summary for a task like influence maximization (which seeks to find central nodes). (2) There are tasks with no summarization algorithms such as the anomaly detection problem discussed above, traveling salesman problem, etc. Also, new tasks are getting defined on networks which may make it impractical to develop specific algorithms for each of them. (3) In real-world, we usually solve the same task repeatedly on similar networks that belong to the same distribution. For example, generating layouts for similar graphs is helpful in context of protein networks [4], network-traffic data [28] etc. Similarly, an advertiser may target the same social network repeatedly with probabilistic influence patterns [10]. Samples of networks from the same distribution are frequently observed in intelligence analysis [8], epidemiology [3] and many other settings.

Hence, leveraging these three observations, in such scenarios, the inherent similarity between networks and the variety of tasks opens a space for *learning* task-based network summaries for networks coming from a distribution. This motivates

the following informal learning problem:

Problem: Task-based Summaries. Given a problem $Prob$, and a distribution D of network instances, can we learn *how* to generate meaningful network summaries that generalizes to unseen instances from D?

Leveraging recent advances in reinforcement learning and deep learning [7], [20], we develop NetGist, a flexible approach which automatically learns *how* to generate a summary for a given set of tasks. To the best of our knowledge, past work has not looked into this novel problem. Our contributions in this paper are, (I) Problem formulation (II) Designing effective learner and (III) Extensive Experiments.

The rest of the paper is organized in the usual way.

II. Problem Formulation

Following prior work in network summarization [23], [11], [24], in a summary, it is natural to *group* ('merge') similar nodes to construct a graph of 'super-nodes' and 'super-edges' with a smaller size than the original network. In this way, each super-node represents homogeneous regions [11] in some sense and the connection between super-nodes highlights the overall structure of the network and important regions of the original graph [23]. Intuitively, our goal is to find a good network summary which guides us to find the solution for a given task.

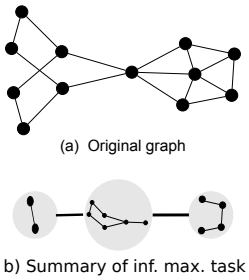
Tasks: The first question is what kind of tasks we want to handle? As our first step towards learning task-based summaries for networks, we choose to solve a set of problems we call **G**raph **O**ptimization **P**roblems (GOP) which includes many popular graph mining problems. Suppose we have a graph $G(V, E)$ where V and E are sets of nodes and edges of the network. Also, assume Θ is the set of parameters of the given task. In a problem $prob \in$ GOP, the goal is to select $O \subseteq U$ (i.e. a set of objects O from the universe U of objects determined by the problem), that maximizes some quality function $F_{prob}(O; G, \Theta)$. For all GOP problems, U is some set of objects defined over the graph G (e.g. nodes, edges, subgraphs, etc.). Note F_{prob} maps a sets of objects to a real value based on the given task $prob$ and its parameters Θ on the graph G. We formally define the set of graph optimization problems as follows:

Definition 1: Graph Optimization Problem (GOP)
Given a graph $G(V, E)$, a set of input parameters Θ and a quality function $F_{prob}(O; G, \Theta) \in \mathbb{R}$ and set U,
Find the best set of objects O^* such that,

$$O^* = \arg\max_{O \subseteq U} F_{prob}(O; G, \Theta) \tag{1}$$

Note, many combinatorial problems such as Minimum Steiner Tree, Maximum Clique, etc. [15] that defined over graphs can naturally be written as a GOP problem. So, the set of GOP problems is broad and includes many problems of interest. We use two common GOP problems to showcase our method: Influence maximization and Community detection.

What is a good summary? To generate any summary we need to (Q 1) Evaluate the structure of the summary graph; and (Q 2) Identify homogeneous regions. Our main insight is that both of these issues are *task-dependent*. We show the summary for the Influence maximization problem in figure (b). It shows the given tasks guide us to both evaluate the structure and characterize homogeneous regions in the summary graph.



The summary highlights how different super-nodes (regions) are connected to each other based on the given task and highlight the central region of the graph (Q1). Also, each super-node in the summary shows a region in the graph which contains nodes with the same role (effect on the solution) based on the given task. So, the central node in the graph is central in the central region as well (Q 2). Note that these properties also lead to visually appealing summaries, as they highlight the structural characteristics of the graph important to the given task.

Measuring Summary Quality is challenging as the answers are inter-dependent. Defining what characterizes a homogeneous region necessarily impacts the right measure to evaluate the overall structure (and vice versa). Our idea is to identify properties and develop a procedure which answers both questions simultaneously.

We first formalize the notions of a 'good summary'. Let $G^*(V^*, E^*)$ be an ideal summary of the original network $G(V, E)$ for an arbitrary GOP problem $prob$ with parameters Θ. Also let O^* be the optimal solution for $prob$ on G. Then we want the following properties to hold for G^*.

Property 1: Let $O^{'*} = \arg\max F_{prob}(O^{'}, G^*, \Theta^*)$. Then, $\forall_{o \in O^*}, \exists H(o) = x \in O^{'*}$.

Property 1 states that there exists a mapping H which maps all objects in the optimal solution for $prob$ on G to an element of the optimal solution for the $prob$ on G'. This property indicates that G and G' have structural similarity w.r.t. $prob$.

Property 2: For any two pairs of nodes v_i and v_j in an arbitrary super-node x in the summary,

$$F_{prob}(\{v_i\}, G, \Theta_x) > F_{prob}(\{v_j\}, G, \Theta_x) \implies F_{prob}(\{v_i\}, x, \Theta) > F_{prob}(\{v_j\}, x, \Theta)$$

Property 2 implies that the order of role of nodes inside each super-node, is preserved. It means the nodes in each super-node have a homogeneous role in determining the solution and they lead to the right solution as their order is preserved.

These two properties indicate if an ideal summary G^* for G w.r.t. to GOP problem $prob$ exists, we can reconstruct the optimal solution to $prob$. So, to determine how far a summary is from G^*, we propose a divide-and-conquer Algorithm 1 to measure the quality of any summary G' of G: first solve $prob$ on G' and recursively solve for $prob$ on each solution obtained in the first step. And intuitively the quality of the solution tells us how good the summary is. Lemma 1 indicates that the Divide and Conquer framework we propose returns optimal score for the summary network that satisfies both the properties.

Lemma 1: Algorithm 1 on a summary network G' returns the optimal solution O^* and optimal score $F_{prob}(O'^*; G', \Theta)$ if G' satisfies both Properties 1 and 2.

Note that validating whether a summary satisfies Property 2 is computationally expensive as it involves repeatedly measuring F_{prob}. Our approach avoids this step and still manages to output the ideal solution and the corresponding optimal score F_{prob} with respect to $prob$ if the summary network satisfies both the properties.

Algorithm 1 Divide and Conquer

Require: $G(V, E)$, $G'(V', E')$, $prob \in$ GOP
1: $O'^* \leftarrow \emptyset$
2: //Divide
3: $\Phi^* \leftarrow \arg\max F_{prob}(\Phi; G', \Theta)$
4: // Conquer
5: **for** $\phi \in \Phi^*$ **do**
6: $G_\phi \leftarrow$ subgraph of ϕ in G
7: $\Theta_\phi \leftarrow$ sub-parameters that related to ϕ
8: $o^*_\phi = \arg\max F_{prob}(o_\phi; G_\phi, \Theta_\phi)$
9: $O'^* \leftarrow O'^* \cup o'^*_\phi$
10: return O'^* and $F_{prob}(O'^*; G, \Theta)$

Problem definition: We are now ready to define our problem formally. Suppose we have a network $G(V, E)$ where V and E are sets of nodes and edges of the network respectively. Also, suppose we are given a graph optimization problem $prob \in$ GOP (Def. 1) which asks to find the best set of objects O^*. The parameters of $prob$ and the graph G are drawn from a probability distribution D (i.e. $(G(V, E), \Theta) \sim D$). For example, D can be the set of probabilistic influence patterns (as discussed in the introduction). We want a summary network $G'(V', E')$ for any graph $G(V, E)$ drawn from D such that $|V'| = \alpha \cdot |V|$, where α is the 'reduction factor'. It is expensive to generate the high-quality summary for all the graphs in D. Hence, our idea is to learn the *process* of learning the summaries instead of merely the final summary graphs. Our goal is to learn a graph summarization process such that solving the given problem on G' using our divide-and-conquer strategy (Algorithm 1) results in a set O'^* such that $F_{prob}(O'^*; G, \Theta)$ is similar to the $F_{prob}(O^*; G, \Theta)$. Formally:

Problem 1: Task-Based Network Summarization (TBNS)
Given a graph optimization problem ($prob \in$ GOP), a distribution of problem instances D, and a reduction factor $0 < \alpha \leq 1$.
Learn a graph summarization process such that for any $(G(V, E), \Theta) \sim D$,

$$\max \rho = \mathbb{E}_{(G,\Theta)\sim D}\left[\frac{F_{prob}(O'^*; G, \Theta)}{F_{prob}(O^*; G, \Theta)}\right] \quad (2)$$

where, O'^* is the solution of the given problem $prob$ on the summary graph G' using Alg. 1.

Comments: Generalizing a learned model to unseen instances is important from the learning perspective. A good model which is trained on a particular 'training' data is expected to do reasonably well in the 'test' data drawn from the same distribution [19]. Hence, our ultimate goal is to maximize the expected ratio ρ over all the instances (including the unseen ones) of (G, Θ) drawn from D. Note that in some real-world scenarios, D can be a single (G, Θ).

III. Our Method

Next, we show how to solve our TBNS problem. We want to precisely learn the steps to be taken for the summarization process, so that we can re-apply the same approach on other graphs from the same distribution. A basic summarization step we take is some sort of a 'merge' operation. The merge operation basically groups nodes in the original network and results in a so-called 'super-node' in the summary network. The question at hand is that once the summarization step is decided, how do we measure the 'goodness' of each step and how do we obtain a quality summary? Reinforcement Learning (RL)—more specifically Q-learning [25]—is a natural solution to the above questions. In general, RL methods learn to take an '*action*' in an '*environment*' to maximize cumulative '*reward*' based on a '*transition*' function.

Recently, the success of deep reinforcement learning [20], where convolutional neural networks are used to learn a representation of 'states', in solving various AI tasks has gained much attention. It is known to perform better and converge faster than the traditional reinforcement learning. Hence, we propose to leverage a deep reinforcement learner to solve our TBNS problem.

A. Overview of NetGist framework

Algorithm 2 Overview of NetGist

Require: $prob \in$ GOP, α, D
1: Randomly Initialize the deep Q-learning parameters
2: **for** $i = 1$ **To** num of samples **do**
3: Sample $G(V, E)$ and Θ drawn from D
4: // learning how to summarize
5: **for** episode=1 **to** T **do**
6: $G'(V', E') \leftarrow G(V, E)$
7: **while** $|V'| > \alpha \cdot |V|$ **do**
8: Merge $G'(V', E')$ // (See Section III-B)
9: // Evaluate
10: $O', F_{prob} \leftarrow$ DivideAndConquer($G, G', prob$) Alg. 1
11: // Optimize
12: Update the deep Q-learning parameters to yield better summary (see Section III-D)
13: **Return** the trained model (which solves the TBNS problem)

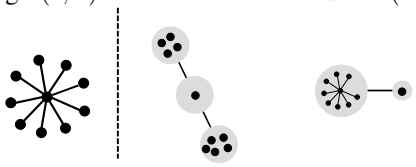
We now give an overview of our NetGist approach (Algorithm 2). Overall, in our deep reinforcement learning paradigm, the initial 'state' is the network $G(V, E)$ drawn from D and each 'action' the learner takes reduces the size of original network G to produce a new 'state', a summary network $G'(V', E')$. The learner repeatedly takes actions till the summary network $G'(V', E')$ is of desired size. Next, we evaluate the $prob$ on G' using Algorithm 1. Based on the quality of the solution, the learner's parameters are updated. The learner repeats the process on more samples from D until it learns to summarize any $(G, \Theta) \sim D$ to G', such that ρ is maximized. To formalize our framework, we need to answer some questions and make design choices. These question

are: **Q1** How to define the universe of meaningful actions that summarizes any network $G(V, E)$ drawn from D into $G'(V', E')$? **Q2** What is the universe of all possible 'states'? and **Q3** What are the reward, policy, transition functions and the overall learning procedure? We answer these questions next.

B. Q1. Universe of Actions

An action a taken at state s basically gives us the next state s'. Since our initial state is a network G from D and our goal is to summarize it, each action a should reduce the size of the network G. As briefly indicated above, we basically define our action as a merge operation. Specifically, a merge operation on a node pair $\{a, b\}$ in the network G would result in a new *super-node* c in G'. Moreover, new edges (v, c) for all nodes v which are neighbors of either nodes being merged would be added to G'. It is important to note that merging different types of nodes is useful for different graph optimization problems. For example, for tasks like community detection, it may be more meaningful to merge nodes which have an edge connecting them to ensure that densely connected nodes are grouped together (Fig. below (c)). In contrast, for task like immunization, it may be more meaningful to merge nodes around a central node to ensure that the central node, which is more likely to be in the solution, remains central in the summary network (Fig. below (b)). Therefore, we allow merging nodes that are any distance apart. To formalize this intuition, we define a k^{th}-order merge as follows:

Definition 2: (k^{th}-Order Node Pair Merge) A k^{th}-Order node pair merge operation merges nodes a and b into a new node c, such that $a \in V$, $b \in V$, and $d(a, b) = k$. We add new edge (c, k) for all the nodes $k \in NB(a) \cup NB(b)$.



In a single action a, we first decide on the order k of the merge operation. Then we merge all possible node pairs, which can be merged with a k^{th}-order merge. Formally, our universe of actions is defined as follows:

Definition 3: **Universe of Actions** The universe of actions A in NetGist is a set of all possible k^{th}-order merges.

Note that our single action is a set of k^{th}-order merge operations, for a fixed order k. Hence, we usually merge more than a single pair of nodes. This ensures that our universe of actions is broad enough to incorporate meaningful merge operations for different tasks and yet restrictive enough to be tractable for learning purposes.

Lemma 2: k^{th}-Order node pair merge is order sensitive. (Proof omitted for the lack of space)

Given the lemma 2 the order of taken actions is essential in generating the summaries. Hence, we should learn the sequence of actions as well as the set of actions to take.

C. Q2. Universe of States

The initial state in our RL framework is any network $G(V, E)$ drawn from D and each action the learner takes, results in a summary network $G'(V', E')$. Recall that our one action is a set of k^{th}-order merge operations. Any state s other than the initial state is a result of one or more actions on the initial state, the network G. Hence the universe of states, which includes the original network as well, is defined as follows:

Definition 4: **Universe of States** The universe of states S in NetGist is a set of networks $G'(V', E')$, such that G' is a result of zero or more merges on the network G from D.

Note that our universe of states does not include all possible summary networks of network G from D. This is because our action merges multiple nodes at once. This reduces the size of the universe of states, which in turn helps in learning. However, as discussed earlier, this set is still large enough to explore meaningful summaries across the search space.

D. Q3. Reward, Policy, Transition and the Learning Procedure

Having described the states and actions earlier, now we can describe our method to learn meaningful summary.

1) Reinforcement learning.: Our idea is to use deep Q-learning as it is an off-policy RL which known to be more sample efficient than the policy gradient methods [25].

We define the reward to be -1 for a state s, unless it is a terminal state. A terminal state in our case is a summary network $G'(V', E')$ which has the desired size. Formally, we define our reward function as follows:

$$r(s, a) = \begin{cases} \max\limits_{O \subseteq U} F_{prob}(O; s_{next}, \Theta) & \text{if } s_{next} \text{ is a terminal state} \\ -1 & \text{otherwise} \end{cases} \quad (3)$$

In Eq. 3 we compute $F_{prob}(O; s_{next}, \Theta)$ on the summary s_{next} with our divide and conquer framework in Alg. 1.

Our goal is to learn optimal Q-value(s, a) which estimates the cumulative reward after taking action a at state s [25] and select actions that result in the highest cumulative reward.

2) Learning algorithm: Next, we elaborate on the whole learning pipeline. Note that our pipeline learns the summarization procedure for graphs in D such that it is generalizable to the unseen graphs. First we define how to estimate Q-value(s, a), using the Q-learning algorithm. In order to have an end-to-end framework to learn the optimal Q-value(s, a) we combine CNNs [13] with Q-learning as often done in literature. CNN helps us to have a compact representation of each state s in the universe of states.

In our deep RL framework the input state s is fed into the CNN layer. The CNN outputs a feature representation of the state, which is then fed into a fully connected Feed Forward Network, FFN. The output layer of the FFN is d-dimensional, where d is the number of possible actions. Each i^{th} entry in the output vector represents the predicted Q-value function Q-value(s, a_i) for the state s and the action a_i.

Lemma 3: Time complexity of NetGist is $O(Itr \cdot VE \log V)$, where Itr is the number of iterations.

IV. Empirical Study

We design various experiments to evaluate NetGist. Our code is available for research purposes[1]. We set $\alpha = 0.3$ for

[1] http://github.com/SorourAmiri/NetGist

all our experiments. However, we examined the robustness of NetGist to changes in α and Θ as parameters of the TBNS problem and found it to be stable over wide ranges.

Data. We collected a number of datasets from various domains for our experiments. See Table I for details.

		Dataset	#Nodes	#Edges	Description
Synthetic	1	ER [5]	20-2000	50-10000	Erods-Reyni graph
	2	BA [5]	20-2000	40-4000	Barabasi-Albert preferential attachment graph
	3	Block Model [5]	20-2000	40-30000	Block models graph
Real-world	4	Karate Club	34	78	A social network among members of a karate club
	5	Work-Place [2]	92	755	mobility networks
	6	High-School [2]	327	5,818	mobility networks
	7	Political Blogs	1,490	16,783	Network of hyper-links between web-blogs on US politics
	8	Facebook [1]	4,039	88,234	A social network
	9	As-Oregon [1]	7,000	15,743	An autonomous systems peering information network

TABLE I: Datasets details.

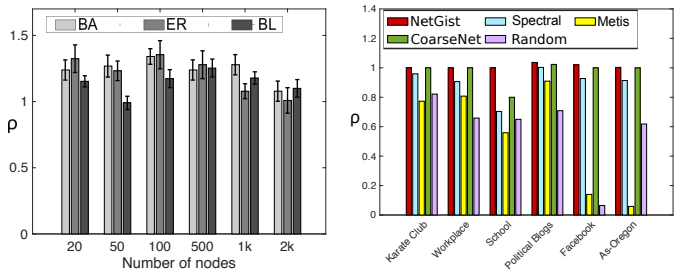
Baselines. We compare the performance of NetGist with other summarization techniques: (I) Random selects random node-pairs and merges them. (II) CoarseNET [23] summarizes the network while ensuring that the propagation properties are preserved. (III) METIS [16] and Spectral [26] partition the network G while ensuring the summary is useful for the Community detection task. We take each partition as the super-node of the summary and connect two super-nodes if there is at least one edge between the nodes in them.

A. Q1. Quality of NetGist on distribution D

Here we evaluate the quality of NetGist on various networks and distributions and show a sample visualization.

Synthetic and real-world networks: We use popular synthetic networks (Erdos-Renyi (ER), Barabasi-Albert (BA) and Block Models) and different real datasets to evaluate NetGist on distribution D. For synthetic graphs we trained NetGist on 80 randomly sampled networks from a distribution and test on 20 previously unseen networks from the same distribution. The Avg. ρ and s.t.d. on the test with respect to Influence maximization tasks are shown in Fig. 2. Also, Fig. 2b shows the result on real-world datasets. In overall, NetGist generalizes well and gets high ρ for all the network distributions with various sizes and even outperforms baseline methods specifically designed for the task, implying NetGist summaries can help solve GOP tasks. (Community detection task is omitted due to lack of space).

Sample Visualization: Here we visualize our learnt summaries from the well-known Karate Club network (Figure 3) to demonstrate the difference between various tasks. The Influence maximization task-based summary consists of two super-nodes which have considerably higher degree than the rest of the network. These two super-nodes contain the optimal solution, while the rest of the super-nodes consists of homogeneous nodes with low influence. For the Community detection task as well, we see that two super-nodes in the summary highlight the homogeneous regions well (which match closely with the ground truth communities in the network). We show more such examples in the appendix.



(a) Influence maximization (b) Influence maximization

Fig. 2: (a) NetGist obtains high ρ consistently indicating that it learns well how to summarize networks for a given task. NetGist outperforms all the baselines and performs consistently for all the tasks and datasets. Note, the quality of NetGist solutions is equal or even better than the algorithms that are designed for the given task across all datasets.

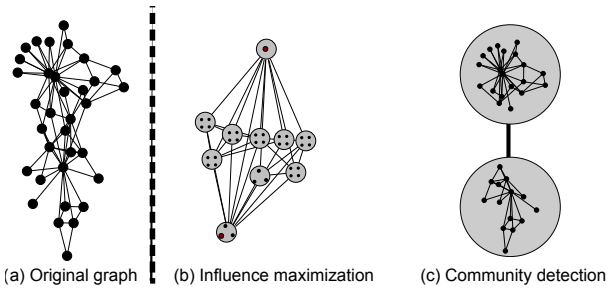


Fig. 3: Karate Club. (a) Original graph (b and c) Summary graphs of two GOP tasks.

B. Q3. Detecting anomalies on mobility graphs

Here we leverage NetGist for the application of discovering epidemiologically relevant anomalies in the Work-Place networks, which are based on a mobility log of employees in a company with five departments. The dataset has been studied before for vaccination problems [12]. Due to variation in organizational roles and mobility patterns, some nodes may play anomalous roles in any epidemic outbreak. Such nodes are difficult to mine due to the temporal nature of data and as different nodes could be anomalous at different times.

Our main idea here is to leverage the inherent uncertainty in the data (the fact that any two people interacting at a given time is probabilistic) to learn the summarization process specific to the epidemiologically related influence maximization task. Once trained, we summarize the contact networks generated from the mobility log by applying the learned policy. Finally, we use the summaries to identify the anomalous nodes. Note that independently summarizing each network snapshot will

not be useful as it discards the relationship between the snapshots (as we observe in our experiments as well).

NetGist runs in less than 0.5s per iteration. In Fig. 1, we show the learnt summary networks for three different snapshots. Color represents different departments and the size of the 'super-nodes' is proportinal to the number of nodes inside them. First note that all the departments are clearly visible in the summaries. This highlights that NetGist is able to capture the temporal stability of the departments in the dataset. Secondly, we take the singleton (unmerged) nodes connecting different departments as the anomalous nodes (shown in black). Interestingly the nodes we identify as anomalous (80, 131, 751, 222, 134) have also been labelled as the 'linkers' in [12]. We consistently find some of the linkers in most time-stamps highlighting the stability of linkers over time as discussed in [12]. On the other hand, we also find different nodes as linkers in different timestamps highlighting that different nodes play anomalous roles at different times. Hence, the visualization of the summaries helps in sense-making of the anomalous nodes and the role they play, which is not obvious from the original networks.

V. Related Work

We briefly discuss related work from multiple areas here.

Network Summarization. There are many summarization methods in many domains [17] for community detection problem [16], diffusion-related problem [23] and network sparsification [18]. We add a new direction to this line of work, by aiming to *learn task-based* summaries automatically instead of *designing* a different algorithm for each one separately.

Deep Learning for Graphs. Researchers have exploited deep learning for various graph mining tasks like node embedding [14], graph classification [21], and graph kernels [27] and structural data embedding [9]. As far as we know, we are the first to leverage deep learning for network summarization.

Learning to learn on Graphs. The field is seeing increasing recent interest. Methods include RNN based meta heuristic for shortest paths [6], neural networks for quadratic problems over graphs [22], reinforcement learning for the traveling salesman problem [7], RL to solve combinatorial problems [10], and so on. In contrast, we present deep reinforcement learning for network summarization in task dependent manner.

VI. Conclusions

In this paper, we generalize over multiple threads of prior work and propose a novel Task-Based Network Summarization (TBNS) problem to automatically learn how to generate task specific network summaries. We proposed an effective method NetGist by leveraging the deep Q-learning framework. As shown by our experiments on both real and synthetic data, NetGist is able to learn meaningful summaries for various tasks and generalize them to the unseen instances. Also, NetGist helps in complex sense-making and anomaly-detection in temporal networks. We can explore using this framework for even more tasks and applications that can be transformed to GOP. Generalizing the set GOP (like to include graph alignment) and speeding up NetGist by leveraging deep network embedding would also be fruitful.

Acknowledgment. This paper is based on work partially supported by the NSF (CAREER IIS-1750407), the NEH (HG-229283-15), ORNL, National Cancer Institute of the National Institutes of Health (UH2CA203768), and a Facebook faculty gift.

References

[1] http://snap.stanford.edu.
[2] http://www.sociopatterns.org/.
[3] B. Adhikari, Y. Zhang, S. E. Amiri, A. Bharadwaj, and B. A. Prakash. Propagation-based temporal network summarization. *TKDE*, 2018.
[4] G. Amitai, A. Shemesh, E. Sitbon, M. Shklar, D. Netanely, I. Venger, and S. Pietrokovski. Network analysis of protein structures identifies functional residues. *Journal of molecular biology*, 2004.
[5] A.-L. Barabási et al. *Network science*. Cambridge university press, 2016.
[6] A. Bay and B. Sengupta. Approximating meta-heuristics with homotopic recurrent neural networks. *arXiv preprint arXiv:1709.02194*, 2017.
[7] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.
[8] T. Coffman, S. Greenblatt, and S. Marcus. Graph-based technologies for intelligence analysis. *Communications of the ACM*, 2004.
[9] H. Dai, B. Dai, and L. Song. Discriminative embeddings of latent variable models for structured data. In *ICML*, 2016.
[10] H. Dai, E. B. Khalil, Y. Zhang, B. Dilkina, and L. Song. Learning combinatorial optimization algorithms over graphs. *NIPS*, 2017.
[11] I. S. Dhillon, Y. Guan, and B. Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *PAMI*, 2007.
[12] M. Génois, C. L. Vestergaard, J. Fournet, A. Panisson, I. Bonmarin, and A. Barrat. Data on face-to-face contacts in an office building suggest a low-cost vaccination strategy based on community linkers. *Network Science*, 2015.
[13] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio. *Deep learning*. MIT press Cambridge, 2016.
[14] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *KDD*, 2016.
[15] D. S. Johnson. Approximation algorithms for combinatorial problems. *Journal of computer and system sciences*, 1974.
[16] G. Karypis and V. Kumar. Multilevel algorithms for multi-constraint graph partitioning. In *Proc. SC98*, 1998.
[17] Y. Liu, A. Dighe, T. Safavi, and D. Koutra. A graph summarization: A survey. *arXiv preprint arXiv:1612.04883*, 2016.
[18] M. Mathioudakis, F. Bonchi, C. Castillo, A. Gionis, and A. Ukkonen. Sparsification of influence networks. In *KDD*, 2011.
[19] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell. *Machine learning: An artificial intelligence approach*. Springer Science & Business Media, 2013.
[20] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 2015.
[21] M. Niepert, M. Ahmed, and K. Kutzkov. Learning convolutional neural networks for graphs. In *ICML*, 2016.
[22] A. Nowak, S. Villar, A. S. Bandeira, and J. Bruna. A note on learning algorithms for quadratic assignment with graph neural networks. *arXiv preprint arXiv:1706.07450*, 2017.
[23] M. Purohit, B. A. Prakash, C. Kang, Y. Zhang, and V. Subrahmanian. Fast influence-based coarsening for large networks. In *KDD*, 2014.
[24] L. Shi, H. Tong, J. Tang, and C. Lin. Vegas: Visual influence graph summarization on citation networks. *TKDE*, 2015.
[25] R. S. Sutton and A. G. Barto. *Introduction to reinforcement learning*. MIT press Cambridge, 1998.
[26] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 2007.
[27] P. Yanardag and S. Vishwanathan. Deep graph kernels. In *KDD*, 2015.
[28] H. Zhang, D. D. Yao, and N. Ramakrishnan. Detection of stealthy malware activities with traffic causality and scalable triggering relation discovery. In *Proceedings of the 9th ACM symposium on Information, computer and communications security*.