PROCEEDINGS A

royalsocietypublishing.org/journal/rspa

Research

Cite this article: Darvariu V-A, Hailes S, Musolesi M. 2021 Goal-directed graph construction using reinforcement learning. *Proc. R. Soc. A* **477**: 20210168. https://doi.org/10.1098/rspa.2021.0168

Received: 23 February 2021
Accepted: 29 September 2021

Subject Areas:
complexity, artificial intelligence

Keywords:
network robustness, complex systems, graph neural networks, reinforcement learning

Author for correspondence:
Victor-Alexandru Darvariu
e-mail: v.darvariu@ucl.ac.uk

Electronic supplementary material is available online at https://doi.org/10.6084/m9.figshare.c.5672367.

THE ROYAL SOCIETY PUBLISHING

Goal-directed graph construction using reinforcement learning

Victor-Alexandru Darvariu[1,2], Stephen Hailes[1] and Mirco Musolesi[1,2,3]

[1]Department of Computer Science, University College London, London, UK
[2]The Alan Turing Institute, London, UK
[3]Department of Computer Science and Engineering, University of Bologna, Bologna, Italy

V-AD, 0000-0001-9250-8175; MM, 0000-0001-9712-4090

Graphs can be used to represent and reason about systems and a variety of metrics have been devised to quantify their global characteristics. However, little is currently known about how to construct a graph or improve an existing one given a target objective. In this work, we formulate the construction of a graph as a decision-making process in which a central agent creates topologies by trial and error and receives rewards proportional to the value of the target objective. By means of this conceptual framework, we propose an algorithm based on reinforcement learning and graph neural networks to learn graph construction and improvement strategies. Our core case study focuses on robustness to failures and attacks, a property relevant for the infrastructure and communication networks that power modern society. Experiments on synthetic and real-world graphs show that this approach can outperform existing methods while being cheaper to evaluate. It also allows generalization to out-of-sample graphs, as well as to larger out-of-distribution graphs in some cases. The approach is applicable to the optimization of other global structural properties of graphs.

1. Introduction

Graphs are mathematical abstractions that can be used to model a variety of systems, from infrastructure and biological networks to social structures. Various methods

for analysing networks have been developed: these have often been used for understanding the systems themselves and range from mathematical models of how families of graphs are generated [1,2] to measures of centrality for capturing the roles of vertices [3] and global network characteristics [4], to name but a few.

A measure that has attracted significant interest from researchers and practitioners is *robustness* [5] (sometimes called *resilience*), which is typically defined as the capacity of the graph to withstand random failures, targeted attacks on key nodes or some combination thereof. A network is considered robust if a large fraction (*critical fraction*) of nodes have to be removed before it becomes disconnected [6], its diameter increases [7] or its largest connected component diminishes in size [8]. Previous work has focused on the robustness of communication networks such as the Internet [9] and infrastructure networks used for transportation and energy distribution [10], for which resilience is a key property.

In many practical cases, an initial network is given and the only way of improving its robustness is through the modification of its structure. This problem was first approached by considering edge addition or rewiring, based on random and preferential (w.r.t. node degree) modifications [8]. Alternatively, a strategy has been proposed that uses a 'greedy' modification scheme based on random edge selection and swapping if the resilience metric improves [11]. Another line of work focuses on the spectral decomposition of the graph Laplacian, and using properties such as the algebraic connectivity [12] and effective graph resistance [13] to guide modifications. While simple and interpretable, these strategies may not yield the best solutions or generalize across networks with varying characteristics and sizes. Certainly, better solutions may be found by exhaustive search, but the time complexity of exploring all the possible topologies and the cost of computing the metric render this strategy infeasible. With the goal of discovering better strategies than existing methods, we ask whether *generalizable network construction strategies for improving robustness can be learned*.

Starting from this motivation, we formalize the process of graph construction and improvement as a Markov decision process (MDP) in which rewards are proportional to the value of a graph-level objective function. We consider two objective functions that quantify robustness as the critical fraction of the network in the presence of random failures and targeted attacks. Inspired by recent successes of reinforcement learning (RL) in solving combinatorial optimization problems on graphs [14,15], we make use of graph neural network (GNN) architectures [16] together with the Deep Q-network (DQN) [17] algorithm. Recent work in goal-directed graph generation and improvement considers performing edge additions for adversarially attacking GNN classifiers [18] and generating molecules with certain desirable properties using domain-specific rewards [19]. By contrast, to the best of our knowledge, this is the first time that RL has been used *to learn how to construct a graph such as to optimize a global structural property*. While, in this paper, we focus on robustness, other intrinsic global properties of graphs, such as efficiency [20] or communicability [21], could be used as optimization targets.

The contribution of this paper is twofold. Firstly, we propose a framework for improving global structural properties of graphs, by introducing the graph construction Markov decision process (GC-MDP). Secondly, focusing on the robustness of graphs under failures and attacks as a core case study, we offer an in-depth empirical evaluation that demonstrates significant advantages over existing approaches in this domain, in terms of both the quality of the solutions found and the time complexity of model evaluation. Since this approach addresses the problem of building robust networks with a DQN, we name it *RNet–DQN*.

The remainder of the paper is structured as follows. We provide the definitions of the GC-MDP and the robustness measures in §2. Section 3 describes state and action representations for deep RL using GNNs. We present our experimental set-up in §4, and discuss our main results in §5. In §6, we review and compare the key works in this area. Finally, we conclude and offer a discussion of avenues for future work in §7.

2. Robust graph construction as a decision-making problem

(a) MDP preliminaries

An MDP is one possible formalization of a decision-making process. The decision-maker, called an *agent*, interacts with an *environment*. When in a *state* $s \in \mathcal{S}$, the agent must take an *action* a out of the set $\mathcal{A}(s)$ of valid ones, receiving a *reward* r governed by the reward function $\mathcal{R}(s,a)$. Finally, the agent finds itself in a new state s', depending on a transition model $\mathcal{P}$ that governs the joint probability distribution $P(s',a,s)$ of transitioning to state s' after taking action a in state s. This sequence of interactions gives rise to a *trajectory*. The agent's goal is to maximize the expected (possibly discounted) sum of rewards it receives over all the trajectories. The tuple $(\mathcal{S},\mathcal{A},\mathcal{P},\mathcal{R},\gamma)$ defines this MDP, where $\gamma \in [0,1]$ is a discount factor. We also define a *policy* $\pi(a|s)$, i.e. a distribution of actions over states, which fully determines the behaviour of the agent. Given a policy π, the *action-value function* $Q_\pi(s,a)$ is defined as the expected return when starting from s, taking action a and subsequently following policy π.

(b) Modelling graph construction

Let $\mathbf{G}^{(N)}$ be the set of labelled, undirected, unweighted graphs with N nodes; each such graph $G=(V,E)$ consists of a vertex set V and edge set E. Let $\mathbf{G}^{(N,m)}$ be the subset of $\mathbf{G}^{(N)}$ with $|E|=m$. We also let $\mathcal{F}:\mathbf{G}^{(N)}\to[0,1]$ be an objective function, and $L\in\mathbb{N}$ be a modification budget. Given an initial graph $G_0=(V,E_0)\in\mathbf{G}^{(N,m_0)}$, the aim is to perform a series of L edge additions to G_0 such that the resulting graph $G_*=(V,E_*)$ satisfies

$$G_* = \arg\max_{G'\in\mathbf{G}'} \mathcal{F}(G'),$$

$$\text{where}\quad \mathbf{G}' = \{G=(V,E)\in\mathbf{G}^{(N,m_0+L)} \mid E_0\subset E\}.$$

This combinatorial optimization problem can be cast as a sequential decision-making process. In order to enable scaling to large graphs, the agent has to select a node at each step, and an edge is added to the graph after every two decisions [18]. Tasks are episodic; each episode proceeds for at most $2L$ steps. A trajectory visualization is shown in figure 1. Formally, we define the GC-MDP as follows.

(i) *State*: The state S_t is a tuple (G_t,σ_t) containing the graph $G_t=(V,E_t)$ and an *edge stub* σ_t. σ_t can be either the empty set $\varnothing$ or the singleton $\{v\}$, where $v\in V$.

(ii) *Action*: A_t corresponds to the selection of a node in V. Letting the degree of node v be d_v, available actions are defined as

$$\mathcal{A}(S_t=((V,E_t),\varnothing)) = \{v\in V \mid d_v < |V|-1\}$$

and

$$\mathcal{A}(S_t=((V,E_t),\{\sigma_t\})) = \{v\in V \mid (\sigma_t,v)\notin E_t\}.$$

(iii) *Transitions*: The transition model is defined as $P(S_t=s' \mid S_{t-1}=s, A_{t-1}=a)=\delta_{S_tS'}$,

$$\text{where } S' = \begin{cases} ((V,E_{t-1}\cup(\sigma_{t-1},a)),\varnothing), & \text{if } 2\mid t \\ ((V,E_{t-1}),\{a\}), & \text{otherwise.}\end{cases}$$

(iv) *Reward*: The reward R_t is defined as follows:[1]

$$R_t = \begin{cases} \mathcal{F}(G_t)-\mathcal{F}(G_0), & \text{if } t=2L \\ 0, & \text{otherwise.}\end{cases}$$

[1]Since $\mathcal{F}$ is very expensive to estimate, we deliberately only provide the reward at the end of the episode in order to make the training feasible computationally, to the detriment of possible credit assignment issues. Intermediate rewards based on the true objective or a related quantity represent a middle ground, which we leave for future work.

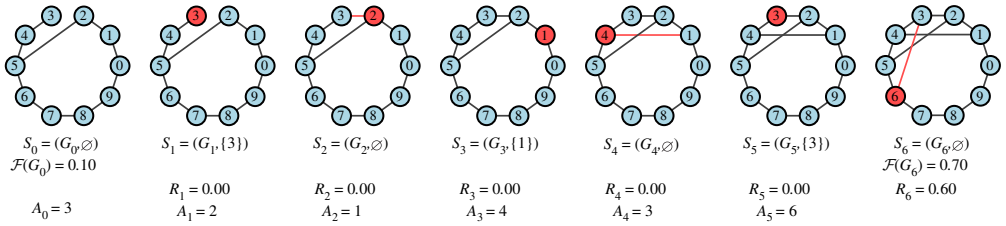

Figure 1. Illustration of a GC-MDP trajectory. The agent is provided with a start state $S_0 = (G_0, \varnothing)$. It must make $L = 3$ edge additions over a sequence of six node selections (actions A_t), receiving rewards R_t proportional to the value of an objective function $\mathcal{F}$ applied to the graph. In this case, $\mathcal{F}$ quantifies the robustness of the network to targeted node removal, computed by removing nodes in decreasing order of their degree and in decreasing order of the labels if two nodes have the same degree. We observe an improvement in the robustness of the graph from $\mathcal{F}(G_0) = 0.1$ to $\mathcal{F}(G_6) = 0.7$. Actions and the corresponding edges are highlighted. (Online version in colour.)

(c) Definition of objective functions for robustness

We are interested in the robustness of graphs as objective functions. Given a graph G, we let the *critical fraction* $p(G, \xi) \in [0, 1]$ be the minimum fraction of nodes that have to be removed from G in some order ξ for it to become disconnected (i.e. have more than one connected component). Connectedness is a crucial operational constraint: the higher this fraction is, the more robust the graph can be said to be.[2] The order ξ in which nodes are removed can have an impact on p, and corresponds to different scenarios: random removal is typically used to model arbitrary failures, while targeted removal is adopted as a model for attack. Formally, we consider both random permutations ξ_{random} of nodes in G, as well as permutations ξ_{targeted}, which are subject to the constraint that nodes must appear in the order of their degree, i.e.

$$\forall v, u \in V.\ \xi_{\text{targeted}}(v) \leq \xi_{\text{targeted}}(u) \iff d_v \geq d_u.$$

We define the objective functions $\mathcal{F}$ in the following way:

(i) *Expected critical fraction to random removal*:

$$\mathcal{F}_{\text{random}}(G) = \mathbb{E}_{\xi_{\text{random}}}[p(G, \xi_{\text{random}})].$$

(ii) *Expected critical fraction to targeted removal*:

$$\mathcal{F}_{\text{targeted}}(G) = \mathbb{E}_{\xi_{\text{targeted}}}[p(G, \xi_{\text{targeted}})].$$

We use Monte Carlo (MC) sampling for estimating these quantities. For completeness, algorithm 1 in the electronic supplementary material describes how the simulations are performed. In the remainder of the paper, we use $\mathcal{F}_{\text{random}}(G)$ and $\mathcal{F}_{\text{targeted}}(G)$ to indicate their estimates obtained in this way. We highlight that evaluating an MC sample has time complexity $O(|V| \times (|V| + |E|))$: it involves checking connectedness (an $O(|V| + |E|)$ operation) after the removal of each of the $O(|V|)$ nodes. Typically, many such samples need to be used to obtain a low-variance estimate of the quantities. Coupled with the number of possible topologies, the high cost renders even shallow search methods infeasible in this domain.

3. Learning to build robust graphs with function approximation

While the problem formulation described in §2 may allow us to work with a tabular RL method, the number of states quickly becomes intractable; for example, there are approximately 10^{57}

[2]We note that while connectedness is required for the specific objective functions considered in this work, it is not required by either the GC-MDP formulation or the learning mechanism itself. Other robustness objectives that quantify, for example, the size of the largest connected component are also applicable, as are fundamentally different objectives.

labelled, connected graphs with 20 vertices [22]. Thus, we require a means of considering graph properties that are label-agnostic, permutation-invariant and generalize across similar states and actions. Graph neural network architectures address these requirements. In particular, we use a graph representation based on a variant of structure2vec (S2V) [23], a GNN architecture inspired by mean-field inference in graphical models.[3] Given an input graph $G = (V, E)$ where nodes $v \in V$ have feature vectors $\mathbf{x}_v$, its objective is to produce for each node v an embedding vector μ_v that captures the structure of the graph as well as interactions between neighbours. This is performed in several rounds of aggregating the features of neighbours and applying an element-wise nonlinear activation function such as the rectified linear unit. For each round $k \in \{1, 2, \ldots, K\}$, the network simultaneously applies updates of the form

$$\mu_v^{(k+1)} = \text{relu}\left(\theta^{(1)}\mathbf{x}_v + \theta^{(2)} \sum_{u \in \mathcal{N}(v)} \mu_u^{(k)}\right),$$

where $\mathcal{N}(v)$ is the neighbourhood of node v. We initialize embeddings with $\mu_v^{(0)} = \mathbf{0}\ \forall v \in V$, and let $\mu_v = \mu_v^{(K)}$. Once node-level embeddings are obtained, permutation-invariant embeddings for a subgraph $\mathcal{S}$ can be derived by summing the node-level embeddings: $\mu(\mathcal{S}) = \sum_{v_i \in \mathcal{S}} \mu_{v_i}$. The node features $\mathbf{x}_v$ are one-hot two-dimensional vectors representing whether v is the edge stub, and their use is required to satisfy the Markovian assumption behind the MDP framework (the agent 'commits' to selecting v, which is now part of its present state). Each state contains at most one edge stub.

In Q-learning [26], the agent estimates the action-value function $Q(s, a)$ introduced earlier, and derives a deterministic policy that acts greedily with respect to it. The agent interacts with the environment and updates its estimates according to

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a' \in \mathcal{A}(s')} Q(s', a') - Q(s, a)].$$

During learning, exploratory random actions are taken with probability ϵ. In the case of high-dimensional state and action spaces, approaches that use a neural network to estimate $Q(s, a)$ have been successful in a variety of domains ranging from general game-playing to continuous control [17,27]. In particular, we use the DQN algorithm: a sample-efficient method that improves on neural fitted Q-iteration [28] by use of an experience replay buffer and an iteratively updated target network for action-value function estimation. Specifically, we use two parametrizations of the Q-function depending on whether the state S_t contains an edge stub

$$Q(S_t = (G_t, \varnothing), A_t) = \theta^{(3)}\text{relu}\,(\theta^{(4)}[\mu_{A_t}, \mu(G_t)])$$

and

$$Q(S_t = (G_t, \{\sigma_t\}), A_t) = \theta^{(5)}\text{relu}\,(\theta^{(6)}[\mu_{\sigma_t}, \mu_{A_t}, \mu(G_t)]),$$

where $[\cdot, \cdot]$ represents concatenation. This lets the model learn *combinations* of relevant node features (e.g. that connecting two central nodes has high Q-value). The use of GNNs has several advantages: firstly, the parameters $\Theta = \{\theta^{(i)}\}_{i=1}^{6}$ can be learned in a goal-directed fashion for the RL objective, allowing for flexibility in the learned representation. Secondly, the embeddings have the potential to generalize to larger graphs since they control how to combine node features of neighbours in the message-passing rounds and are not restricted to graphs of a particular size. We note that the underlying S2V parameters $\theta^{(1)}, \theta^{(2)}$ are shared between the two Q-function parametrizations.

[3]The problem formulation does not depend on the specific GNN or RL algorithm used. While further advances developed by the community in these areas [24,25] can be incorporated, in this paper we focus on aspects specific to the challenges of optimizing the global properties of graphs.

4. Experimental set-up

(a) Learning environment

We build a learning environment that allows for the definition of an arbitrary graph objective function $\mathcal{F}$ and provides a standardized interface for agents. Our implementation of the environment, RNet–DQN and baseline agents, and experimental suite is provided as a code repository containing Docker image blueprints that enable the reproduction of the results presented herein (up to hardware differences), including the relevant tables and figures. The instructions about how to obtain, configure and run the code are provided in the electronic supplementary material.

(b) Baselines

We compare against the following approaches:

- *Random*: This strategy randomly selects an available action.
- *Greedy*: This strategy uses lookahead and selects the action that gives the biggest improvement in the estimated value of $\mathcal{F}$ over one edge addition.
- *Preferential*: Previous works have considered preferential additions between nodes with the two lowest degrees [8], connecting a node with the lowest degree to a random node [12] or connecting the two nodes with the lowest degree product [13], i.e. adding an edge between the vertices v, u that satisfy $\operatorname{argmin}_{v,u} d_v \cdot d_u$. We find the latter works best in all settings tested, and refer to it as *LDP*.
- *Fiedler vector* (*FV*): The concept of FV was introduced by [29] and for robustness improvement by [12]. This strategy adds an edge between the vertices v, u that satisfy $\operatorname{argmax}_{v,u} |\mathbf{y}_v - \mathbf{y}_u|$, where $\mathbf{y}$ is the FV, i.e. the eigenvector of the graph Laplacian $\mathcal{L}$ corresponding to the second smallest eigenvalue.
- *Effective resistance* (*ERes*): The concept of ERes was introduced by [30] and for robustness improvement as a local pairwise approximation by [13]. This strategy selects vertices v, u that satisfy $\operatorname{argmax}_{v,u} \Omega_{v,u}$. $\Omega_{v,u}$ is defined as $(\hat{\mathcal{L}}^{-1})_{vv} + (\hat{\mathcal{L}}^{-1})_{uu} - 2(\hat{\mathcal{L}}^{-1})_{vu}$, where $\hat{\mathcal{L}}^{-1}$ is the pseudoinverse of $\mathcal{L}$.
- *Supervised learning* (*SL*): We consider an SL baseline by regressing on $\mathcal{F}$ to learn an approximate $\hat{\mathcal{F}}$. We use the same S2V architecture as RNet–DQN, which we train using MSE loss instead of the Q-learning loss. To select actions for a graph G, the agent considers all graphs G' that are one edge away, selecting the one that satisfies $\operatorname{argmax}_{G'} \hat{\mathcal{F}}$.

(c) Neural network architecture

For all experiments, we use an S2V embedding vector of length 64. The neural network architecture used for RNet–DQN and SL is formed of state–action embeddings obtained using S2V followed by a multi-layer perceptron; the single output unit corresponds to the $Q(s, a)$ estimate for RNet–DQN and the predicted $\hat{\mathcal{F}}$ for SL, respectively. Details of hyperparameters used for the two learning-based models are provided in the electronic supplementary material.

(d) Evaluation protocol

We evaluate RNet–DQN and baselines on both synthetic and real-world graphs. We allow agents a number of edge additions L equivalent to a percentage τ of total possible edges. As an evaluation metric, we report the cumulative reward obtained by the agents, which quantifies the improvement in the objective function value between the final and original graphs. Specifically, in the context of the robustness metrics used, the values provided measure the difference in the expected fraction of nodes that need to be removed for the network to become disconnected. Training is performed separately for each graph family, objective function $\mathcal{F}$ and

value of L. Where an agent is non-deterministic (either through intrinsic stochasticity or need for training), we repeat its evaluation (and training where applicable, starting from a different random initialization of the network weights) to compute confidence intervals. For the learned models, we record both average and maximum performance. No hyperparameter tuning is performed because of computational budget constraints. Details about the experimental settings are provided in the electronic supplementary material.

(e) Synthetic graphs

We consider graphs generated through the following models:

— *Erdös–Rényi* (*ER*): A graph sampled uniformly out of $\mathbf{G}^{(N,m)}$ [31]. We use $m = 20/100 * (N * (N-1))/2$, which represents 20% of all possible edges.
— *Barabási–Albert* (*BA*): A growth model where n nodes each attach preferentially to M existing nodes [1]. We use $M = 2$.

We consider graphs with $|V| = 20$, allowing agents to add a percentage of the total number of edges equal to $\tau \in \{1, 2, 5\}$, which yields $L \in \{2, 5, 10\}$. For RNet–DQN and SL, we train on a disjoint set of graphs $\mathbf{G}^{\text{train}}$. We periodically measure performance on another set $\mathbf{G}^{\text{validate}}$, storing the best model found. We use $|\mathbf{G}^{\text{train}}| = 10^4$ and $|\mathbf{G}^{\text{validate}}| = 10^2$. The performance of all agents is evaluated on a set $\mathbf{G}^{\text{test}}$ with $|\mathbf{G}^{\text{test}}| = 10^2$ generated using the ER and BA models. In order to evaluate out-of-distribution generalization, we repeat the evaluation on graphs with up to $|V| = 100$ (only up to $|V| = 50$ for greedy and SL due to computational cost; see next section) and scale m (for ER) and L accordingly. For non-deterministic agents, evaluation (and training, where applicable) is repeated across 50 random seeds.

(f) Real-world graphs

In order to evaluate our approach on real-world graphs, we consider infrastructure networks (for which robustness is a critical property) extracted from two datasets: *Euroroad* (road connections in mainland Europe and parts of West and Central Asia [32,33], $|V| = 1174$) and *Scigrid* (a dataset of the European power grid [34], $|V| = 1479$). We split these graphs by the country in which the nodes are located, selecting the largest connected component in case they are disconnected. We then select those with $20 \leq |V| \leq 50$, obtaining six infrastructure graphs for Scigrid and eight for Euroroad. The partitioning and selection procedure yields infrastructure graphs for the following countries:

— *Euroroad*: Finland, France, Kazakhstan, Poland, Romania, Russia, Turkey, Ukraine.
— *Scigrid*: Switzerland, Czech Republic, UK, Hungary, Ireland, Sweden.

Since, in this context, the performance on individual instances matters more than generalizability, we train and evaluate on each graph separately (effectively, the sets $\mathbf{G}^{\text{train}}$, $\mathbf{G}^{\text{validate}}$, $\mathbf{G}^{\text{test}}$ all have cardinality 1 and contain the same graph). *SL* is excluded from this experiment since we consider a single network. Evaluation is repeated across 10 random seeds.

5. Results

In table 1, we present the results of our experimental evaluation for synthetic graphs. We also display the evolution of the validation loss during training in figure 2. Out-of-distribution generalization results are shown in figure 3.

The results for real-world graphs are provided in table 2. Additionally, figure 4 displays examples of the original and improved topologies found by our approach.

Table 1. Mean cumulative reward per episode obtained by the agents on synthetic graphs with $|V| = 20$, grouped by objective function, graph family and number of edge additions L. Each reported value represents the improvement in the expected critical fraction between the final and initial graphs. Italic font indicates the best performing method (i.e. the highest value in the row).

								SL		RNet–DQN	
objective	G	L	random	LDP	FV	ERes	greedy	avg.	best	avg.	best
$\mathcal{F}_{\text{random}}$	BA	2	0.018 ± 0.001	0.036	0.051	0.053	0.033	0.048 ± 0.002	*0.057*	0.051 ± 0.001	*0.057*
		5	0.049 ± 0.002	0.089	0.098	0.106	0.079	0.099 ± 0.003	0.122	0.124 ± 0.001	*0.130*
		10	0.100 ± 0.003	0.158	0.176	0.180	0.141	0.161 ± 0.008	0.203	0.211 ± 0.001	*0.222*
	ER	2	0.029 ± 0.001	0.100	0.103	0.103	0.082	0.094 ± 0.001	0.100	0.098 ± 0.001	*0.104*
		5	0.071 ± 0.002	0.168	0.172	*0.175*	0.138	0.158 ± 0.002	0.168	0.164 ± 0.001	0.173
		10	0.138 ± 0.002	0.238	0.252	*0.253*	0.217	0.221 ± 0.005	0.238	0.240 ± 0.001	0.249
$\mathcal{F}_{\text{targeted}}$	BA	2	0.010 ± 0.001	0.022	0.018	0.018	0.045	0.022 ± 0.002	0.033	0.042 ± 0.001	*0.047*
		5	0.025 ± 0.001	0.091	0.037	0.077	0.077	0.055 ± 0.003	0.077	0.108 ± 0.001	*0.117*
		10	0.054 ± 0.003	0.246	0.148	0.232	0.116	0.128 ± 0.014	0.217	0.272 ± 0.002	*0.289*
	ER	2	0.020 ± 0.002	0.103	0.090	0.098	*0.149*	0.102 ± 0.002	0.118	0.122 ± 0.001	0.128
		5	0.050 ± 0.002	0.205	0.166	0.215	*0.293*	0.182 ± 0.008	0.238	0.268 ± 0.001	0.279
		10	0.098 ± 0.003	0.306	0.274	0.299	0.477	0.269 ± 0.016	0.374	0.461 ± 0.003	*0.482*

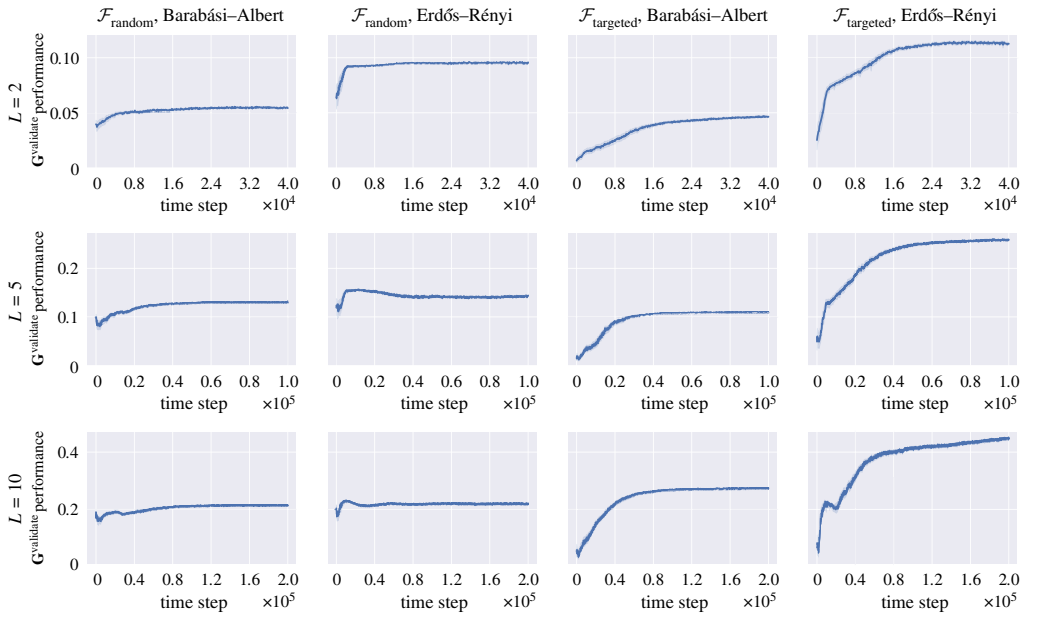

Figure 2. Performance on $\mathbf{G}^{\text{validate}}$ for synthetic graphs as a function of training steps. Note the different x-axis scales for each row: more training steps are typically required for longer edge addition sequences. (Online version in colour.)

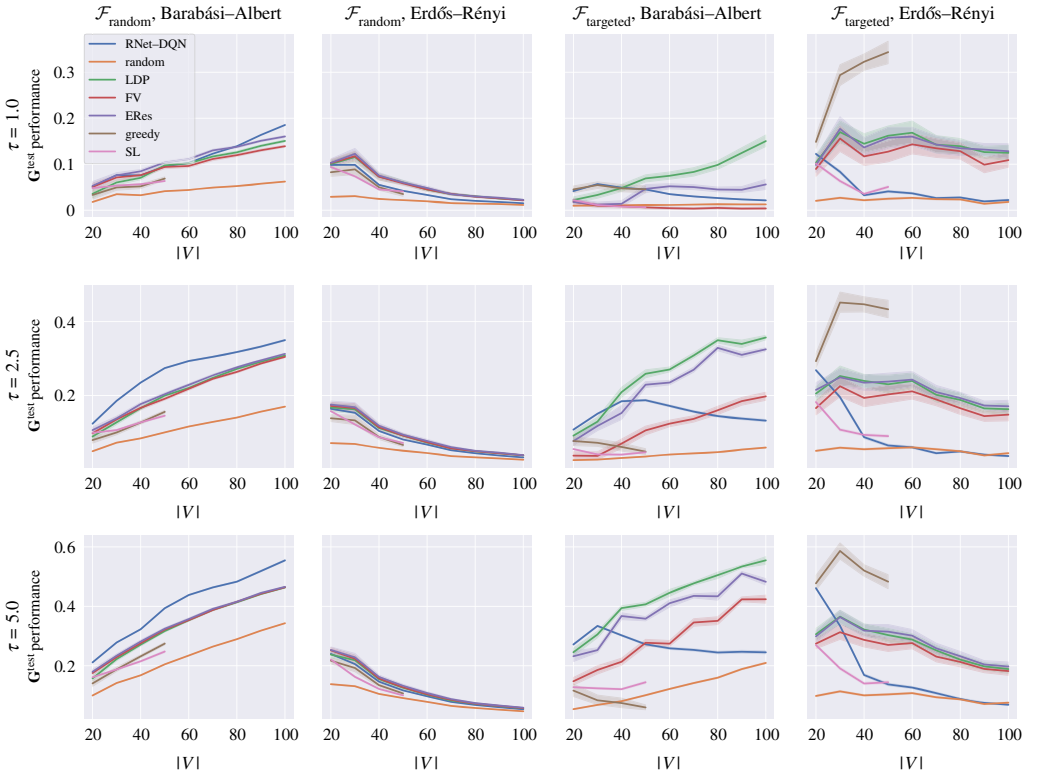

Figure 3. Performance on out-of-distribution synthetic graphs as a function of graph size, grouped by target problem and percentage of edge additions τ. For RNet–DQN and SL, models trained on graphs with $|V| = 20$ are used. (Online version in colour.)

Table 2. Results obtained on real-world graphs, split by graph instance. Each reported value represents the improvement in the expected critical fraction between the final and initial graphs. Italic font indicates the best performing method (i.e. the highest value in the row).

								RNet–DQN	
objective	dataset	instance	random	LDP	FV	ERes	greedy	avg.	best
$\mathcal{F}_{\text{random}}$	Euroroad	Finland	0.080 ± 0.019	0.133	0.163	0.170	0.162	0.161 ± 0.015	*0.189*
		France	0.057 ± 0.016	0.149	0.181	0.163	0.151	0.178 ± 0.013	*0.202*
		Kazakhstan	0.107 ± 0.018	0.165	0.191	0.180	0.160	0.179 ± 0.010	*0.203*
		Poland	0.082 ± 0.033	0.186	0.170	0.201	0.140	0.196 ± 0.014	*0.230*
		Romania	0.076 ± 0.025	0.196	0.170	*0.243*	0.203	0.207 ± 0.013	0.235
		Russia	0.084 ± 0.016	0.135	0.224	0.157	0.187	0.199 ± 0.017	*0.230*
		Turkey	0.092 ± 0.023	0.191	0.198	0.198	0.191	0.215 ± 0.011	*0.247*
		Ukraine	0.071 ± 0.017	0.158	0.186	0.151	0.098	0.163 ± 0.022	*0.205*
	Scigrid	Switzerland	0.050 ± 0.035	0.191	0.160	0.174	0.182	0.198 ± 0.017	*0.226*
		Czech Rep.	0.091 ± 0.020	0.242	0.239	0.252	0.214	0.334 ± 0.020	*0.375*
		UK	0.111 ± 0.020	0.263	0.273	0.290	0.224	0.321 ± 0.022	*0.379*
		Hungary	0.051 ± 0.029	0.176	0.179	0.175	0.117	0.148 ± 0.017	*0.185*
		Ireland	0.090 ± 0.014	0.208	0.211	0.213	0.177	0.201 ± 0.013	*0.228*
		Sweden	0.097 ± 0.029	0.187	0.213	0.195	0.197	0.213 ± 0.022	*0.276*
$\mathcal{F}_{\text{targeted}}$	Euroroad	Finland	0.069 ± 0.018	0.149	0.112	0.112	*0.307*	0.273 ± 0.009	0.300
		France	0.032 ± 0.019	0.199	0.120	0.120	0.074	0.218 ± 0.006	*0.228*
		Kazakhstan	0.052 ± 0.021	0.161	0.137	0.124	0.229	0.236 ± 0.014	*0.257*
		Poland	0.010 ± 0.008	0.101	0.114	0.084	0.108	0.230 ± 0.008	*0.248*
		Romania	0.029 ± 0.021	0.167	0.056	0.126	0.148	0.238 ± 0.021	*0.270*
		Russia	0.000 ± 0.000	0.000	0.000	0.053	0.000	0.110 ± 0.036	*0.155*
		Turkey	0.044 ± 0.021	0.126	0.155	0.126	0.143	0.233 ± 0.018	*0.264*
		Ukraine	0.031 ± 0.023	0.074	0.037	0.083	0.135	0.164 ± 0.006	*0.178*
	Scigrid	Switzerland	0.030 ± 0.024	0.000	0.103	0.098	0.045	0.128 ± 0.006	*0.139*
		Czech Rep.	0.038 ± 0.026	0.116	0.116	0.116	0.163	0.242 ± 0.027	*0.284*
		UK	0.070 ± 0.047	0.190	0.095	0.184	0.207	0.252 ± 0.027	*0.326*
		Hungary	0.027 ± 0.028	*0.190*	0.000	0.129	0.143	*0.190 ± 0.000*	*0.190*
		Ireland	0.047 ± 0.023	0.101	0.084	0.106	0.079	0.259 ± 0.011	*0.288*
		Sweden	0.061 ± 0.021	0.142	0.121	0.201	0.094	0.232 ± 0.008	*0.261*

(a) Main findings

We summarize our findings as follows:

RNet–DQN provides competitive performance, especially for longer action sequences. Across all settings tested, RNet–DQN performed significantly better than random. On synthetic graphs, the best model obtained the highest performance in eight out of 12 settings tested, while the average performance is at least 89% of that of the best-performing configuration. For BA graphs, RNet–DQN obtained the best performance across all tasks

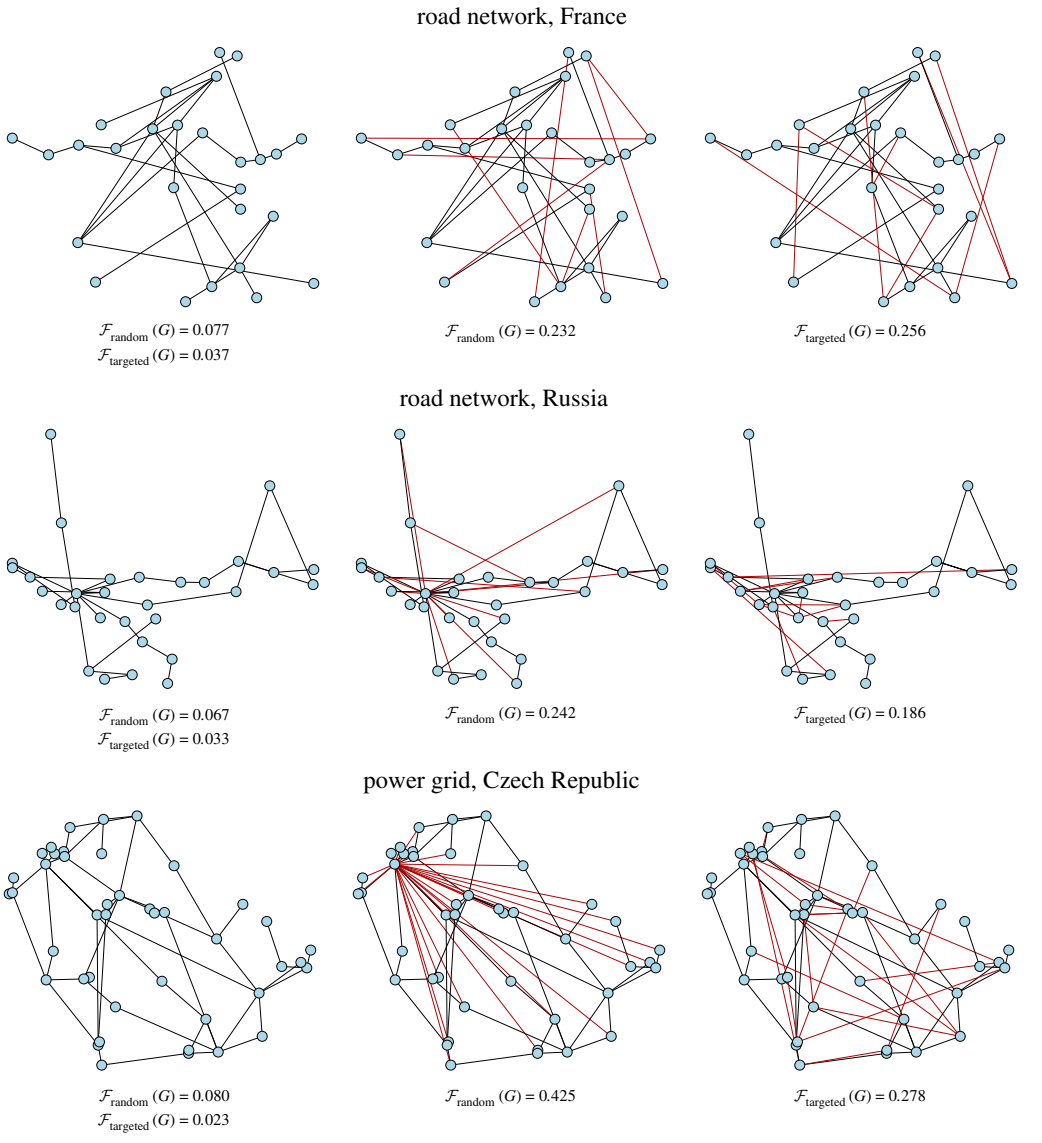

Figure 4. Several examples of the solutions found by RNet–DQN on real-world graphs. Each row of the illustration shows the original network on the left, while the central and right panels show the network optimized for resilience to random and targeted removals, respectively. Objective function values are shown underneath. The solutions for $\mathcal{F}_{\text{random}}$ typically assign more connections to a few central nodes, notably discovering the hub pattern in the third example. For $\mathcal{F}_{\text{targeted}}$, the added edges are spread around the network, reducing the impact of attacks. However, the algorithm might discover more complex patterns that are not directly interpretable, as shown in the solutions for the first example network. (Online version in colour.)

tested. For ER graphs, ERes performed slightly better when considering $\mathcal{F}_{\text{random}}$; for $\mathcal{F}_{\text{targeted}}$ the greedy baseline performed better for shorter sequences. For real-world graphs, RNet–DQN obtained the best performance across all tasks.

Strategies for improving $\mathcal{F}_{random}$ are easier to learn. The performance gap between the trained model and the baselines is smaller for $\mathcal{F}_{\text{random}}$, suggesting that it is less complex to learn. This is also supported by the evaluation losses monitored during training, which show that performance improves and plateaus more quickly. For $\mathcal{F}_{\text{random}}$, the network with randomly initialized parameters already yields policies with satisfactory results,

and training brings a small improvement. By contrast, the improvements for $\mathcal{F}_{\text{targeted}}$ are much more dramatic.

Out-of-distribution generalization only occurs for $\mathcal{F}_{random}$. The performance on larger out-of-distribution graphs is preserved for the $\mathcal{F}_{\text{random}}$ objective, and especially for BA graphs we observe strong generalization. The performance for $\mathcal{F}_{\text{targeted}}$ decays rapidly, obtaining worse performance than the baselines as the size increases. The poor performance of the greedy policy means that the $Q(s,a)$ estimates are no longer accurate under distribution shift. There are several possible explanations, e.g. the inherent noise of estimating $\mathcal{F}_{\text{random}}$ makes the neural network more robust to outliers or that central nodes impact message passing in larger graphs differently. We think investigating this phenomenon is a worthwhile future direction of this work, since out-of-distribution generalization does occur for $\mathcal{F}_{\text{random}}$ and evaluating the objective functions directly is prohibitively expensive for large graphs.

Performance on real-world graphs is comparatively better w.r.t. the baselines. This is expected since training is performed separately for each graph to be optimized.

(b) Time complexity

We also compare the time complexities of all approaches considered below.

— *RNet–DQN*: $O(|V|+|E|)$ operations at each step: constructing node and graph-level embeddings and, based on these embeddings, performing the forward pass in the neural network to estimate $Q(s,a)$ for all valid actions.
— *Random*: $O(1)$ for sampling, assuming the environment checks action validity.
— *Greedy*: $O(|V|^4 \times (|V|+|E|))$. The improvement in $\mathcal{F}$ is estimated for all $O(|V|^2)$ possible edges. For each edge, this involves $O(|V|)$ MC simulations. As described in §2, each MC simulation has complexity $O(|V| \times (|V|+|E|))$.
— *LDP*: $O(|V|^2)$, computing the product of node degrees.
— *FV, ERes*: $O(|V|^3)$, since they involve computing the eigendecomposition and the Moore–Penrose pseudoinverse of the graph Laplacian, respectively (may be faster in practice).
— *SL*: $O(|V|^2 \times (|V|+|E|))$. $\hat{\mathcal{F}}$ is predicted for $O(|V|^2)$ graphs that are one edge away, then an *argmax* is taken.

It is worth noting that the analysis above does not account for the cost of training, the complexity of which is difficult to determine as it depends on many hyperparameters and the specific characteristics of the problem at hand. The approach is thus advantageous in situations in which predictions need to be made quickly, over many graphs, or the model transfers well from a cheaper training regime. We also remark that, even though the method requires an upfront cost for training, this can be seen as a constant term if the number of problem instances over which we would like to obtain predictions is large. These characteristics are shared with other emergent work that tackles combinatorial optimization with machine learning [15,35].

6. Related work and discussion

(a) Network resilience

Network resilience was first quantified by the average shortest path distance as a function of the number of removed nodes [7]. Analysing two scale-free communication networks, the authors found that this type of network has good robustness to random failure but is vulnerable to targeted attacks. A more extensive investigation [36] analysed the robustness of several real-world networks as well as some generated by synthetic models using a variety of attack strategies. Another area of interest is the analysis of the phase transitions of the graph in terms of connectivity under the two attack strategies [6,9]. Optimal network topologies have also been

discovered—for example, under the objective of resilience to both failures and attacks, the optimal network has a bi-modal or tri-modal degree distribution [37,38]. There exists evidence to suggest that the topological robustness of infrastructure systems is correlated with operational robustness [39]. More broadly, the resilience of systems is highly important in structural engineering and risk management [40,41].

(b) Graph neural networks and combinatorial optimization

Neural network architectures able to deal not solely with Euclidean but also with manifold and graph data have been developed in recent years [42], and applied to a variety of problems where their capacity for representing structured and relational information can be exploited [43]. A sub-category of such approaches are message-passing neural networks (MPNNs) [16], often referred to as GNNs instead. Significant progress has been achieved in machine learning for combinatorial optimization problems [35], such as minimum vertex cover and the travelling salesman problem by framing them as an SL [44] or RL [14] task. Combining GNNs with RL algorithms has yielded models capable of solving several graph optimization problems with the same architecture while generalizing to graphs an order of magnitude larger than those used during training [15]. The problem as formulated in this paper is a combinatorial optimization problem, related to problems in network design that arise in operations research [45]—albeit with a different objective.

(c) Graph generation

Similarities exist between the current work and the area of graph generative modelling, which tries to learn a generative model of graphs [46–48] in a computationally efficient way given some existing examples. This generation, however, is not necessarily conditioned by an objective that captures a global property to be optimized. Other lines of work target constructing, *ab initio*, graphs that have desirable properties: examples include neural architecture search [49,50] and molecular graph generation [19,51,52]. The concurrent work GraphOpt [53] tackles the *inverse problem*: given a graph, the goal is to learn the underlying objective function that leads to its generation. This is achieved with maximum entropy inverse RL and GNNs.

(d) Relationship to RL–S2V

Our work builds on RL–S2V, a method that was applied for the construction of adversarial examples against graph classifiers [18]. However, it is worth noting that there are a series of key differences with respect to that approach. First, RL–S2V is not designed to address the problem of constructing robust graphs or, more generally, learning to construct graphs according to a given goal. Secondly, there are two key algorithmic differences from RL–S2V with respect to the GC-MDP formulation: the reward function used, which in this case quantifies a global property of the graph itself, as well as the definition of the action spaces and the transition model, which account for excluding already-existing edges (RL–S2V ignores this, leading to some of the edge budget being wasted). Since the values of structural properties we consider are increasing in the number of edges (the complete graph has robustness 1), RNet–DQN generally yields strictly better performance results.

7. Conclusion and outlook

In this work, we have addressed the problem of improving a graph structure given the goal of maximizing the value of a global objective function. We have framed it for the first time as a decision-making problem and we have formalized it as the graph construction MDP (GC-MDP). Our approach, named *RNet–DQN*, uses reinforcement learning and graph neural networks as key components for generalization. As a case study, we have considered the problem of improving graph robustness to random and targeted removals of nodes. Our experimental evaluation on

synthetic and real-world graphs shows that, in certain situations, this approach can deliver performance superior to existing methods, in terms of both the solutions found (i.e. the resulting robustness of the graphs) and the time complexity of model evaluation. Furthermore, we have shown the ability to transfer to out-of-sample graphs, as well as the potential to transfer to out-of-distribution graphs larger than those used during training.

(a) Extensions

The proposed approach can be applied to other problems based on different definitions of robustness or considering fundamentally different objective functions such as efficiency [20], path diversity [54] and assortativity [55], which are of interest in various biological, communication and social networks. Since our formulation and algorithm are objective-agnostic, we expect that they are applicable out-of-the-box for other objectives, even those for which no strong baselines are currently known. As such, this approach may be a useful tool for the discovery of new graph improvement algorithms for objectives that can be evaluated programatically, either in closed form or via simulations. Potential limitations might be related to the complexity of the objective functions and the related computational demands. We also view the interpretability of the approach, which is less straightforward than those based on known mathematical concepts such as the FV, to be an important research direction. Since there is an active interest in the interpretability of both GNNs and RL [56,57], we consider that there is scope for developing techniques that are tailor-made for explaining policies learned by RL on graphs.

(b) Operationalization

Beyond considering other objectives, in order to operationalize the proposed algorithm, it is possible to integrate a variety of refinements, which can include capturing heterogeneous edge costs (e.g. different capacities per link in a communication network), extending the action space to support heterogeneous edge types (e.g. addition of different types of edges with specific characteristics) and integrating domain-specific link constraints (e.g. planarity). For critical scenarios, it is also possible to verify that the resulting solutions satisfy some given formal properties and constraints [58]. Furthermore, considering multi-criteria objective functions [59] is important for cases where properties of the solutions must be balanced [60]. Various choices exist for representing this trade-off and should be captured on a case-by-case basis depending on the application: for example, a linear combination may be sufficient in certain situations, while others are characterized by economies of scale. We also remark that our formulation captures operational scenarios in which the cost of constructing a link is significantly greater than the cost of its maintenance (e.g. as with road networks). Our method can also be adapted for situations in which operational cost is instead greater by formulating a rewiring operation that keeps the number of links constant.

(c) Broader applications

The approach described in this work can be used to improve the properties of a variety of human-made infrastructure systems, such as communication networks, transportation networks and power grids. We also envisage potential applications in biological (e.g. hypothesis testing for understanding the characteristics of brain networks [61]), ecological (e.g. design of more resilient ecosystems [62]) and social (e.g. design of organizational structures [63]) networks. As far as biological networks are concerned, the brain is hypothesized to optimize a trade-off between efficiency and wiring cost [61]. Our method could be used to test different hypotheses related to the resulting structure of brain networks over evolutionary times and also during their development. Indeed, the evolution of such networks in time has been captured (for example, Sulston *et al.* [64] mapped the development of the *Caenorhabditis elegans* connectome). This can be achieved by applying the optimization procedure for different objective functions and comparing

the obtained networks with the 'ground truth'. With respect to the potential application in ecosystem management, the graph formalization can be used to model interactions between species in a given environment. As such, our method has potential applications to study, in simulation, the impact of introducing or removing species from an ecosystem so as to achieve a desired outcome. Specifically, in the context of robustness, we can consider optimizing the resilience of an ecosystem to intrinsic or extrinsic shocks, a task of fundamental importance [65]. The dynamics of interactions between species may be modelled in simulation using well-known models of, for example, predator–prey mechanics [66]. With respect to social networks, for example, our method can be applied to derive optimal communication strategies and related team structures so as to optimize a given objective for an organization. Finally, there are also potential applications for networks of artificial agents (i.e. robots). There is a significant body of work in the robotics literature that treats the problem of maintaining robust communication in a network of agents working together to complete a task in an environment that contains obstacles or adversaries. For instance, Stump *et al.* [67] use properties of the graph Laplacian (namely, the FV and its associated eigenvalue) to ensure that the underlying communication network remains robust. Since we have empirically shown superior performance to using the FV, our approach could also lead to gains in this deployment scenario.

Data accessibility. The original real-world datasets used in this research (Scigrid, Euroroad) are publicly available and were retrieved via the Scigrid project website www.power.scigrid.de/pages/downloads.html and KONECT http://konect.cc/, respectively. They can be downloaded from their respective portals without registration. Scigrid is licensed under the Open Database License (ODbL) v1.0, while Euroroad is license-free. The scripts and instructions used to extract the subgraphs corresponding to individual countries in the infrastructure networks are available in the code repository, which is part of the electronic supplementary material.

Authors' contributions. V.-A.D., S.H. and M.M. designed and developed the study, reviewed the results and wrote the manuscript. V.-A.D. wrote the implementation and performed the data analysis. All authors gave final approval for publication and agree to be held accountable for the work performed herein.

Competing interests. The authors declare no competing interests with respect to this work.

Funding. This work was supported by The Alan Turing Institute under the UK EPSRC grant no. EP/N510129/1.

References

1. Barabási A-L, Albert R. 1999 Emergence of scaling in random networks. *Science* **286**, 509–512. (doi:10.1126/science.286.5439.509)
2. Watts DJ, Strogatz SH. 1998 Collective dynamics of 'small-world' networks. *Nature* **393**, 440. (doi:10.1038/30918)
3. Bianchini M, Gori M, Scarselli F. 2005 Inside PageRank. *ACM Trans. Internet Technol.* **5**, 92–128. (doi:10.1145/1052934.1052938)
4. Newman MEJ. 2018 *Networks*. Oxford, UK: Oxford University Press.
5. Newman MEJ. 2003 The structure and function of complex networks. *SIAM Rev.* **45**, 167–256. (doi:10.1137/S003614450342480)
6. Cohen R, Erez K, Havlin S. 2000 Resilience of the internet to random breakdowns. *Phys. Rev. Lett.* **85**, 4626–4628. (doi:10.1103/PhysRevLett.85.4626)
7. Albert R, Jeong H, Barabási A-L. 2000 Error and attack tolerance of complex networks. *Nature* **406**, 378–382. (doi:10.1038/35019019)
8. Beygelzimer A, Grinstein G, Linsker R, Rish I. 2005 Improving network robustness by edge modification. *Physica A* **357**, 593–612. (doi:10.1016/j.physa.2005.03.040)
9. Cohen R, Erez K, Havlin S. 2001 Breakdown of the internet under intentional attack. *Phys. Rev. Lett.* **86**, 3682–3685. (doi:10.1103/PhysRevLett.86.3682)
10. Cetinay H, Devriendt K, Van Mieghem P. 2018 Nodal vulnerability to targeted attacks in power grids. *Appl. Netw. Sci.* **3**, 34. (doi:10.1007/s41109-018-0089-9)
11. Schneider CM, Moreira AA, Andrade JS, Havlin S, Herrmann HJ. 2011 Mitigation of malicious attacks on networks. *Proc. Natl Acad. Sci. USA* **108**, 3838–3841. (doi:10.1073/pnas.1009440108)
12. Wang H, Van Mieghem P. 2008 Algebraic connectivity optimization via link addition. In *Proc. of the Third Int. Conf. on Bio-Inspired Models of Network Information and Computing Systems, BIONETICS08, Hyogo Japan, 25–28 November 2008*. Brussels, Belgium: ICST.

13. Wang X, Pournaras E, Kooij RE, Van Mieghem P. 2014 Improving robustness of complex networks via the effective graph resistance. *Eur. Phys. J. B* **87**, 221. (doi:10.1140/epjb/e2014-50276-0)
14. Bello I, Pham H, Le QV, Norouzi M, Bengio S. 2016 Neural combinatorial optimization with reinforcement learning. (http://arxiv.org/abs/1611.09940)
15. Khalil E, Dai H, Zhang Y, Dilkina B, Song L. 2017 Learning combinatorial optimization algorithms over graphs. In *Proc. of the 31st Conf. on Neural Information Processing Systems (NeurIPS 2017), Long Beach, CA, USA, 4–9 December 2017*. Red Hook, NY: Curran Associates.
16. Gilmer J, Schoenholz SS, Riley PF, Vinyals O, Dahl GE. 2017 Neural message passing for quantum chemistry. In *Proc. of the 34th Int. Conf. on Machine Learning (ICML 2017), Sydney, Australia, 6–11 August 2017*. PMLR.
17. Mnih V *et al.* 2015 Human-level control through deep reinforcement learning. *Nature* **518**, 529–533. (doi:10.1038/nature14236)
18. Dai H, Li H, Tian T, Huang X, Wang L, Zhu J, Song L. 2018 Adversarial attack on graph structured data. In *Proc. of the 35th Int. Conf. on Machine Learning (ICML 2018), Stockholm, Sweden, 10–15 July 2018*. PMLR.
19. You J, Liu B, Ying R, Pande V, Leskovec J. 2018 Graph convolutional policy network for goal-directed molecular graph generation. In *Proc. of the 32nd Conf. on Neural Information Processing Systems (NeurIPS 2018), Montreal, Canada, 2–8 December 2018*. Red Hook, NY: Curran Associates.
20. Latora V, Marchiori M. 2001 Efficient behavior of small-world networks. *Phys. Rev. Lett.* **87**, 198701. (doi:10.1103/PhysRevLett.87.198701)
21. Estrada E, Hatano N. 2008 Communicability in complex networks. *Phys. Rev. E* **77**, 036111. (doi:10.1103/PhysRevE.77.036111)
22. OEIS Foundation. 2020 The on-line encyclopedia of integer sequences. See https://oeis.org/A001187.
23. Dai H, Dai B, Song L. 2016 Discriminative embeddings of latent variable models for structured data. In *Proc. of the 33rd Int. Conf. on Machine Learning (ICML 2016), New York City, NY, 19–24 June 2016*. PMLR.
24. Hessel M *et al.* 2018 Rainbow: combining improvements in deep reinforcement learning. In *Proc. of the 32nd AAAI Conf. on Artificial Intelligence (AAAI-18), New Orleans, LA, 2–7 February 2018*. Palo Alto, CA: AAAI Press.
25. Maron H, Ben-Hamu H, Serviansky H, Lipman Y. 2019 Provably powerful graph networks. In *Proc. of the 33rd Conf. on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada, 8–14 December 2019*. Red Hook, NY: Curran Associates.
26. Watkins CJCH, Dayan P. 1992 Q-learning. *Machine Learn.* **8**, 279–292.
27. Lillicrap TP *et al.* 2016 Continuous control with deep reinforcement learning. In *Proc. of the 6th Int. Conf. on Learning Representations (ICLR 2018), Vancouver, Canada, 30 April–3 May 2018*. OpenReview.
28. Riedmiller M. 2005 Neural fitted Q iteration – first experiences with a data efficient neural reinforcement learning method. In *Proc. of the 16th European Conf. on Machine Learning (ECML 2005), Porto, Portugal, 3–7 October 2005*. Berlin, Germany: Springer-Verlag.
29. Fiedler M. 1973 Algebraic connectivity of graphs. *Czechoslovak Math. J.* **23**, 298–305. (doi:10.21136/CMJ.1973.101168)
30. Ellens W, Spieksma F, Van Mieghem P, Jamakovic A, Kooij R. 2011 Effective graph resistance. *Linear Algebra Appl.* **435**, 2491–2506. (doi:10.1016/j.laa.2011.02.024)
31. Erdös P, Rényi A. 1960 On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci.* **5**, 17–60.
32. Kunegis J. 2013 KONECT: the Koblenz network collection. In *Proc. of the 22nd Int. World Wide Web Conf. (WWW'13) Companion, Rio de Janeiro, Brazil, 13–17 May 2013*. New York, NY: Association for Computing Machinery.
33. Šubelj L, Bajec M. 2011 Robust network community detection using balanced propagation. *Eur. Phys. J. B* **81**, 353–362. (doi:10.1140/epjb/e2011-10979-2)
34. Medjroubi W, Müller UP, Scharf M, Matke C, Kleinhans D. 2017 Open data in power grid modelling: new approaches towards transparent grid models. *Energy Rep.* **3**, 14–21. (doi:10.1016/j.egyr.2016.12.001)
35. Bengio Y, Lodi A, Prouvost A. 2021 Machine learning for combinatorial optimization: a methodological tour d'horizon. *Eur. J. Oper. Res.* **290**, 405–421. (doi:10.1016/j.ejor.2020.07.063)

36. Holme P, Kim BJ, Yoon CN, Han SK. 2002 Attack vulnerability of complex networks. *Phys. Rev. E* **65**, 056109. (doi:10.1103/PhysRevE.65.056109)
37. Tanizawa T, Paul G, Cohen R, Havlin S, Stanley HE. 2005 Optimization of network robustness to waves of targeted and random attacks. *Phys. Rev. E* **71**, 047101. (doi:10.1103/PhysRevE.71.047101)
38. Valente AXCN, Sarkar A, Stone HA. 2004 Two-peak and three-peak optimal complex networks. *Phys. Rev. Lett.* **92**, 118702. (doi:10.1103/PhysRevLett.92.118702)
39. Solé RV, Rosas-Casals M, Corominas-Murtra B, Valverde S. 2008 Robustness of the European power grids under intentional attack. *Phys. Rev. E* **77**, 026102.
40. Cimellaro GP, Reinhorn AM, Bruneau M. 2010 Framework for analytical quantification of disaster resilience. *Eng. Struct.* **32**, 3639–3649. (doi:10.1016/j.engstruct.2010.08.008)
41. Ganin AA, Massaro E, Gutfraind A, Steen N, Keisler JM, Kott A, Mangoubi R, Linkov I. 2016 Operational resilience: concepts, design and analysis. *Sci. Rep.* **6**, 1–12. (doi:10.1038/srep19540)
42. Bronstein MM, Bruna J, LeCun Y, Szlam A, Vandergheynst P. 2017 Geometric deep learning: going beyond Euclidean data. *IEEE Signal Process Mag.* **34**, 18–42. (doi:10.1109/MSP.2017.2693418)
43. Battaglia PW *et al.* 2018 Relational inductive biases, deep learning, and graph networks. (http://arxiv.org/abs/1806.01261).
44. Vinyals O, Fortunato M, Jaitly N. 2015 Pointer networks. In *Proc. of the 29th Conf. on Neural Information Processing Systems (NeurIPS 2015), Montréal, Canada, 7–12 December 2015*. Red Hook, NY: Curran Associates.
45. Ahuja RK, Magnanti TL, Orlin JB, Reddy MR. 1995 Applications of network optimization. In *Handbooks in operations research and management science* (eds MO Ball *et al.*), pp. 1–83. Network Models, vol. 7. Amsterdam, The Netherlands: Elsevier.
46. Li Y, Vinyals O, Dyer C, Pascanu R, Battaglia P. 2018 Learning deep generative models of graphs. In *Proc. of the 6th Int. Conf. on Learning Representations (ICLR 2018) Workshop Track, Vancouver, Canada, 30 April–3 May 2018*. OpenReview.
47. Liao R, Li Y, Song Y, Wang S, Nash C, Hamilton WL, Duvenaud D, Urtasun R, Zemel RS. 2019 Efficient graph generation with graph recurrent attention networks. In *Proc. of the 33rd Conf. on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada, 8–14 December 2019*. Red Hook, NY: Curran Associates.
48. You J, Ying R, Ren X, Hamilton WL, Leskovec J. 2018 GraphRNN: generating realistic graphs with deep auto-regressive models. In *Proc. of the 35th Int. Conf. on Machine Learning (ICML 2018), Stockholm, Sweden, 10–15 July 2018*. PMLR.
49. Liu H, Simonyan K, Vinyals O, Fernando C, Kavukcuoglu K. 2018 Hierarchical representations for efficient architecture search. In *Proc. of the 6th Int. Conf. on Learning Representations (ICLR 2018), Vancouver, Canada, 30 April–3 May 2018*. OpenReview.
50. Zoph B, Le QV. 2017 Neural architecture search with reinforcement learning. In *Proc. of the 5th Int. Conf. on Learning Representations (ICLR 2017), Toulon, France, 24–26 April 2017*. OpenReview.
51. Bradshaw J, Paige B, Kusner MJ, Segler MHS, Hernández-Lobato JM. 2019 A model to search for synthesizable molecules. In *Proc. of the 33rd Conf. on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada, 8–14 December 2019*. Red Hook, NY: Curran Associates.
52. Jin W, Barzilay R, Jaakkola T. 2018 Junction tree variational autoencoder for molecular graph generation. In *Proc. of the 35th Int. Conf. on Machine Learning (ICML 2018), Stockholm, Sweden, 10–15 July 2018*. PMLR.
53. Trivedi R, Yang J, Zha H. 2020 GraphOpt: learning optimization models of graph formation. In *Proc. of the 37th Int. Conf. on Machine Learning (ICML 2020), Online, 12–18 July 2020*. PMLR.
54. Gvozdiev N, Vissicchio S, Karp B, Handley M. 2018 On low-latency-capable topologies, and their impact on the design of intra-domain routing. In *Proc. of the 2018 Conf. of the ACM Special Interest Group on Data Communication (SIGCOMM), Budapest, Hungary, 20–25 August 2018*. New York, NY: Association for Computing Machinery.
55. Newman MEJ. 2002 Assortative mixing in networks. *Phys. Rev. Lett.* **89**, 208701. (doi:10.1103/PhysRevLett.89.208701)
56. Verma A, Murali V, Singh R, Kohli P, Chaudhuri S. 2018 Programmatically interpretable reinforcement learning. In *Proc. of the 35th Int. Conf. on Machine Learning (ICML 2018), Stockholm, Sweden, 10–15 July 2018*. PMLR.

57. Ying R, Bourgeois D, You J, Zitnik M, Leskovec J. 2019 Gnnexplainer: Generating explanations for graph neural networks. In *Proc. of the 33rd Conf. on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada, 8–14 December 2019*. Red Hook, NY: Curran Associates.
58. Garcıa J, Fernández F. 2015 A comprehensive survey on safe reinforcement learning. *J. Mach. Learn. Res.* **16**, 44.
59. Roijers DM, Vamplew P, Whiteson S, Dazeley R. 2013 A survey of multi-objective sequential decision-making. *J. Artif. Intell. Res.* **48**, 67–113. (doi:10.1613/jair.3987)
60. Keeney RL, Raiffa H, Meyer RF. 1993 *Decisions with multiple objectives: preferences and value trade-offs*. Cambridge, UK: Cambridge University Press.
61. Bullmore E, Sporns O. 2012 The economy of brain network organization. *Nat. Rev. Neurosci.* **13**, 336–349. (doi:10.1038/nrn3214)
62. Westman WE. 1978 Measuring the inertia and resilience of ecosystems. *BioScience* **28**, 705–710. (doi:10.2307/1307321)
63. Wreathall J. 2017 Properties of resilient organizations: an initial view. In *Resilience engineering* (eds E Hollnagel, DD Woods, N Leveson), pp. 275–285. New York, NY: CRC Press.
64. Sulston JE, Schierenberg E, White JG, Thomson JN. 1983 The embryonic cell lineage of the nematode *Caenorhabditis elegans*. *Dev. Biol.* **100**, 64–119. (doi:10.1016/0012-1606(83)90201-4)
65. Allesina S, Bodini A, Pascual M. 2009 Functional links and robustness in food webs. *Phil. Trans. R. Soc. B* **364**, 1701–1709. (doi:10.1098/rstb.2008.0214)
66. Berryman AA. 1992 The origins and evolution of predator-prey theory. *Ecology* **73**, 1530–1535. (doi:10.2307/1940005)
67. Stump E, Jadbabaie A, Kumar V. 2008 Connectivity management in mobile robot teams. In *Proc. of the 2008 IEEE Int. Conf. on Robotics and Automation, Pasadena, CA, 19–23 May 2008*, pp. 1525–1530. New York, NY: IEEE.