



Flattening the Curve Through Reinforcement Learning Driven Test and Trace Policies

Andrei C. Rusu^(✉), Katayoun Farrahi, and Mahesan Niranjan

University of Southampton, Southampton, UK
{a.rusu,k.farrahi,mn}@soton.ac.uk

Abstract. An effective way of limiting the diffusion of viruses when vaccines are unavailable or insufficiently potent to eradicate them is through running widespread “test and trace” programmes. Although these have been instrumental during the COVID-19 pandemic, they also lead to significant increases in public spending and societal disruptions caused by the numerous isolation requirements. What is more, after the health measures were relaxed across the world, these programmes were unable to prevent substantial upsurges in infections. Here we propose an alternative approach to conducting pathogen testing and contact tracing that is adaptable to the budgeting requirements and risk tolerances of regional policy makers, while still breaking the high risk transmission chains. To that end, we propose several agents that rank individuals based on the role they possess in their interaction network and the epidemic state over which this diffuses, showing that testing or isolating just the top ranked can achieve adequate levels of containment without incurring the costs associated with standard strategies. Additionally, we extensively compare all the policies we derive, and show that a reinforcement learning actor based on graph neural networks outcompetes the more competitive heuristics by up to 15% in the containment rate, while far surpassing the standard random samplers by margins of 50% or more. Finally, we clearly demonstrate the versatility of the learned policies by appraising the decisions taken by the deep learning agent in different contexts using a diverse set of prediction explanation and state visualization techniques.

Keywords: epidemic-control · target-test-and-trace · reinforcement-learning

1 Introduction

The recent pandemic caused by the SARS-CoV-2 virus has fundamentally shaped the way we plan for and respond to the spread of highly-infectious pathogens. Drastic control measures like imposing general lockdowns proved to be particularly damaging to the global economy and the wellbeing of the population [42], causing widespread discontent among all social strata.¹ As such, less restrictive

¹ COVID-19 Attitudes Survey by YouGov: <https://tinyurl.com/yougov-attitudes>.

health interventions were introduced in lieu to curb dangerous infection rates, such as educating the public to socially distance, deploying large-scale testing schemes and quarantining contacts through different tracing mechanisms [21]. Despite the advent of highly effective vaccines [3,12], financing and support for these measures continued for several months in the majority of the Western world, fueled by evidence of their continued efficacy [63,83]. However, with the emergence of seemingly milder variants [91], concerns about the limitations [67] or the societal impact [15,60] of the aforementioned interventions, and growing evidence of reduced public compliance [18], several administrations decided to significantly reduce the resources allocated for these programmes. In the United Kingdom, for example, the new “living with COVID” strategy meant appreciable cost reductions could be achieved [61], while heavy disruptions like the recent “pingdemic” could be entirely avoided [76,81]. Unfortunately, blindly scaling down the public health efforts to break transmission chains has proven unsuccessful as cases across the country soared yet again within a relatively short timeframe,² trend that has been replicated across Europe [34]. With the vaccine protection waning over time [24,55], and with demand for further doses decreasing among healthy adults [94,97], similar surges could reoccur henceforth.

In this work, we propose a major shift in the implementation of “test and trace” programmes that is adaptable to a country’s budget and risk tolerance, while minimizing the burden of viral infection chains. To achieve this, we study different types of targeted policies for conducting testing and isolating contacts in an epidemic under fixed budgeting requirements, and show that a reinforcement learning agent can derive powerful and generalizable policies that outperform all baselines considered in terms of infection reach. We validate our results on several epidemic, budget and interaction network configurations, illustrating the versatility of our proposed method. Moreover, we demonstrate that even static non-learning agents significantly outcompete customary untargeted strategies.

The contributions in this paper are threefold:

1. We put forward a novel way of operating public health interventions in a realistic scenario where economic and societal disruptions are to be minimized: restricting the testing and tracing efforts to higher-risk individuals. To that end, we derive highly-effective policies using different agents, including centrality-based, neighborhood-based and learning-based, comparing them against more traditional approaches, such as random, acquaintance or frequency-based sampling. Our reinforcement learning agent, backed by a Graph Neural Network (GNN) adapted from the recent development of [65], is shown to outperform the other methods in both tasks across numerous configurations, despite being trained using a simple test prioritization setup with partial observable information.
2. Aside from presenting the numerical results and epidemic curves resulted from running our control policies over multiple simulations, we also study what the learning-based agent chooses to focus on while making its decisions. For such a system to be deployed in the real world, policy makers

² COVID-19 Infection Survey by ONS: <https://tinyurl.com/ons-covid19>.

need to be reasonably confident the model produces sensible outputs. At the same time, testing or isolation decisions have to be explainable and verifiable when audited or contested. Here, we explore a perturbation-based technique for explaining the policy derived by an agent’s GNN module, GraphLIME [39], putting into perspective the former’s superior adaptability. Moreover, we propose visualizations for the inferred node embeddings that can be used to direct community-wide interventions or scrutinize the model’s performance.

3. We apply our framework to several scenarios featuring COVID-specific spreading models, including a multi-site mean-field [40,83] and an agent-based model, with parameters obtained from [20], and show that our agents consistently perform well across a diverse set of experimental setups.

2 Related Work

2.1 Epidemic Modelling

Traditionally, simulating epidemics has been accomplished using either equation-based or agent-based models. The first of these is possibly the most common, owing its appreciable success to early work by [45], where the modelled population was said to transition between disease-specific compartments according to a system of ordinary differential equations. Recent years, however, have seen agent-based approaches become more popular, partly due to their superior granularity and ability to assess a system’s behavior at the individual level [98]. Government-advising groups in the United Kingdom employed this paradigm during the initial waves of the COVID-19 pandemic to assess the effects of public health interventions [25,35]. Others used such formulations to study the combined effects of manual tracing with digital solutions at various application uptakes, employing parameters fitted to infection data from several regions [1,83]. In this study, we simulate viral epidemics using a modified version of a recently-proposed multi-site mean-field model [83], which relies on the SEIR compartmental formulation but retains the capacity to leverage an individual’s locality information through contact graphs and mean effects [23,40]. For completeness, we also investigate our policies in a purely agent-based setup, similar in spirit to the network-based approaches proposed in recent works [1,65]. In both cases, we employ the COVID-specific dynamics parameters inferred by [20], and allow all disease-unrelated events to be time-discretized (i.e. selecting an action or updating the active links set takes place every t_u days, with $t_u=1$).

2.2 Graph Neural Networks and Reinforcement Learning

A few years back, graph neural networks became one of the de facto machine learning tools for processing graph-structured information [27,110]. The earliest studies in this space defined a GNN as a set of two functions: transition f_θ and output o_θ [30,86]. The former expresses the dependence between a node i and its vicinity, while the latter controls the space spanned by the model output.

These functions form the basis of what later came to be known as the message-passing paradigm [29], which quickly became dominant in the field due to its effectiveness and computational efficiency [11]. After graph convolutional networks (GCN) were first introduced [48], many GNN successes followed course [11,32]. Motivated by the accomplishments of attention mechanisms in natural language processing [19,104] and computer vision [4,46], authors soon enhanced GNNs with attention capabilities, often increasing their performance (e.g. GAT [105], GATv2 [10]). Expressive power bounds for the message passing algorithm were later noted by [108], who then proposed an architecture that reaches the upper limit of the widely-used Weisfeiler-Lehman heuristic (1-WL): the Graph Isomorphism Network (GIN). Efforts to break node symmetries and surpass this upper bound have been significant ever since, with many approaches currently existing: augmenting the nodes with random features [85], modifying the message passing rule [6], or changing the input graph structure itself [71]. Additionally, issues such as feature *oversmoothing* [36,74] and *bottlenecks* [2] have been identified as common reasons for underperforming message passing systems, with proposed solutions ranging from maintaining a low layer count and connecting all nodes in the last layer to ease information flow, to augmenting the message exchange routine (e.g. Neural Sheaf Diffusion [8]). Our framework leverages ideas from GATv2 and GIN to attain expressive power and computational efficiency, while reducing the impact of the above problems by using randomised node features and a small number of GNN layers, with a final fully-adjacent layer to mitigate the over-squashing of long-range dependencies.

A widely-used approach for explaining predictions in deep learning involves perturbing the inputs and fitting local explainable models to each data point and its corresponding perturbations. LIME [80] and SHAP [58] are two popular examples of this methodology. Although the above are directly applicable to GNNs, they do not possess the capability to leverage structural information from graph data or capture nonlinear relationships between the inputs and the outputs. To solve these limitations, GraphLIME was proposed [39]. GraphLIME replaces the local perturbations matrix with stacked node features selected from a node’s neighborhood, fitting nonlinear interpretable models using HSIC Lasso [109].

Among many other domains, GNNs have also been extensively used in the context of epidemiology. From the literature dedicated to COVID-19, we note here several noteworthy efforts: infection forecasting [44,75], full population state estimation [102], finding “patient 0” [90], and controlling public interventions, such as testing [65] or vaccination policies [41].

Sequential decision processes are often modelled via Markov Decision Processes (MDPs) of the form $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$ [78], where \mathcal{S} is a state space, \mathcal{A} is an action space, \mathcal{P} is a transition probability matrix, while \mathcal{R} is a reward function for the state-action pairs. Agents sample actions from their policy $a_t \sim \pi(a|s_t; \theta)$, with $a \in \mathcal{A}$ and θ a parametrization, then execute them, transitioning to different states s_{t+1} and earning rewards R_t , according to the environment’s \mathcal{P} and \mathcal{R} . The goal of reinforcement learning (RL) is to solve MDPs by predicting and maximizing the γ -discounted returns of future rewards $G_t = \sum_{i=1}^T \gamma^{i-1} R_{t+i}$ [100].

This is routinely achieved through supervision using a w -parameterized model $V(s_t|w)$ that predicts G_t , and the returns or some intermediary estimates as targets. The first of these approaches is called the Monte Carlo (MC) algorithm, and is known to be effective despite presenting several drawbacks: slow offline learning and high variance [99,100]. For example, a variation of MC featuring search trees was used to derive competitive policies in 2-player board games [92,93]. In contrast, the online temporal difference (TD) learning method casts the sum of the current estimate of the next-step return $G_{t+1} = V(s_{t+1}|w)$ and R_{t+1} as a regression target, lowering the variance and speeding up training [103]. The latter constitutes the basis for many RL algorithms to date, such as the on-policy SARSA [82,100] and the off-policy Q-learning [106], which proved successful in multiple problem instances: reaching or outperforming human-level performance in games [69,70], autonomous car driving [49], and many others.

Approaches that directly optimize both θ and w are called actor-critics [52], and have become the preferred algorithmic choice when faster convergence rates are sought after and sample efficiency is not required. Recent years have seen actor-critic methods like the Proximal Policy Optimization (PPO) [88] and Deep Deterministic Policy Gradient (DDPG) [56] achieve state-of-the-art results across a wide range of challenging tasks [54,88]. Although online implementations are possible, these agents have traditionally been trained using MC.

Learning policies in environments with combinatorial action spaces such as ours has generally been considered a difficult undertaking. In spite of this, RL methods proved to be effective in instances like multiple item [96] or thread popularity selection [33]. In the context of epidemics, an RL system based on multi-armed bandits and demographics data was recently introduced by the Greek authorities to prioritize the COVID-19 testing allocations at border control [5]. For classic combinatorial problems, such as the travel salesman (TSP) and its vehicle routing variants, RL approaches have also been shown to perform well [7,53]. Incorporating graph embeddings into the RL agents have generally lead to improved solvers, outcompeting other learning methods [17,43].

2.3 Influencing Graph Dynamics

The problem of influencing diffusion processes over networks has been studied in many different settings before, most notably for solving influence maximization [73], optimizing immunization strategies [77], and targeting pathogen testing [66]. It has been long established that random vaccination policies tend to be sub-optimal, and even simple heuristics like acquaintance sampling can outperform them [16,68]. Centrality-based strategies were also explored in this context, with PageRank [14], eigenvector [62] or betweenness centrality [84] becoming popular choices. For influence maximization, degree-based strategies were shown to render competitive results (e.g. LIR [57], degree discount [13]). Over time, however, multiple authors have identified problem instances where any centrality measure used by itself can lead to suboptimal results [9,77]. The question of which heuristic to use for what problem has since become a focal point in many application domains. As an alternative, reinforcement learning techniques have been proposed

for mixing different heuristics in an optimal manner, thus reducing the impact of the aforementioned drawbacks [65,101]. Node targeting for detecting the state of a spreading process is a slightly less explored use case of control in the literature, but efficient heuristics that exploit the known state of a vertex’s neighborhood instead of centrality-derived information have proven to be successful [66]. The domain of prioritizing contact tracing, however, remains largely uninvestigated to date, but recent work suggests that isolating subsets of individuals based on the frequency of appearing in the vicinity of positive cases can lead to similar levels of containment as naively isolating every contact [51].

Meirom et al. introduce a reinforcement learning model that can derive general control policies for diffusion processes over networks, using test prioritization and influence maximization as illustrations [65]. A GNN-based controller, cast in an actor-critic framework, learns effective policies using simulated data, integrating local and long-distance information over time. The elegance of the approach stems from the fact that the training process is not conditioned on having the full epidemic state made available to the agent. The work also shows that it is possible to learn a policy on small networks (e.g. 1000) and deploy it on larger graphs with similar statistics (e.g. 50000, the size of a small city). Our study builds on top of this versatile control framework, but differs from the aforementioned work in several key aspects: First, we extend the problem formulation to cover prioritizing both testing and tracing, amending the framework to accommodate ranking nodes from eligible subsets. The latter also enables us to add a simple extension to all our agents which empirically improves performance: restricting the action space to exclude recently-tested negative individuals. Second, we analyze the control outcomes more thoroughly, looking at longer evaluation episodes than 25 days, plotting epidemic curves, and interpreting the agents’ decisions using a perturbation-based explainability technique designed for graphs, GraphLIME. Third, we employ COVID-specific spreading parameters and analyze the behavior of the policies beyond agent-based modelling. Finally, we perform a range of algorithmic changes in our implementation to improve efficiency: using bootstrapping and eligibility traces to mitigate the memory cost of the offline PPO routine, a shared network between the actor and the critic [88] to enrich the graph embeddings, a GATv2 layer in the diffusion module to enable a better tracking of the point-to-point spreading process, multiple GIN layers followed by a final fully-adjacent one in the information module to increase its expressive power, as discussed above, and standard scaling for bounding the exploding node hidden states instead of L^2 normalization or GRU-based transformations.

3 Methodology

3.1 Simulating Epidemics

We simulate several epidemics using the SEIR compartmental model together with COVID-specific parameters obtained from [20]: a base infection rate of $b = 0.0791$ and an average exposed duration of $e = 3.7$ days. In order to remove

stochastic artefacts that may conceal performance differences, most of our setups assume that nodes remain infectious for the whole duration of the episode unless they get isolated (i.e. recovery rate $\rho = 0$). Intuitively, the impact of this assumption becomes significant only when the problem becomes *oversaturated* (i.e. the testing budget k and/or the recovery rate ρ are large enough for any agent to achieve containment). For completeness, however, we also present results when ρ is varied (see Section 4.1).

The viral infection diffuses over different interaction network configurations, with events getting generated by either a multi-site mean-field (see [83]) or an agent-based model. These configurations correspond to both common artificial generation methods, such as Erdős–Rényi [22], dual Barabási-Albert [72] or Holme–Kim [38], and real interaction patterns. In practice, the graph connections needed for conducting such fine-grained control would have to be inferred from a monitoring system, like a digital tracing mechanism [65] or human mobility tracking via GPS [89], process which requires careful data anonymization. We assign a transmission weight $w_j \sim \mathcal{U}(0.5, 1)$ to every edge j in our graphs, calculating an interaction’s transmission probability by scaling w_j with the base factor b . In the multi-site mean-field simulations, stochasticity is ensured by the events sampling procedure, which is efficiently performed using Gillespie’s algorithm [28]. In contrast, the agent-based model relies on sampled exposed-state and recovery durations for each node, $d_i \sim \mathcal{N}(e, 1)$, $r_i \sim \mathcal{N}(\frac{1}{\rho}, 1)$, and the w_j weights to induce variability among individuals. For further details, please consult Appendix A.2.

3.2 Control Setup

Each epidemic is allowed to progress until at least c_a days have passed since the simulation began and a minimum of c_i nodes become infected before the agent commences its interventions. In the first day of control, the agent is informed at random about the status of a proportion c_k of the *infected* population, after which it is only allowed to test k individuals and isolate k_c contacts of recently-detected positive nodes (i.e. in the previous 6 timesteps) per day. As the actor is not aware of a node’s state unless it is a part of c_k or it got tested recently, the environment is partially observable. In this work, we fix $c_a = 5$, $c_i = 5\%$ and $c_k = 25\%$, while the budgets are varied between experiments. A block diagram of our framework, which includes the agents’ class hierarchy, is provided in Fig A1.

During evaluation, each agent is asked to select the top- k nodes to test and the top- k_c contacts to isolate every day, according to their appraisal of the epidemic and graph states. Consequently, this constitutes an instance of the subset selection problem [79], where nodes that are traced by the system or found to be positive are marked as isolating, becoming incapable of infecting other nodes. In principle, those individuals remain disconnected from the graph, yet we allow messages to continue flowing through their connections during the training phase of the learning-based agents. Importantly, the process of tracing is assumed to be carried with delays shorter than a day, which usually implies that a contact tracing application is already deployed and functioning [26,107]. To evaluate the

efficacy of each policy, we analyze the fraction of nodes kept healthy through the entire epidemics and the corresponding infection curves.

3.3 Baseline and Learning Agents

In this study, we consider a wide variety of baseline agents for controlling the viral diffusion that leverage separate heuristics: Random samplers (or *randag*); Acquaintance samplers (or *acq*); Centrality-based (e.g. Degree or *deg*, Eigenvector or *eig*, PageRank or *prank*, Closeness, Betweenness); Neighborhood state-based (or *neigh*). The latter is the only baseline that uses information about the epidemic state, targeting the nodes that have the highest number of positively-detected neighbors in their 2-hops vicinity via lexicographical ordering [66].

Aside from the above, when ranking the contacts of positives, additional information can be exploited through heuristic methods: the frequency with which nodes appear in the neighborhood of detected cases. We derive two baselines from the above: Frequency, which randomly samples nodes with probabilities proportional to the individual frequencies (equivalent to the tracing mechanism studied in the multi-site mean-field approach of [83]), and Backward, which greedily picks the nodes with the highest frequencies (as per [51]).

We also propose a simple yet powerful extension to these baselines: recollection of recent negative test results. This effectively restricts the action space to untested nodes in the past t_n days, speeding up the network exploration. We set $t_n = 3$, an appropriate timeline for COVID-19 [95], which renders good results empirically.

Our learning-based agents are inspired by the recent publication of [65], leveraging multiple GNNs due to their proven efficacy for targeting testing campaigns. The abstract structure of our models remains similar to the previous work, with a single-layered diffusion module and a long-range information module, followed by two multi-layer perceptrons (MLPs), one that computes the node hidden states h_i , and another that defines the output space. However, our proposed solution features several improvements or simplifications: First, we utilize two output MLPs to produce a score for each vertex and a full state score from the same model, thus sharing the embedding space between the two. Second, we employ a GATv2 layer in the diffusion module to leverage attention when aggregating information from the immediate neighborhood of each node, and 3 GIN layers followed by a fully-adjacent layer in the information module to improve the expressivity and long-range information flow. Finally, after experimenting with different normalization schemes to mitigate the issue of the exploding hidden states h_i (problem also outlined in the aforementioned study), we propose the usage of standard scaling, which leads to stable training behaviors.

In addition to the above, we carefully scrutinised different combinations of node features, choosing the following final set for training our policies: the degree and eigenvector centralities, the number of infected vertices in the 1-hop and 2-hop neighborhoods, 5 random features that break structure symmetries, and 4 test-state features: a one-hot vector of size 3, marking the test status of node i at the previous timestamp (untested, negative or positive), and a binary value

marking whether the vertex has ever tested positive. To allow for the hidden states to incorporate information from these features before the training commences, we disable gradient updates for the first 11 passes.

The ranking of nodes can be performed by both a supervised learning (SL) and a reinforcement learning (RL) agent, with little to no changes to the underlying neural network architecture. The SL agent is trained as a simple node classifier by optimizing a binary cross-entropy loss on the infection status of each vertex, with the output space representing the next-step infection likelihood. In contrast, our RL agent gets optimized via a surrogate PPO objective, which only needs access to the total number of infected at each time point (for more details, refer to Appendix A.3), ultimately solving for the criterion below, where $E(t)$, $I(t)$ and $R(t)$ are the number of individuals in each compartment at time t :

$$\min \sum_{t=t_0}^{\infty} \gamma^{t-t_0} (E(t) + I(t) + R(t)) \quad (1)$$

Here, two reward functions can be used: negative of the number of infected or the number of susceptible vertices at time t (corresponding models denoted as *rl* and *rlpos*, respectively). The performance between the two varies due to numerical reasons, but the differences are small (see Fig A5). Consequently, Section 4 features only the former in the summary tables.

To ensure sufficient exploration during training, the RL agent passes the raw outputs of the ranking model through a softmax function that features a decaying temperature, starting from $\epsilon = 0.5$. Note that other strategies are also possible here, including the transforms proposed in [64] and [65], but our simple alternative proved sufficiently effective at exploring the state space. During evaluation, the sampling process is turned off, greedy actions are taken instead, and the edges connected to positively-identified vertices are masked before being fed to the information module, limiting feature *oversmoothing*. In contrast, we allow the single-layer diffusion GNN to utilize to the aforementioned links such that the positive-related node features can pass through to their neighbors.

By comparing the training behavior of the SL and RL agents with the containment achieved by the centrality-based actors with recollection, we observe a clear distinction between the two, as reflected by Fig 1. While the RL policy outperforms all baselines in several episodes, despite not entering evaluation mode as of yet (i.e. when exploration would be turned off), the SL policy struggles to compete. Further evidence of the SL agent’s underperformance can be seen in the plots of Fig A2, as well as in the extensive comparison previously conducted by [65]. Consequently, we focus our main analysis in Section 4 on the policies derived by the RL actors, comparing them against the rest of the baseline agents.

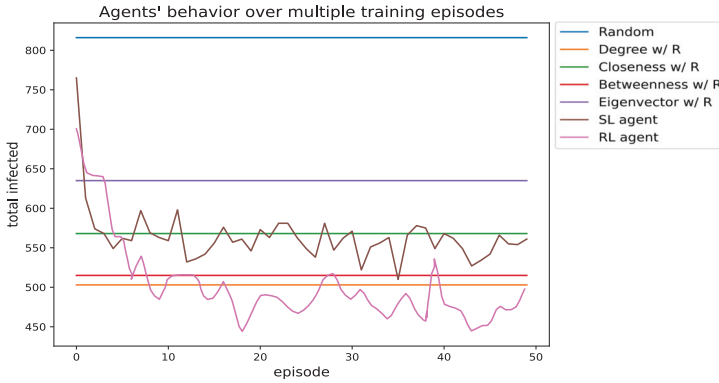


Fig. 1. The learning agents’ training behavior. Results obtained by the centrality-based agents and the random tester are plotted for comparison.

4 Results and Discussion

4.1 Prioritizing COVID Testing in Static Graphs

We first explore our agents’ policies in the context of targeted testing campaigns. To that end, we investigate the fraction of nodes kept healthy throughout various epidemics triggered across different network models when the budget of daily testing k is fixed, while k_c is set to 0. Once tested positive by the framework, a node gets isolated and eventually acquires immunity, thus remaining uninfected until the end of the simulation. As stated previously, most of our setups assume nodes do not spontaneously become uninfected (i.e. $\rho = 0$), but for completeness we present results for different full-recovery rates in Table 1.

Despite being trained for only 50 episodes on a single epidemic configuration spanning a preferential attachment network of 1000 nodes, our reinforcement learning agent consistently outperforms the other baselines across a range of different network sizes (see Table 2), budgets (see Fig A4), and wiring configurations (see Fig A5). Interestingly, as previously hinted by [65], the learning-based agents poses a great generalization capability when the daily budgets scale with

Table 1. Fraction kept healthy with budget $k = 1\%$ and different recovery rates. Average over 5 seeded runs for each of the considered 5 realizations of Barabási-Albert networks with $N = 1000$ nodes and a mean degree of approximately 3. “w/R” denotes agents with recollection of recent negative test results.

Agents	$\rho = 0$	$\rho = 0.01$	$\rho = 0.02$	$\rho = 0.03$
Degree	0.555 \pm 0.027	0.616 \pm 0.034	0.662 \pm 0.039	0.697 \pm 0.039
Degree (w/R)	0.744 \pm 0.032	0.769 \pm 0.028	0.801 \pm 0.028	0.847 \pm 0.025
PageRank (w/R)	0.720 \pm 0.026	0.755 \pm 0.023	0.792 \pm 0.037	0.834 \pm 0.039
RL	0.822 \pm 0.033	0.846 \pm 0.026	0.876 \pm 0.026	0.897 \pm 0.026

Table 2. Fraction kept healthy with budget $k = 1\%$ and different population sizes. Average over 5 seeded runs for each of the considered 5 realizations of Barabási-Albert networks with a mean degree of approximately 3. “w/R” denotes agents with recollection of recent negative test results. Here, a single model is trained for 50 episodes on a network of size 1000, but its policy is able to generalize to appreciably larger graphs.

Agents	N = 500	N = 1000	N = 2000	N = 5000	N = 20000
Degree	0.533 ± 0.037	0.555 ± 0.027	0.552 ± 0.017	0.567 ± 0.027	0.557 ± 0.005
Degree (w/R)	0.731 ± 0.031	0.744 ± 0.032	0.736 ± 0.027	0.737 ± 0.025	0.746 ± 0.009
PageRank (w/R)	0.709 ± 0.019	0.720 ± 0.026	0.724 ± 0.021	0.729 ± 0.021	0.725 ± 0.008
RL	0.817 ± 0.032	0.822 ± 0.033	0.811 ± 0.024	0.821 ± 0.025	0.803 ± 0.026

Table 3. Fraction kept healthy for 1000 nodes. Results are averaged over 5 runs for each of the 5 realizations of a configuration model built using real tracing statistics.

Agents	$k = 20$	$k = 50$
Acquaintance (w/R)	0.465 ± 0.086	0.736 ± 0.085
Degree (w/R)	0.406 ± 0.020	0.746 ± 0.025
Eigenvector (w/R)	0.186 ± 0.013	0.409 ± 0.026
PageRank (w/R)	0.363 ± 0.016	0.668 ± 0.039
RL	0.506 ± 0.029	0.831 ± 0.047

the number of nodes, making possible a deployment into larger networks, irrespective of the training graph size and artificially without losing efficacy.

Several epidemic curves corresponding to prioritizing testing in 5000 nodes graphs are shown in Fig A3. We note the random approaches perform strikingly poorer than all our informed policies, while the impact of recollection is apparent. Moreover, in spite of using recollection, the heuristics considered remained inferior to the RL policy in terms of the average containment rate.

4.2 Prioritizing Testing in Dynamic Graphs

In the previous section, we analyzed scenarios in which the connections between nodes remain fixed for the entire simulation. However, in practice, the interaction patterns change over time. In Fig 2, we present boxplots of the percentage of nodes kept healthy obtained by different agents on several preferential attachment networks whose active edges are sampled every day (a uniform random fraction is sampled daily from $\mathcal{U}[0.4, 0.8]$). The reinforcement learning agent was retrained to accommodate this dynamic context, allowing the model to pass messages through the most recent edges only. The top performing policies were also evaluated on dynamic networks built using statistics from a real contact tracing network [65], the resulting average containments being displayed in Table 3.

4.3 Targeted Test and Trace Programmes

Next, we investigate the extent to which different combinations of agents tasked with conducting testing and contact tracing under the constraints of a fixed

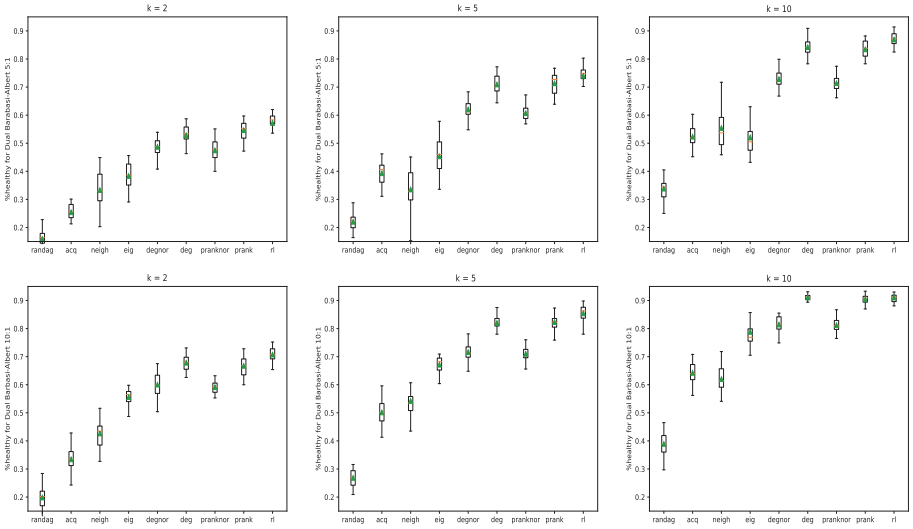


Fig. 2. Infection control performance on different dynamic network architectures. The uncertainties are shown as boxplots.

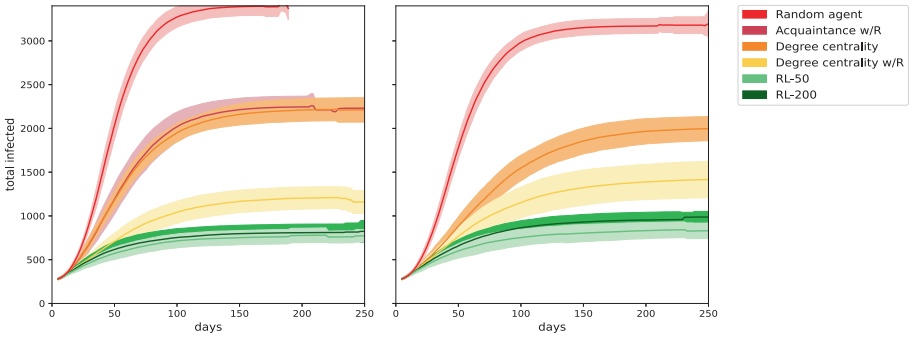


Fig. 3. Averaged epidemic curves and their standard deviations during test and trace control. These are for 5000 nodes Barabási-Albert networks featuring a mean degree of approximately 3, with a daily testing budget of $k = 1\%$ and no tracing on the left, and $k = 10$ with a limit of $k_c = 25$ traced contacts on the right. Two RL agents are displayed: one trained for 50, and the another for 200 episodes.

budget can reduce the spread of a pathogen. For this problem, we train an RL agent for 200 episodes on the same testing task as before, and compare the resulting policy against the other baselines. Tables 4 and 5 confirm the RL tester improves the overall quality of the test and trace programmes, irrespective of the chosen tracer. That being said, employing the same agent to perform the ranking of contacts as well generally improves the containment.

We also inspect the averaged epidemic curves associated with these targeted test and trace campaigns when $N = 5000$. The results obtained by each agent

Table 4. Percentage of nodes kept healthy for graphs of size 1000 and an approximate mean degree of 3, with budgets $k = 2$, $k_c = 5$. Averages over 5 runs for each of the considered 5 realizations of the following: dual Barabási-Albert with $m_1 = 5$, $m_2 = 1$ (BA 5-1) and $m_1 = 10$, $m_2 = 1$ (BA 10-1), Holme-Kim (PC), and Erdős-Rényi (ER).

Agents (Test + Trace)	BA 5-1	BA 10-1	PC	ER
Random + Random	0.389 ± 0.044	0.446 ± 0.060	0.131 ± 0.023	0.199 ± 0.023
Random + Frequency	0.387 ± 0.033	0.465 ± 0.059	0.202 ± 0.030	0.195 ± 0.022
Acquaintance (w/R) + Random	0.541 ± 0.054	0.657 ± 0.054	0.212 ± 0.031	0.215 ± 0.017
Acquaintance (w/R) + Frequency	0.582 ± 0.055	0.674 ± 0.059	0.228 ± 0.040	0.217 ± 0.021
Acquaintance (w/R) + Backward	0.591 ± 0.056	0.769 ± 0.080	0.213 ± 0.039	0.208 ± 0.019
Acquaintance (w/R) + RL	0.644 ± 0.048	0.806 ± 0.058	0.248 ± 0.038	0.217 ± 0.018
Degree (w/R) + Degree	0.764 ± 0.038	0.915 ± 0.032	0.528 ± 0.053	0.333 ± 0.037
RL + Random	0.818 ± 0.034	0.882 ± 0.026	0.542 ± 0.050	0.438 ± 0.043
RL + Frequency	0.832 ± 0.035	0.890 ± 0.033	0.567 ± 0.054	0.448 ± 0.048
RL + Backward	0.849 ± 0.033	0.923 ± 0.023	0.590 ± 0.058	0.434 ± 0.047
RL + Degree	0.853 ± 0.034	0.928 ± 0.014	0.614 ± 0.055	0.453 ± 0.039
RL + RL	0.876 ± 0.025	0.936 ± 0.009	0.620 ± 0.050	0.451 ± 0.039

Table 5. Percentage of nodes kept healthy when controlling epidemics over a dynamic real interaction network of 74 vertices, derived from the Social Evolution dataset [59]. Averages over 5 runs for each of the considered 5 infection seeds. Test budget is $k = 2$.

Agents (Test + Trace)	$k_c = 2$	$k_c = 4$
Random + Frequency	0.511 ± 0.130	0.659 ± 0.114
Acquaintance (w/R) + Frequency	0.494 ± 0.113	0.649 ± 0.089
Acquaintance (w/R) + Backward	0.522 ± 0.115	0.654 ± 0.126
Neighborhood (w/R)	0.620 ± 0.108	0.704 ± 0.107
Degree	0.614 ± 0.107	0.741 ± 0.084
Degree (w/R)	0.636 ± 0.104	0.750 ± 0.084
RL	0.711 ± 0.089	0.773 ± 0.069

is shown on the second column of Fig 3, with the first serving as a test-only reference (i.e. values from Fig A3). As stated before, heuristics with recollection bring large improvements over random policies, yet the RL agents outcompete them in most setups. Note the performance of $k = 50$ tests is similar to $k = 10$ tests, but tracing up to $k_c = 25$ contacts daily. While the balance between these will depend on various factors, the results highlight the effectiveness of tracing.

4.4 Agents Interacting with Different Spreading Dynamics

To assess the ability of the agents to generalize to other spreading dynamics, we compare their achieved containment rates recorded with both a multi-site mean-

Table 6. Fraction kept healthy for 2000 nodes and an average degree of 3. Results represent averages over 5 runs for each of the considered 5 instances of a dual Barabási-Albert model ($m_1 = 10$, $m_2 = 1$). Testing budget is $k = 2$ and no tracing is conducted.

Agents	Multi-site	Agent-based
Random	0.164 ± 0.037	0.195 ± 0.034
Acquaintance (w/R)	0.251 ± 0.033	0.263 ± 0.035
Degree	0.390 ± 0.032	0.394 ± 0.029
Degree (w/R)	0.443 ± 0.032	0.457 ± 0.034
RL	0.468 ± 0.035	0.477 ± 0.034

Table 7. Fraction kept healthy for 2000 nodes and an average degree of 3. Results represent averages over 5 runs for each of the considered 5 instances of a dual Barabási-Albert model ($m_1 = 10$, $m_2 = 1$). Budgets are $k = 2$ and $k_c = 10$.

Agents (Test + Trace)	Multi-site	Agent-based
Random + Random	0.372 ± 0.035	0.371 ± 0.042
Acquaintance (w/R) + Backward	0.633 ± 0.046	0.627 ± 0.053
Degree (w/R) + Degree	0.841 ± 0.034	0.809 ± 0.028
RL + Backward	0.867 ± 0.029	0.851 ± 0.030
RL + Degree	0.889 ± 0.026	0.856 ± 0.025
RL + RL	0.911 ± 0.020	0.882 ± 0.018

field and an agent-based model run with similar hyperparameters. The RL agent retains all the learned parameters inferred from the previous experiments.

Despite the fact that the control mechanism in the mean-field case relies on discretizing a continuous-time process, we observe minor differences between the two simulation approaches (Tables 6 and 7). This confirms that the agents continue to perform well irrespective of the underlying dynamics.

4.5 Explaining and Visually-Inspecting the Learning Agent’s Policy

To derive explanations for the decision taken by our reinforcement learning policy, we employ the GraphLIME algorithm, fitting multiple interpretable models to the raw action-values the model outputs. Fig 4 presents the feature importances derived by GraphLIME for a given day in the early stages of an epidemic, highlighting that the RL agent preferentially attends to the centrality features when it does not possess enough information about the diffusion state. As soon as the tester records positive individuals in the vicinity of a vertex, the rank of the latter increases. After neighborhoods become filled with known infections, the agent targets the affected sectors by focusing on the epidemic state features (see Fig 5). As previous results also suggest, the degree remains an effective predictor

for a node’s importance throughout the process. Interestingly, the *untested* flag is often correlated with the action scores, which may indicate the agent favors exploring unknown sectors or reinforcing testing in recently-targeted regions. To put into perspective the significance of adaptability, we show in Fig 6 an example where an RL tester starts by targeting the same node as a degree-directed policy, but then quickly changes its behavior to also test bridging vertices. The ability to plan ahead and adapt to potential threats leads in this case to a successful

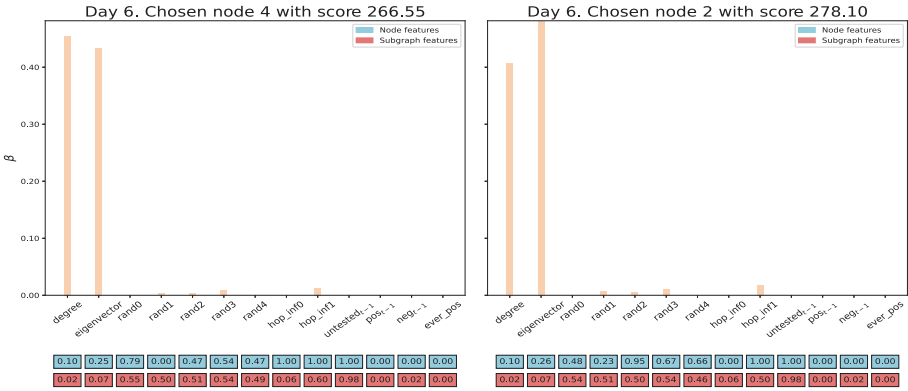


Fig. 4. Explaining early predictions on a 200 nodes network using the β importances from GraphLIME. Initially, the agent does not possess information about the epidemic state, and as such, it focuses on the centrality features. Top row displays each node’s feature values, while neighborhood averages are shown underneath.

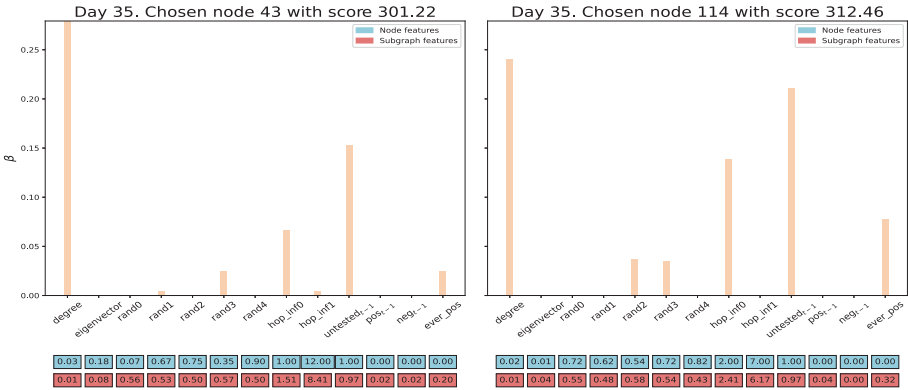


Fig. 5. Explaining later predictions on a 200 nodes network using the β importances from GraphLIME. During the later stages of an outbreak, the agent shifts its focus towards the epidemic state features, like the previously untested and positive flags, or the number of infected neighbors. Numbers in the the first row represent each node’s feature values, while the second row displays the neighborhood averages.

containment of the pathogen to the first cluster, while the degree agent is unable to stop the infection of every community. We note the RL infers that bridges are important transmission vehicles in spite of never computing the time-consuming betweenness centralities (also see Appendix A.1). Considering the promising results exhibited by our RL policy, we hypothesize that useful patterns emerge within the ranking model’s hidden states h_i . To verify our assumption, we plot t-SNE mappings and dendrograms for these embeddings across different days (refer to Fig A6). The detected positives (colored in blue) have a tendency to be grouped together, while new infections (red) get pushed to a handful of clusters within the same region. Such visualizations could be used for scrutinizing the actions of an agent or deriving effective community-wide health interventions.

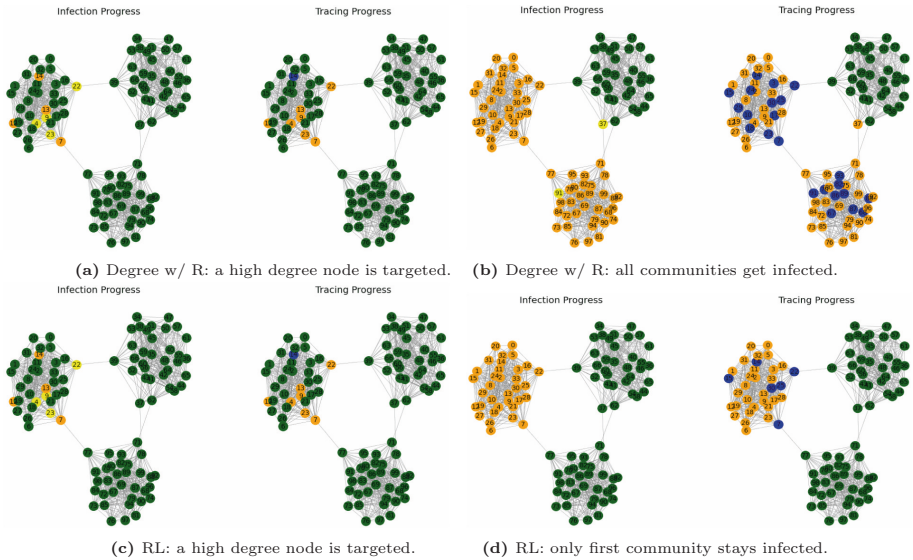


Fig. 6. Visualization of the spread for the Degree w/R and the RL agents. This corresponds to a stochastic-block network [37] with three communities. Susceptibles are green, exposed yellow, infectious orange, and detected blue. In the first day, the two policies are identical, but later on the RL agent preferentially targets the bridges. (Color figure online)

5 Conclusion and Future Work

In this study, we show how policies for controlling an epidemic through testing and tracing in a resource-limited environment can be learned using expressive graph neural networks that can integrate both local and long range infection dynamics. Across many different scenarios, a policy inferred by a reinforcement

learning agent outperforms a wide range of ad-hoc rules drawing from the connectivity properties of the underlying interaction graph, achieving containment rates of up to 15% higher than degree-based solutions with recollection, and more than 50% higher than random samplers. Interestingly, our agent also exhibits strong transferability, with one model trained on small preferential attachment networks being able to control the viral diffusion on several graphs of tens of thousands of vertices and diverse linkage patterns. While building on previous efforts [65], we explore the role of contact tracing, compare different ways of modelling the infection spread (multi-mean-field versus individual agent-based), and scrutinize a varied set of heuristics. Exploring further epidemic configurations and assessing the proposed test and trace framework on real region-level data would constitute natural extensions to this work.

Additionally, we demonstrate how orderings derived by the deep learning model can be interpreted using the node features, as well as propose visualization strategies for the cluster structures that arise in the latent space of the ranking module. We believe future work could expand on the aforementioned ideas to derive more effective public health interventions and decision-making appraisals.

Acknowledgements. We thank Dr Eli Meir for correspondence clarifying certain aspects from their study. MN was funded by EPSRC grant EP/S000356/1 Artificial and Augmented Intelligence for Automated Scientific Discovery. AR was funded by the EPSRC via a scholarship from the University of Southampton.

A Appendix

A.1 Performance Analysis

We compare the mean total elapsed time for running epidemics using each of our testing agents in Table A1. These results corresponds to the wall clock time recorded on an average Windows machine equipped with an Intel i7-7700 CPU, an NVIDIA RTX 3060 GPU and 32GB of random access memory.

Table A1. Average wall clock time per epidemic during evaluation. Configuration: Barabási-Albert networks of 2000 nodes, an average degree of approximately 3, and a daily testing budget of $k = 2$.

Agents	Wall time (s)	Agents	Wall time (s)
Random	1.12	Acquaintance (w/R)	1.12
Degree	3.23	Degree (w/R)	3.19
Closeness (w/R)	787.33	Betweenness (w/R)	1176.32
Eigenvector (w/R)	7.8	Pagerank (w/R)	6.39
Neighborhood (w/R)	1.49	RL/SL	15.92

A.2 Epidemic Modelling

All our epidemic models rely on the SEIR compartmental formulation, but the diffusion process remains bound by the interaction network configuration.

The multi-site mean-field models considered in this work rely on exponential waiting times sampled via Gillespie’s algorithm to obtain subsequent events, with the state transition probabilities defined as follows:

$$\begin{aligned}
 p(S \rightarrow E) &= b w_j \Delta t \\
 p(E \rightarrow I) &= e^{-1} \Delta t \\
 p(I \rightarrow R) &= \rho \Delta t,
 \end{aligned} \tag{2}$$

where b is the base transmission rate, w_j are time-dependent edge weights, e is the exposed state duration, ρ is the recovery rate, while Δt is a time interval.

In contrast, the agent-based model loops through every node i at every time step, executing the appropriate transition events when one of the normally-distributed samples (d_i or r_i) decreases to 0. Concurrently, every edge j is visited to check whether an infection event occurs over that connection, according to the transmission probability defined in Eq 2.

A.3 Algorithmic Details for the Proximal Policy Optimization

We start by reminding the reader about some general reinforcement learning quantities and relations:

$$\begin{aligned}
 \hat{A}_t^{(\gamma,0)}(a_t; \theta) &= \delta_t^\gamma(\theta) = R_t + \gamma V(s_{t+1}; \theta_T) - V(s_t; \theta) \\
 \hat{A}_t^{(\gamma,1)}(a_t; \theta) &= G_t^\gamma - V(s_t; \theta) \\
 \hat{A}_t^{(\gamma,\lambda)}(a_t; \theta) &= \sum_{l=0}^T (\gamma\lambda)^l \delta_{t+l}^\gamma(\theta) \\
 r_t^{ORIG}(\theta) &= \frac{\pi(a_t|s_t; \theta)}{\pi(a_t|s_t; \theta_k)} \quad r_t^{SARSA}(\theta) = \frac{\pi(a_t|s_{t+1}; \theta)}{\pi(a_t|s_t; \theta_k)}
 \end{aligned} \tag{3}$$

In Eq 3, R_t is the reward obtained by the agent after taking action $a_t \sim \pi(a|s_t; \theta_k)$ and transitioning from state s_t to s_{t+1} . The value of a given state s is approximated using a neural network $V(s, \theta)$, which, together with R_t and the discount factor γ , determines the TD-error δ^γ ; θ_k parameterizes the acting policy, θ_T is a delayed state of θ_k that parameterizes the regression target in online learning [69], $r_t^{ORIG}(\theta)$ denotes the ratio between a policy parameterized by a given θ and the acting policy, while $r_t^{SARSA}(\theta)$ represents an alternative formulation for the latter that replaces the numerator with the policy of θ evaluated at the next state s_{t+1} . Finally, the $\hat{A}_t^{(\gamma,\lambda)}$ terms represent different forms of the advantage function, as given by [87], with the special cases $\lambda = 0$, when the advantage is equal to the TD-error, and $\lambda = 1$, when the minuend of the RHS equation is the discounted return of the episode, G_t .

We rewrite the Proximal Policy Optimization (PPO) equations in terms of the quantities above in Eq 4, where \mathcal{E} , c_1 and c_2 are hyperparameters, $\text{clip}(\cdot)$ is a function that clips its argument to the specified range, $\text{transform}(\cdot)$ is a function that modifies the gradient descent update according to a specific optimizer (e.g. Adam [47]), while $\mathcal{H}_t(\theta)$ is an entropy regularizer [31]. In contrast to the original formulation, Eq 5 describes our proposed modification of PPO to allow for optimizing the objective in a memory-efficient online manner. In particular, we rewrite the loss terms using the one-step advantage function $\hat{A}_t^{(\gamma,0)}(a_t; \theta)$, and introduce an intermediary operation that accumulates the gradients of our modified loss using a unified eligibility trace [100], in a similar fashion to the methodology employed by [50], obtaining a backward-view approximation of the generalized advantage estimate $\hat{A}_t^{(\gamma,\lambda)}$ in the process [87]. We note that, by setting $r_t = r_t^{\text{SARSA}}$, we can eliminate the requirement of storing s_t in memory for the subsequent timestamp, while retaining the benefits of ratio clipping. This works well empirically since major shifts between s_{t+1} and s_t are not common in our environment. Based on previous work and our own assessment, we set $\gamma = 0.99$, $\lambda = 0.97$, $\mathcal{E} = 0.2$, $c_1 = 0.5$, $c_2 = 0.01$, and update the target value network every 5 episodes across all our experiments.

$$\begin{aligned}
\mathcal{L}_t^{\text{CLIP}}(\theta) &= \min[r_t(\theta)\hat{A}_t^{(\gamma,\lambda)}(a_t; \theta), \text{clip}(r_t(\theta), 1 - \mathcal{E}, 1 + \mathcal{E})\hat{A}_t^{(\gamma,\lambda)}(a_t; \theta)] \\
\mathcal{L}_t^{\text{VF}}(\theta) &= [\hat{A}_t^{(\gamma,1)}(a_t; \theta)]^2 \quad \mathcal{H}_t(\theta) = - \sum_{a \in A} \pi(a|s_t; \theta) \log \pi(a|s_t; \theta) \\
\mathcal{L}_t^{\text{PPO}}(\theta) &= \mathbb{E}_t[-\mathcal{L}_t^{\text{CLIP}}(\theta) + c_1\mathcal{L}_t^{\text{VF}}(\theta) - c_2\mathcal{H}_t(\theta)] \\
\theta_{k+1} &= \arg \min_{\theta} \mathcal{L}_t^{\text{PPO}}(\theta)
\end{aligned} \tag{4}$$

$$\begin{aligned}
\mathcal{L}_t^{\text{OCLIP}}(\theta) &= \min[r_t(\theta)\hat{A}_t^{(\gamma,0)}(a_t; \theta), \text{clip}(r_t(\theta), 1 - \mathcal{E}, 1 + \mathcal{E})\hat{A}_t^{(\gamma,0)}(a_t; \theta)] \\
\mathcal{L}_t^{\text{OVF}}(\theta) &= [\hat{A}_t^{(\gamma,0)}(a_t; \theta)]^2 \quad \mathcal{H}_t(\theta) = - \sum_{a \in A} \pi(a|s_t; \theta) \log \pi(a|s_t; \theta) \\
\mathcal{L}_t^{\text{OPPO}}(\theta) &= -\mathcal{L}_t^{\text{OCLIP}}(\theta) + c_1\mathcal{L}_t^{\text{OVF}}(\theta) - c_2\mathcal{H}_t(\theta) \\
E_t &= \gamma\lambda E_{t-1} + \frac{\nabla_{\theta_k} \mathcal{L}_t^{\text{OPPO}}(\theta_k)}{s}, \text{ with } s = \delta_t^\gamma(\theta_k) \text{ or } s = 1 \\
\Delta\theta_k &= \text{transform}(\delta_t^\gamma(\theta_k)E_t) \\
\theta_{k+1} &= \theta_k - \Delta\theta_k
\end{aligned} \tag{5}$$

A.4 Supporting Figures

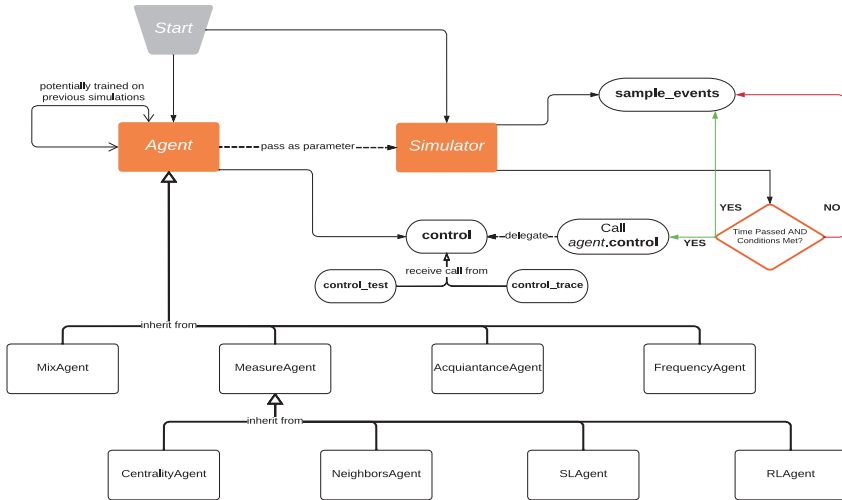


Fig. A1. Block diagram of our control framework. The *Agent* is passed as a parameter to the *Simulator*, and every time the latter samples enough events for the conditions to be met, a call to the *control(.)* method of the first is performed. The aforementioned function performs some preprocessing steps, and then calls *control_test(.)* and *control_trace(.)*, which are responsible for the actual node ranking and are specific to each type of agent. Combinations of agents can be selected with the *MixAgent*.

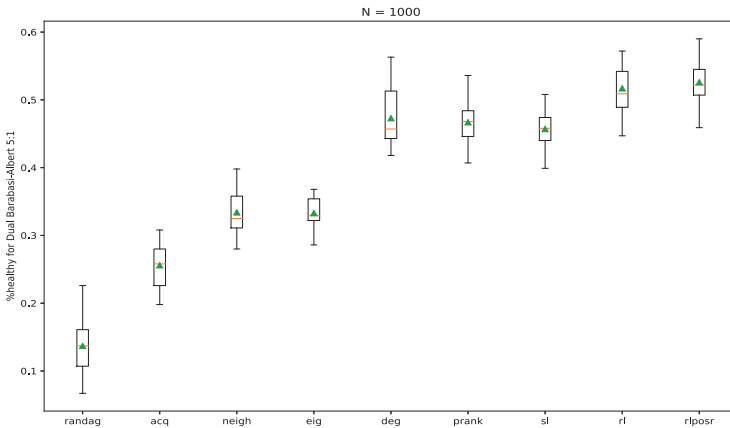


Fig. A2. Infection control performance on different network architectures of 1000 nodes and a daily testing budget of $k = 2$. Uncertainties shown as boxplots.

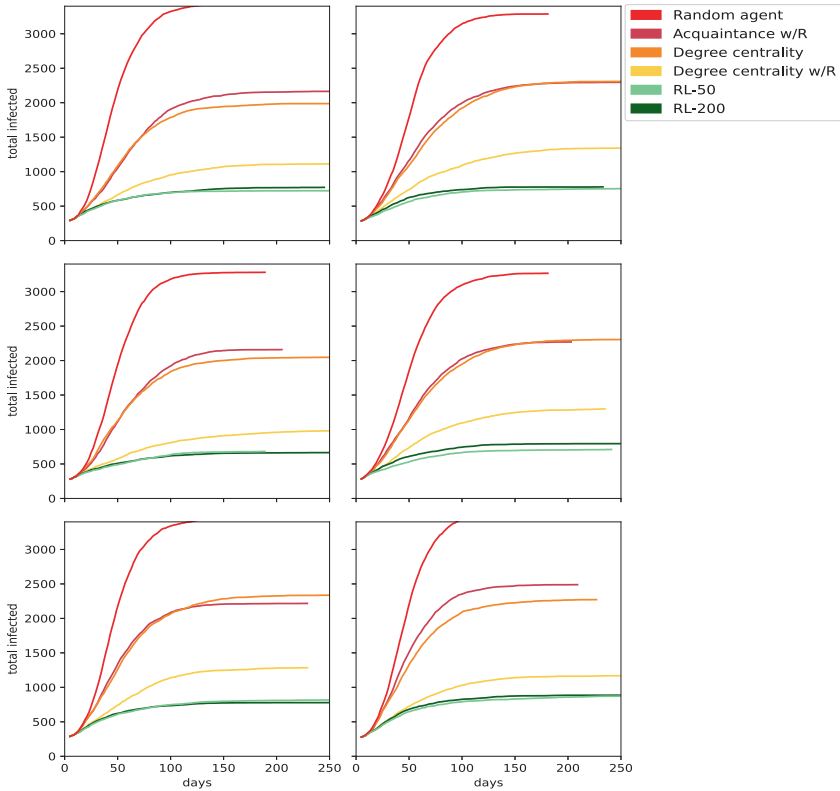


Fig. A3. Epidemic curves for different network and epidemic seeds. These correspond to multiple 5000 nodes Barabási-Albert networks featuring a mean degree of 3, with a testing budget of $k = 1\%$. Here, two versions of the RL agent are displayed: one trained for 50, and one trained for 200 episodes. The y-axis limit is set to 3200 to facilitate the comparisons, yet the random agents perform poorer than this level.

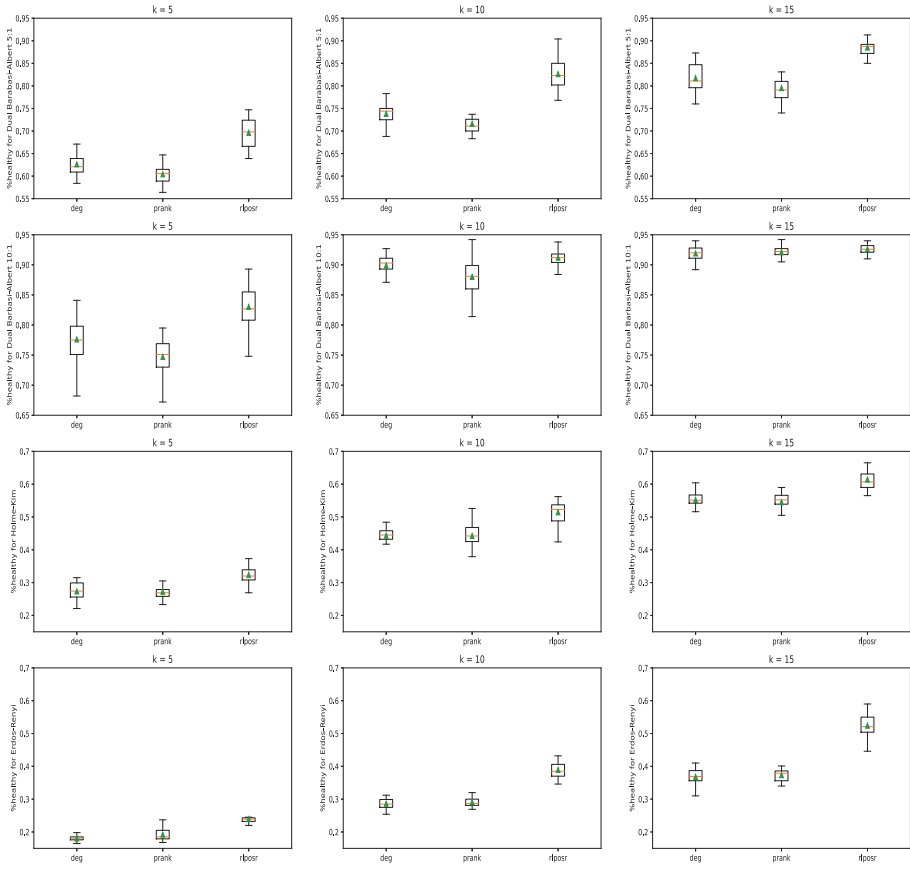


Fig. A4. Infection control performance on different static network architectures with varying budgets. The uncertainties are shown as boxplots.

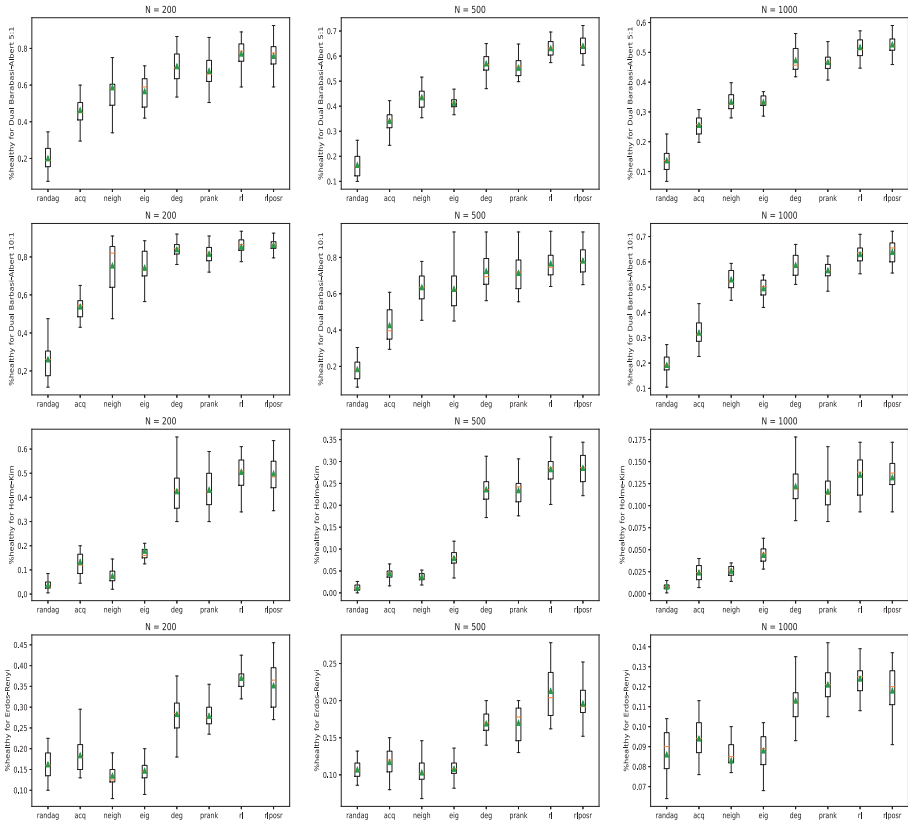


Fig. A5. Infection control performance on different static network architectures and sizes, with a budget of $k = 2$. Uncertainties are shown as boxplots.

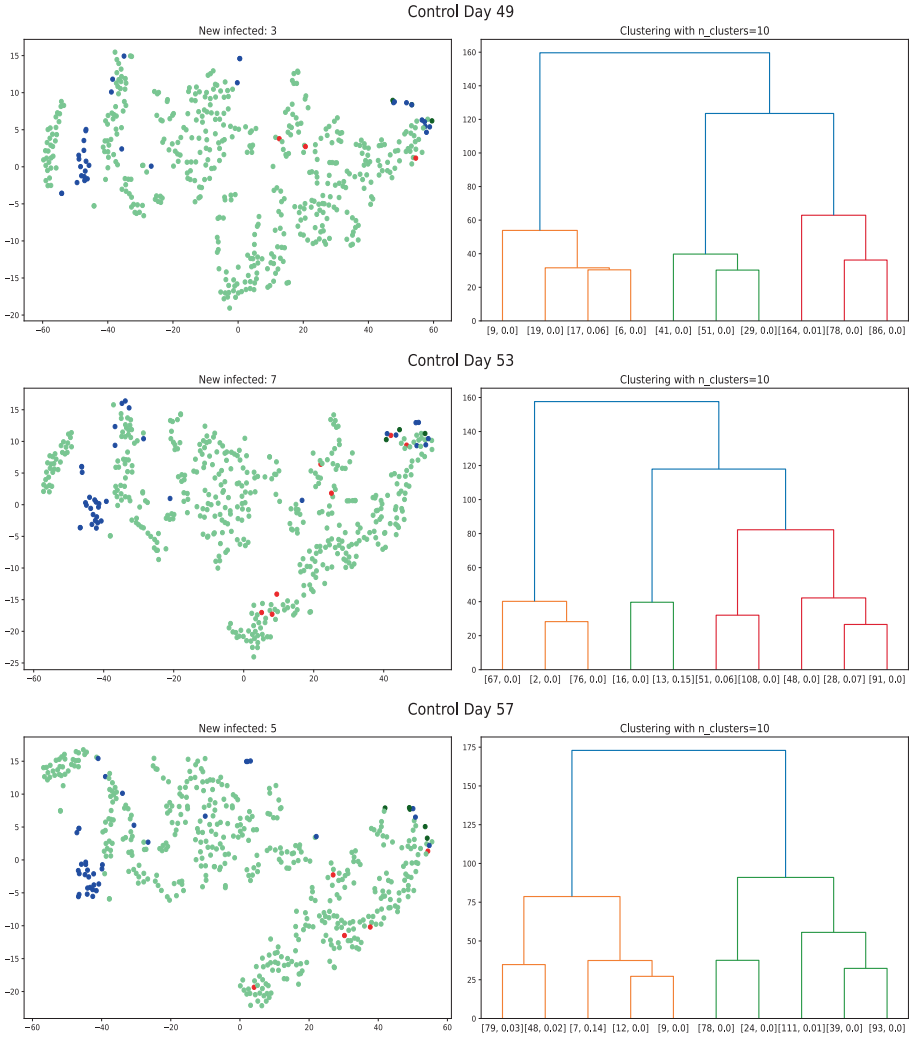


Fig. A6. t-SNE plots of the node hidden states and dendrogram corresponding to their hierarchical clustering into 10 groups. As can be observed, the agent mostly groups detected (blue) nodes in a region of the space, while the new undetected infections (red) are predicted to appear within the risk regions on the right. Recent negative results are plotted as dark green. The dendrogram on the right displays the cardinality and the infection probability associated with each cluster.

A.5 Control Framework

The logic behind our epidemic control framework in the continuous-time simulation scenario is outlined in Algorithm 1. The class hierarchy of the agents, together with their logic, can be consulted in Algorithm 2. Refer to Table A2 for details about the variables involved in these.

Table A2. Legend for the control framework pseudocode.

Name(s)	Description(s)
R	GNN-based ranking model (shared across all epidemics).
E_{conf}	Episode configuration. Consists of tuples mapping an episode ID (e_{id}) to its exploration-control variable e_e .
S_{conf}	Simulation configuration. Enum that defines the maximum network, infection and event seeds, which in turn control the range of the loops over each seeded configuration.
s_{net}, s_{inf}, s_{ev}	Interaction network, infection and event seeds.
N_p, S_p, A_p	Interaction network, simulator and agent hyperparameters. A_p contains the sampling strategy st and learning rate lr .
N, S, A	Interaction network, simulator and agent main objects.
i_c, i_u	Iterators for time-discretized events: dynamic control and edge-updating interaction events.
e, t	Interaction event enum and its corresponding time value.
k_{tst}, k_{ct}	Daily budgets for testing and contact-tracing isolations.
c_{tst}, c_{ct}	Sensible candidates to rank for testing and tracing.
n_{tst}, n_{ct}	Nodes chosen by the agent for testing and tracing.
d	Boolean that determines whether the action is sampled or greedily taken from top-k ranking.
st	Sampling strategy employed by the RLAgent. This can be one of the following: 'softmax', 'escort-transform' [64], 'nvidia-explore' [65].
m	Node ranking scores computed by a specific agent.
v	Epidemic state score computed by the GNN ranking model.
B	Replay buffer for the offline RLAgent.
L	Last step information required by the online RLAgent.
$a, \log \pi_a$	Sampled action and its corresponding log of probability.
r_{t-1}	Reward of previous action taken (i.e. for action sampled and executed at time $t - 1$).

Algorithm 1 Epidemic control framework

```

1: global variables
2:    $R$  ▷ GNN ranking model
3: end global variables
4: procedure RUN_EPIDEMIC( $E_{conf}, S_{conf}, N_p, S_p, A_p$ )
5:   for each  $(e_{id}, e_\epsilon) \in E_{conf}$  do ▷ Episode ID and  $\epsilon$ 
6:     for  $s_{net} \in \{0, \dots, S_{conf}.MAX\_NET\_SEED\}$  do
7:        $N \leftarrow$  INIT_NET( $N_p, s_{net}$ )
8:       ▷  $N$  keeps in memory the edges over all timestamps
9:       for  $s_{inf} \in \{0, \dots, S_{conf}.MAX\_INF\_SEED\}$  do
10:         $S \leftarrow$  INIT_SIMULATOR( $S_p, s_{inf}, N$ )
11:         $A \leftarrow$  INIT_AGENT( $A_p, R, e_{id}, e_\epsilon$ )
12:         $i_c \leftarrow 0$  ▷ Iterator for control timestamps
13:         $i_u \leftarrow 0$  ▷ Iterator for edge-update timestamps
14:        for  $s_{ev} \in \{0, \dots, S_{conf}.MAX\_EVENT\}$  do
15:           $e \leftarrow S.SAMPLE\_NEXT\_EVENT()$ 
16:           $t \leftarrow e.TIME$ 
17:           $S.RUN\_EVENT(e, N)$ 
18:          if  $S.SHOULD\_CONTROL(N, t, i_c)$  then
19:             $i_c \leftarrow \lfloor t \rfloor$  ▷ Floor function
20:             $(n_{tst}, n_{ct}) \leftarrow A.CONTROL(N, i_c)$ 
21:             $S.UPDATE\_STATES(N, n_{tst}, n_{ct})$ 
22:             $i_c \leftarrow i_c + 1$ 
23:          end if
24:          if  $S.SHOULD\_UPDATE\_EDGES(N, t, i_u)$  then
25:             $i_u \leftarrow \lfloor t \rfloor$ 
26:             $N.UPDATE\_EDGES(i_u)$ 
27:             $i_u \leftarrow i_u + 1$ 
28:          end if
29:        end for
30:        ▷ Logging & offline parameter updates (if any)
31:         $A.FINISH(N)$ 
32:      end for
33:    end for
34:  end for
35: end procedure

```

Algorithm 2 Control agents' hierarchy

```

36: struct AGENT
37:    $k_{tst}, k_{ct}$  ▷ Budget for testing and contact tracing
38:   procedure CONTROL( $N, i_c$ )
39:      $c_{tst} \leftarrow$  CANDIDATES.TEST( $N, i_c$ )
40:     ▷ Calls CONTROL_BOTH by default; can be overridden
41:      $n_{tst} \leftarrow$  CONTROL_TEST( $N, c_{tst}, k_{tst}$ )
42:      $c_{ct} \leftarrow$  CANDIDATES_TRACE( $N, i_c, n_{tst}$ )
43:     ▷ Calls CONTROL_BOTH by default; can be overridden
44:      $n_{ct} \leftarrow$  CONTROL_TRACE( $N, c_{ct}, k_{ct}$ )
45:     return ( $n_{tst}, n_{ct}$ )
46:   end procedure
47: end struct
48: struct MEASUREAGENT(AGENT)
49:    $d$  ▷ Boolean controlling if sampling or top-k ranking
50:   procedure CONTROL_BOTH( $N, c, k$ )
51:     ▷ Compute score for each node in  $c$ ; RL samples  $k$  nodes
52:      $m \leftarrow$  COMPUTE_MEASURES( $N, c, k$ )
53:     if  $d$  then
54:       return  $c$ [ARGTOPK( $m, k$ )] ▷ Heap sort top-k ranking
55:     else
56:       return  $m$ 
57:     end if
58:   end procedure
59: end struct
60: struct SLAGENT(MEASUREAGENT)
61:    $lr$  ▷ Learning rate; if 0, evaluation mode is assumed
62:   procedure COMPUTE_MEASURES( $N, c, k$ )
63:     if  $lr > 0$  then
64:       ( $m, v$ )  $\leftarrow$  R.FORWARD( $N$ ) ▷ Message passing
65:       BACKPROP_LOSS( $N, m$ ) ▷ BCE on infection status
66:     else
67:       ( $m, v$ )  $\leftarrow$  R.FORWARD(SUBGRAPH( $N, c$ ))
68:     end if
69:     return  $m$ 
70:   end procedure
71: end struct
72: struct RLAGENT(MEASUREAGENT)
73:    $lr$  ▷ Learning rate; if 0, evaluation mode is assumed
74:    $st$  ▷ Action sampling strategy (e.g. softmax)
75:    $e_\epsilon$  ▷ Sampling noise (i.e.  $\epsilon$ -greedy, softmax temperature)
76:    $B$  ▷ Replay buffer; if null, conduct online learning
77:    $L$  ▷ Keep last step information for online learning
78:   procedure COMPUTE_MEASURES( $N, c, k$ )
79:     if  $lr > 0$  then
80:       ▷ Reward of previous action
81:        $R_{t-1} \leftarrow -N$ .NUM_INFECTED()
82:       ( $m, v$ )  $\leftarrow$  R.FORWARD( $N$ ) ▷ Message passing
83:       ( $a, \log \pi_a$ )  $\leftarrow$  SAMPLE( $m, k, st, e_\epsilon$ ) ▷ Sample action
84:       ▷ Existence of  $B$  determines training online/offline
85:       if  $B$  is null then
86:         ▷ Compute online RL objective
87:         ▷ Using  $m, \{s_{t-1}, a_{t-1}\} \in L$  to compute  $\log \pi_{a_{t-1}}$ 
88:         BACKPROP_LOSS( $R_{t-1}, L, m, v$ )
89:         L.CLEAR()
90:         ▷ Add ( $s_t, a_t, \log \pi_{a_t}, V_t$ ) to one-step buffer
91:         L.ADD( $N, a, \log \pi_a, v$ ) ▷  $\log \pi_a$  used for  $r_t^{SARSA}$ 
92:       else
93:         ▷ Add ( $R_{t-1}, s_t, a_t, \log \pi_{a_t}, V_t$ ) to replay buffer
94:         B.ADD( $R_{t-1}, N, a, \log \pi_a, v$ )
95:       end if
96:     else
97:       ( $m, v$ )  $\leftarrow$  R.FORWARD(SUBGRAPH( $N, c$ ))
98:     end if
99:     return  $m$ 
100:   end procedure
101: end struct

```

References

1. Abueg, M., et al.: Modeling the combined effect of digital exposure notification and non-pharmaceutical interventions on the COVID-19 epidemic in Washington state. In: medRxiv, p. 2020.08.29.20184135. Cold Spring Harbor Laboratory Press (2020). <https://doi.org/10.1101/2020.08.29.20184135>
2. Alon, U., Yahav, E.: On the bottleneck of graph neural networks and its practical implications. In: International Conference on Learning Representations (2022)
3. Andrews, N., et al.: COVID-19 vaccine effectiveness against the omicron (B.1.1.529) variant. *New Engl. J. Med.* **386**(16), 1532–1546 (2022). <https://doi.org/10.1056/NEJMoa2119451>
4. Bao, H., Dong, L., Wei, F.: BEiT: BERT pre-training of image transformers (2021). <https://doi.org/10.48550/arXiv.2106.08254>
5. Bastani, H., et al.: Efficient and targeted COVID-19 border testing via reinforcement learning. *Nature* **599**(7883), 108–113 (2021). <https://doi.org/10.1038/s41586-021-04014-z>, <https://www.nature.com/articles/s41586-021-04014-z>
6. Beaini, D., Passaro, S., Létourneau, V., Hamilton, W.L., Corso, G., Liò, P.: Directional Graph Networks (2021). <https://doi.org/10.48550/arXiv.2010.02863>
7. Bello, I., Pham, H., Le, Q.V., Norouzi, M., Bengio, S.: Neural combinatorial optimization with reinforcement learning (2017). <https://doi.org/10.48550/arXiv.1611.09940>
8. Bodnar, C., Di Giovanni, F., Chamberlain, B.P., Liò, P., Bronstein, M.M.: Neural sheaf diffusion: a topological perspective on heterophily and oversmoothing in GNNs (2022). <https://doi.org/10.48550/arXiv.2202.04579>
9. Braha, D., Bar-Yam, Y.: From centrality to temporary fame: dynamic centrality in complex networks. *Complexity* **12**(2), 59–63 (2006). <https://doi.org/10.1002/cplx.20156>
10. Brody, S., Alon, U., Yahav, E.: How attentive are graph attention networks? (2022). <https://doi.org/10.48550/arXiv.2105.14491>
11. Bronstein, M.: Deep learning on graphs: successes, challenges, and next steps (2022). <https://towardsdatascience.com/deep-learning-on-graphs-successes-challenges-and-next-steps-7d9ec220ba8>
12. Bruxvoort, K.J., et al.: Effectiveness of mRNA-1273 against delta, mu, and other emerging variants of SARS-CoV-2: test negative case-control study. *BMJ* **375**, e068848 (2021). <https://doi.org/10.1136/bmj-2021-068848>
13. Chen, W., Wang, Y., Yang, S.: Efficient influence maximization in social networks. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD 2009, p. 199. ACM Press, Paris (2009). <https://doi.org/10.1145/1557019.1557047>
14. Chung, F., Horn, P., Tsias, A.: Distributing antidote using PageRank vectors. *Internet Math.* **6**(2), 237–254 (2009). <https://doi.org/10.1080/15427951.2009.10129184>
15. Clair, R., Gordon, M., Kroon, M., Reilly, C.: The effects of social isolation on well-being and life satisfaction during pandemic. *Humanit. Soc. Sci. Commun.* **8**(1), 1–6 (2021). <https://doi.org/10.1057/s41599-021-00710-3>
16. Cohen, R., Havlin, S., ben-Avraham, D.: Efficient immunization strategies for computer networks and populations. *Phys. Rev. Lett.* **91**(24), 247901 (2003). <https://doi.org/10.1103/PhysRevLett.91.247901>
17. Dai, H., Khalil, E.B., Zhang, Y., Dilkina, B., Song, L.: Learning combinatorial optimization algorithms over graphs (2018)

18. Davis, E.L., et al.: Contact tracing is an imperfect tool for controlling COVID-19 transmission and relies on population adherence. *Nat. Commun.* **12**(1), 5412 (2021). <https://doi.org/10.1038/s41467-021-25531-5>
19. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs] (2019)
20. Di Domenico, L., Pullano, G., Sabbatini, C.E., Boëlle, P.Y., Colizza, V.: Impact of lockdown on COVID-19 epidemic in Île-de-France and possible exit strategies. *BMC Med.* **18**(1), 240 (2020). <https://doi.org/10.1186/s12916-020-01698-4>
21. Dighe, A., et al.: Response to COVID-19 in South Korea and implications for lifting stringent interventions. *BMC Med.* **18**(1), 321 (2020). <https://doi.org/10.1186/s12916-020-01791-8>
22. Erdős, P., Rényi, A.: On random graphs I. *Publicationes Mathematicae Debrecen* **6**, 290 (1959)
23. Farrahi, K., Emonet, R., Cebrian, M.: Epidemic contact tracing via communication traces. *PLoS ONE* **9**(5), e95133 (2014). <https://doi.org/10.1371/journal.pone.0095133>
24. Ferdinands, J.M.: Waning 2-Dose and 3-dose effectiveness of mRNA vaccines against COVID-19—associated emergency department and urgent care encounters and hospitalizations among adults during periods of delta and omicron variant predominance—VISION network, 10 States, August 2021–January 2022. *MMWR Morbidity Mortality Weekly Rep.* **71** (2022). <https://doi.org/10.15585/mmwr.mm7107e2>
25. Ferguson, N., et al.: Report 9: impact of non-pharmaceutical interventions (NPIs) to reduce COVID19 mortality and healthcare demand. Technical report, Imperial College London (2020). <https://doi.org/10.25561/77482>
26. Ferretti, L., et al.: Quantifying SARS-CoV-2 transmission suggests epidemic control with digital contact tracing. *Science* **368** (2020). <https://doi.org/10.1126/science.abb6936>
27. Fung, V., Zhang, J., Juarez, E., Sumpter, B.: Benchmarking graph neural networks for materials chemistry. *NPJ Comput. Mater.* **7**, 84 (2021). <https://doi.org/10.1038/s41524-021-00554-0>
28. Gillespie, D.T.: Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.* **81**(25), 2340–2361 (1977). <https://doi.org/10.1021/j100540a008>
29. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry (2017). <https://doi.org/10.48550/arXiv.1704.01212>
30. Gori, M., Monfardini, G., Scarselli, F.: A new model for learning in graph domains. In: *Proceedings of the International Joint Conference on Neural Networks*, vol. 2, pp. 729–734 (2005). <https://doi.org/10.1109/IJCNN.2005.1555942>
31. Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor (2018)
32. Hamilton, W.L., Ying, R., Leskovec, J.: Inductive representation learning on large graphs. arXiv:1706.02216 [cs, stat] (2018)
33. He, J., et al.: Deep reinforcement learning with a combinatorial action space for predicting popular reddit threads. In: *EMNLP* (2019)
34. Henley, J.: COVID surges across Europe as experts warn not let guard down. *The Guardian* (2022). <https://www.theguardian.com/world/2022/jun/21/covid-surges-europe-ba4-ba5-cases>
35. Hinch, R., et al.: Effective configurations of a digital contact tracing app: a report to NHSX. Technical report (2020)

36. Hoang, N., Maehara, T.: Revisiting graph neural networks: all we have is low-pass filters. arXiv:1905.09550 [cs, math, stat] (2019)
37. Holland, P.W., Laskey, K.B., Leinhardt, S.: Stochastic blockmodels: first steps. *Soc. Netw.* **5**(2), 109–137 (1983). [https://doi.org/10.1016/0378-8733\(83\)90021-7](https://doi.org/10.1016/0378-8733(83)90021-7)
38. Holme, P., Kim, B.J.: Growing scale-free networks with tunable clustering. *Phys. Rev. E* **65**(2), 026107 (2002). <https://doi.org/10.1103/PhysRevE.65.026107>
39. Huang, Q., Yamada, M., Tian, Y., Singh, D., Yin, D., Chang, Y.: GraphLIME: local interpretable model explanations for graph neural networks (2020). <https://doi.org/10.48550/arXiv.2001.06216>
40. Huerta, R., Tsimring, L.S.: Contact tracing and epidemics control in social networks. *Phys. Rev. E* **66**(5), 056115 (2002). <https://doi.org/10.1103/PhysRevE.66.056115>
41. Jhun, B.: Effective vaccination strategy using graph neural network ansatz (2021). <https://doi.org/10.48550/arXiv.2111.00920>
42. Joffe, A.R.: COVID-19: rethinking the lockdown groupthink. *Front. Public Health* **9** (2021)
43. Joshi, C.K., Laurent, T., Bresson, X.: An efficient graph convolutional network technique for the travelling salesman problem (2019)
44. Kapoor, A., et al.: Examining COVID-19 forecasting using spatio-temporal graph neural networks. arXiv:2007.03113 [cs] (2020)
45. Kermack, W.O., McKendrick, A.G., Walker, G.T.: A contribution to the mathematical theory of epidemics. *Proc. Roy. Soc. London Ser. A Containing Pap. Math. Phys. Character* **115**(772), 700–721 (1927). <https://doi.org/10.1098/rspa.1927.0118>
46. Khan, S., Naseer, M., Hayat, M., Zamir, S.W., Khan, F.S., Shah, M.: Transformers in vision: a survey. *ACM Comput. Surv.* 3505244 (2022). <https://doi.org/10.1145/3505244>
47. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization (2017). <https://doi.org/10.48550/arXiv.1412.6980>
48. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, 24–26 April 2017. Conference Track Proceedings (2017). [OpenReview.net](https://openreview.net)
49. Kiran, B.R., et al.: Deep reinforcement learning for autonomous driving: a survey. *IEEE Trans. Intell. Transp. Syst.* **23**(6), 4909–4926 (2022). <https://doi.org/10.1109/TITS.2021.3054625>
50. Kobayashi, T.: Adaptive and multiple time-scale eligibility traces for online deep reinforcement learning. *Robot. Auton. Syst.* **151**, 104019 (2022). <https://doi.org/10.1016/j.robot.2021.104019>
51. Kojaku, S., Hébert-Dufresne, L., Mones, E., Lehmann, S., Ahn, Y.Y.: The effectiveness of backward contact tracing in networks. *Nat. Phys.* **17**(5), 652–658 (2021). <https://doi.org/10.1038/s41567-021-01187-2>
52. Konda, V., Tsitsiklis, J.: Actor-critic algorithms. In: *Advances in Neural Information Processing Systems*, vol. 12. MIT Press (1999)
53. Kool, W., van Hoof, H., Welling, M.: Attention, learn to solve routing problems! (2019). <https://doi.org/10.48550/arXiv.1803.08475>
54. Lazaridis, A., Fachantidis, A., Vlahavas, I.: Deep reinforcement learning: a state-of-the-art walkthrough. *J. Artif. Intell. Res.* **69**, 1421–1471 (2020). <https://doi.org/10.1613/jair.1.12412>

55. Leung, K., Wu, J.T.: Managing waning vaccine protection against SARS-CoV-2 variants. *Lancet* **399**(10319), 2–3 (2022). [https://doi.org/10.1016/S0140-6736\(21\)02841-5](https://doi.org/10.1016/S0140-6736(21)02841-5)
56. Lillicrap, T.P., et al.: Continuous control with deep reinforcement learning. Paper presented at 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico (2016)
57. Liu, D., Jing, Y., Zhao, J., Wang, W., Song, G.: A fast and efficient algorithm for mining top-k nodes in complex networks. *Sci. Rep.* **7**(1), 43330 (2017). <https://doi.org/10.1038/srep43330>
58. Lundberg, S., Lee, S.I.: A unified approach to interpreting model predictions (2017). <https://doi.org/10.48550/arXiv.1705.07874>
59. Madan, A., Cebrian, M., Moturu, S., Farrahi, K., Pentland, A.S.: Sensing the “health state” of a community. *IEEE Pervasive Comput.* **11**(4), 36–45 (2012). <https://doi.org/10.1109/MPRV.2011.79>
60. Martinez-Garcia, M., Sansano-Sansano, E., Castillo-Hornero, A., Femenia, R., Roomp, K., Oliver, N.: Social isolation during the COVID-19 pandemic in Spain: a population study (2022). <https://doi.org/10.1101/2022.01.22.22269682>
61. Mason, R., Allegretti, A., Devlin, H., Sample, I.: UK treasury pushes to end most free Covid testing despite experts’ warnings. *The Guardian* (2022)
62. Masuda, N.: Immunization of networks with community structure. *New J. Phys.* **11**(12), 123018 (2009). <https://doi.org/10.1088/1367-2630/11/12/123018>
63. Matrajt, L., Leung, T.: Evaluating the effectiveness of social distancing interventions to delay or flatten the epidemic curve of coronavirus disease. *Emerg. Infect. Dis.* **26**(8), 1740–1748 (2020). <https://doi.org/10.3201/eid2608.201093>
64. Mei, J., Xiao, C., Dai, B., Li, L., Szepesvari, C., Schuurmans, D.: Escaping the gravitational pull of softmax. In: *Advances in Neural Information Processing Systems*, vol. 33, pp. 21130–21140. Curran Associates, Inc. (2020)
65. Meiroum, E., Maron, H., Mannor, S., Chechik, G.: Controlling graph dynamics with reinforcement learning and graph neural networks. In: *Proceedings of the 38th International Conference on Machine Learning*, pp. 7565–7577. PMLR (2021)
66. Meiroum, E., Milling, C., Caramanis, C., Mannor, S., Shakkottai, S., Orda, A.: Localized epidemic detection in networks with overwhelming noise. *ACM SIGMETRICS Perform. Eval. Rev.* **43**(1), 441–442 (2015). <https://doi.org/10.1145/2796314.2745883>
67. Mercer, T.R., Salit, M.: Testing at scale during the COVID-19 pandemic. *Nat. Rev. Genet.* **22**(7), 415–426 (2021). <https://doi.org/10.1038/s41576-021-00360-w>
68. Miller, J.C., Hyman, J.M.: Effective vaccination strategies for realistic social networks. *Phys. A* **386**(2), 780–785 (2007). <https://doi.org/10.1016/j.physa.2007.08.054>
69. Mnih, V., et al.: Playing atari with deep reinforcement learning (2013). <https://doi.org/10.48550/arXiv.1312.5602>
70. Mnih, V., et al.: Human-level control through deep reinforcement learning. *Nature* **518**(7540), 529–533 (2015). <https://doi.org/10.1038/nature14236>
71. Morris, C., et al.: Weisfeiler and leman go neural: higher-order graph neural networks. *arXiv:1810.02244 [cs, stat]* (2020)
72. Moshiri, N.: The dual-Barabási-Albert model (2018)
73. Murata, T., Koga, H.: Extended methods for influence maximization in dynamic networks. *Comput. Soc. Netw.* **5**(1), 1–21 (2018). <https://doi.org/10.1186/s40649-018-0056-8>
74. Oono, K., Suzuki, T.: Graph neural networks exponentially lose expressive power for node classification. *arXiv:1905.10947 [cs, stat]* (2021)

75. Panagopoulos, G., Nikolentzos, G., Vazirgiannis, M.: Transfer graph neural networks for pandemic forecasting. arXiv:2009.08388 [cs, stat] (2021)
76. Pandit, J.A., Radin, J.M., Quer, G., Topol, E.J.: Smartphone apps in the COVID-19 pandemic. *Nat. Biotechnol.* **40**(7), 1013–1022 (2022). <https://doi.org/10.1038/s41587-022-01350-x>
77. Preciado, V.M., Zargham, M., Enyioha, C., Jadbabaie, A., Pappas, G.J.: Optimal resource allocation for network protection against spreading processes. *IEEE Trans. Control Netw. Syst.* **1**(1), 99–108 (2014). <https://doi.org/10.1109/TCNS.2014.2310911>
78. Puterman, M.L.: *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 1st edn. Wiley, USA (1994)
79. Rayner, D.C., Sturtevant, N.R., Bowling, M.: Subset selection of search heuristics. In: *IJCAI* (2019)
80. Ribeiro, M.T., Singh, S., Guestrin, C.: “Why should i trust you?”: explaining the predictions of any classifier (2016). <https://doi.org/10.48550/arXiv.1602.04938>
81. Rimmer, A.: Sixty seconds on . . . the pingdemic. *BMJ* **374**, n1822 (2021). <https://doi.org/10.1136/bmj.n1822>
82. Rummery, G., Niranjan, M.: On-line Q-learning using connectionist systems. Technical report CUED/F-INFENG/TR 166 (1994)
83. Rusu, A., Farrahi, K., Emonet, R.: Modelling digital and manual contact tracing for COVID-19 Are low uptakes and missed contacts deal-breakers? Preprint. *Epidemiology* (2021). <https://doi.org/10.1101/2021.04.29.21256307>
84. Salathé, M., Jones, J.H.: Dynamics and control of diseases in networks with community structure. *PLOS Comput. Biol.* **6**(4), e1000736 (2010). <https://doi.org/10.1371/journal.pcbi.1000736>, <https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1000736>
85. Sato, R., Yamada, M., Kashima, H.: Random features strengthen graph neural networks (2021). <https://doi.org/10.48550/arXiv.2002.03155>
86. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. *IEEE Trans. Neural Netw.* **20**(1), 61–80 (2009). <https://doi.org/10.1109/TNN.2008.2005605>
87. Schulman, J., Moritz, P., Levine, S., Jordan, M., Abbeel, P.: High-dimensional continuous control using generalized advantage estimation (2018). <https://doi.org/10.48550/arXiv.1506.02438>
88. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv:1707.06347 [cs] (2017)
89. Serafino, M., et al.: Digital contact tracing and network theory to stop the spread of COVID-19 using big-data on human mobility geolocalization. *PLOS Comput. Biol.* **18**(4), e1009865 (2022). <https://doi.org/10.1371/journal.pcbi.1009865>
90. Shah, C., et al.: Finding patient zero: learning contagion source with graph neural networks (2020)
91. Sigal, A.: Milder disease with Omicron: is it the virus or the pre-existing immunity? *Nat. Rev. Immunol.* **22**(2), 69–71 (2022). <https://doi.org/10.1038/s41577-022-00678-4>
92. Silver, D., et al.: Mastering the game of Go with deep neural networks and tree search. *Nature* **529**(7587), 484–489 (2016). <https://doi.org/10.1038/nature16961>
93. Silver, D., et al.: Mastering chess and shogi by self-play with a general reinforcement learning algorithm (2017). <https://doi.org/10.48550/arXiv.1712.01815>
94. Smith, J.: Demand for Covid vaccines falls amid waning appetite for booster shots. *Financial Times* (2022). <https://www.ft.com/content/9ac9f8fc-1ab3-4cb2-81bf-259ba612f600>

95. Smith, R.L., et al.: Longitudinal assessment of diagnostic test performance over the course of acute SARS-CoV-2 infection. *J. Infect. Dis.* **224**(6), 976–982 (2021). <https://doi.org/10.1093/infdis/jiab337>
96. Song, H., et al.: Solving continual combinatorial selection via deep reinforcement learning. In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pp. 3467–3474 (2019). <https://doi.org/10.24963/ijcai.2019/481>
97. Su, Z., Cheshmehzangi, A., McDonnell, D., da Veiga, C.P., Xiang, Y.T.: Mind the “Vaccine Fatigue”. *Front. Immunol.* **13** (2022)
98. Sukumar, S.R., Nutaro, J.J.: Agent-based vs. equation-based epidemiological models: a model selection case study. In: *2012 ASE/IEEE International Conference on BioMedical Computing (BioMedCom)*, pp. 74–79 (2012). <https://doi.org/10.1109/BioMedCom.2012.19>
99. Sutton, R.S.: Learning to predict by the methods of temporal differences. *Mach. Learn.* **3**(1), 9–44 (1988). <https://doi.org/10.1007/BF00115009>
100. Sutton, R.S., Barto, A.G.: *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning Series, 2nd edn. The MIT Press, Cambridge (2018)
101. Tian, S., Mo, S., Wang, L., Peng, Z.: Deep reinforcement learning-based approach to tackle topic-aware influence maximization. *Data Sci. Eng.* **5**(1), 1–11 (2020). <https://doi.org/10.1007/s41019-020-00117-1>
102. Tomy, A., Razzanelli, M., Di Lauro, F., Rus, D., Della Santina, C.: Estimating the state of epidemics spreading with graph neural networks. *Nonlinear Dyn.* **109**(1), 249–263 (2022). <https://doi.org/10.1007/s11071-021-07160-1>
103. van Hasselt, H., Madjihurem, S., Hessel, M., Silver, D., Barreto, A., Borsa, D.: Expected eligibility traces. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 9997–10005 (2021). <https://doi.org/10.1609/aaai.v35i11.17200>
104. Vaswani, A., et al.: Attention is all you need. In: *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc. (2017)
105. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. *arXiv:1710.10903 [cs, stat]* (2018)
106. Watkins, C.: *Learning from delayed rewards* (1989)
107. Wymant, C., et al.: The epidemiological impact of the NHS COVID-19 app. *Nature* **594**(7863), 408–412 (2021). <https://doi.org/10.1038/s41586-021-03606-z>
108. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? *arXiv:1810.00826 [Cs, Stat]* (2019)
109. Yamada, M., Jitkrittum, W., Sigal, L., Xing, E.P., Sugiyama, M.: High-dimensional feature selection by feature-wise kernelized lasso. *Neural Comput.* **26**(1), 185–207 (2014). https://doi.org/10.1162/NECO_a.00537
110. Zhou, J., et al.: Graph neural networks: a review of methods and applications. *AI Open* **1**, 57–81 (2020). <https://doi.org/10.1016/j.aiopen.2021.01.001>