PROCEEDINGS A

royalsocietypublishing.org/journal/rspa

Research

**Cite this article:** Darvariu V-A, Hailes S, Musolesi M. 2023 Planning spatial networks with Monte Carlo tree search. *Proc. R. Soc. A* **479**: 20220383.
https://doi.org/10.1098/rspa.2022.0383

Received: 3 June 2022
Accepted: 5 December 2022

**Subject Areas:**
artificial intelligence

**Keywords:**
planning, spatial networks, Monte Carlo tree search

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



# Planning spatial networks with Monte Carlo tree search

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

[1]University College London, London, UK
[2]The Alan Turing Institute, London, UK
[3]University of Bologna, Bologna, Italy

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

We tackle the problem of goal-directed graph construction: given a starting graph, finding a set of edges whose addition maximally improves a global objective function. This problem emerges in many transportation and infrastructure networks that are of critical importance to society. We identify two significant shortcomings of present reinforcement learning methods: their exclusive focus on topology to the detriment of spatial characteristics (which are known to influence the growth and density of links), as well as the rapid growth in the action spaces and costs of model training. Our formulation as a deterministic Markov decision process allows us to adopt the Monte Carlo tree search framework, an artificial intelligence decision-time planning method. We propose improvements over the standard upper confidence bounds for trees (UCT) algorithm for this family of problems that addresses their single-agent nature, the trade-off between the cost of edges and their contribution to the objective, and an action space linear in the number of nodes. Our approach yields substantial improvements over UCT for increasing the efficiency and attack resilience of synthetic networks and real-world Internet backbone and metro systems, while using a wall clock time budget similar to other search-based algorithms. We also demonstrate that our approach scales to significantly larger networks than previous reinforcement learning methods, since it does not require training a model.

Electronic supplementary material is available online at https://doi.org/10.6084/m9.figshare.c.6350517.



THE ROYAL SOCIETY PUBLISHING

## 1. Introduction

Graphs are a pervasive representation that arise naturally in a variety of disciplines; however, their non-Euclidean structure has traditionally proven challenging for machine learning and decision-making approaches.

The emergence of the graph neural network learning paradigm [1] and geometric deep learning more broadly [2] has brought about encouraging breakthroughs in diverse application areas for graph-structured data: relevant examples include combinatorial optimization [3–5], recommendation systems [6,7] and computational chemistry [8–11].

There is an increasing interest in the problem of goal-directed graph construction, in which the aim is to build or to modify the topology of a graph (i.e. add a set of edges) so as to maximize the value of a global objective function, subject to a budget constraint. As an example scenario, consider the following: given a road network and a budget of 200 km of highways that can be invested, a national transportation department must decide where to place them with the goal of minimizing average trip time for drivers. Another practical instance of this problem is to place 500 km of tracks between stations in a train network such that, in the event of infrastructure failures or disruptions, customers have many alternative journeys that they can make. Similar network planning and design scenarios arise in a variety of other infrastructure systems including communication, power, and water distribution networks. The family of related problems unified by this framework is illustrated at a high level in the first panel of figure 1 and is mathematically defined in equation (3.1).

As this task involves an element of exploration (optimal solutions are not known *a priori*), its formulation as a decision-making process is a suitable paradigm. The authors of [12] formulated the optimization of a global structural graph property as a Markov decision process (MDP) and approached it using a variant of the RL-S2V [13] reinforcement learning algorithm, showing that generalizable strategies for improving a global network objective can be learned, and can obtain performance superior to prior conventional approaches [14–17] for certain classes of problems.

However, several real-world networks are embedded in space and this fact adds constraints and trade-offs in terms of the topologies that can be created [18,19]. In fact, since there is a cost associated with edge length, connections tend to be local and long-range connections must be justified by some gain (e.g. providing connectivity to a hub). Moreover, existing methods based on reinforcement learning are challenging to scale, due to the sample complexity of current training algorithms, the linear increase of possible actions with the number of nodes, and the complexity of evaluating the global objectives (typically polynomial in the number of nodes). Furthermore, objective functions defined over nodes' positions, such as efficiency, are key for understanding their organization [20], but existing methods only consider topological aspects of the problem. Additionally, training data (i.e. instances of real-world graphs) are scarce and we are typically interested in a specific starting graph (e.g. an infrastructure network to be improved).

In this paper, for the first time, we consider the problem of the construction of spatial graphs as a decision-making process that explicitly captures the influence of space on graph-level objectives, the realisability of links, and connection budgets. Furthermore, to address the scalability issue, we select an *optimal set of edges* to add to the graph through planning, which sidesteps the problem of sample complexity since we do not need to learn a generalizable strategy. We adopt the Monte Carlo tree search (MCTS) framework—specifically, the upper confidence bounds for trees (UCT) algorithm [21]—and show it can successfully be applied to the derivation of graph construction strategies. We illustrate our approach in figures 1 and 2. Finally, we propose several improvements over the basic UCT method in the context of spatial networks. These relate to important characteristics of this family of problems: namely, their single-agent, deterministic nature; the inherent trade-off between the cost of edges and their contribution to the global objective; and an action space that is linear in the number of nodes in the network. Our proposed approach, spatial graph UCT (SG-UCT), is designed with these characteristics in mind and relies

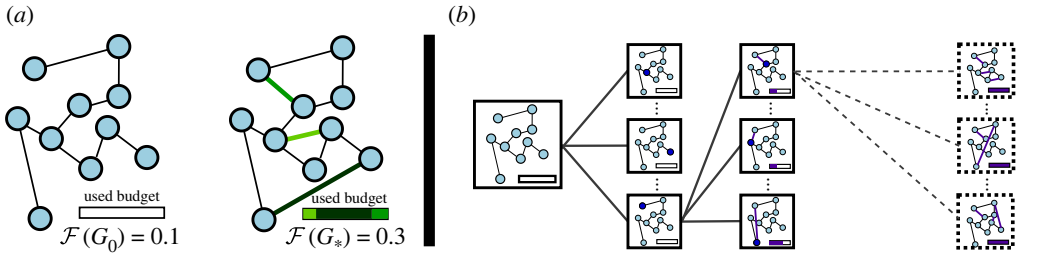


**Figure 1.** Schematic of our approach. (*a*) Given a spatial graph $G_0$, an objective function $\mathcal{F}$, and a budget defined in terms of edge lengths, the goal is to add a set of edges such that the resulting graph $G_*$ maximally increases $\mathcal{F}$. (*b*) We formulate this problem as a deterministic MDP, in which states are graphs, actions represent the selection of a node, transitions add an edge every two steps, and the reward is based on $\mathcal{F}$. We use Monte Carlo tree search to plan *the optimal set of edges to be added* using knowledge of this MDP, and propose a method (SG-UCT) that improves on standard UCT.

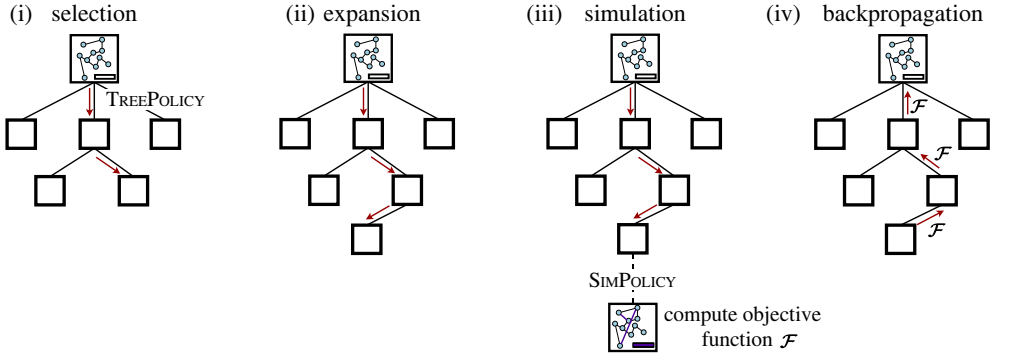


**Figure 2.** Illustration of Monte Carlo tree search applied to the construction of spatial networks. (i) Starting from the root node, which contains the original graph $S_0$, the tree is traversed until an expandable node is reached. (ii) The algorithm expands the tree by adding a child to this node. The strategy for traversing the tree and deciding which node to expand is called *tree policy*, and typically relies on statistics (such as average reward) stored by each node. (iii) From the newly added node, a trajectory is sampled using a *simulation policy* until a terminal state is reached and the algorithm cannot add more edges to the graph. A reward is received depending on the objective function $\mathcal{F}$ defined on the graph. (iv) The path is traversed back to the root node, updating the statistics of nodes along the way. After several iterations of steps (i)–(iv), the algorithm will select action $A_0$ corresponding to the child of the root with the highest average reward. The search then continues with the next state $S_1$ at the root.

on a limited set of assumptions. For this reason, it can be applied to a large class of spatial networked systems.

As objective functions, we consider the global properties of network efficiency and robustness to targeted attacks. While these represent a variety of practical scenarios, our approach is broadly applicable to any other structural property. We perform an evaluation on synthetic graphs generated by a spatial growth model and several real-world internet backbone networks and metro transportation systems. Our results show that SG-UCT performs best out of all methods that have been proposed in the past in all the settings we tested; moreover, the performance gain over UCT is substantial (24% on average and up to 54% over UCT on the largest networks tested in terms of a robustness metric). In addition, we conduct an ablation study that explores the impact of the single algorithmic mechanisms, highlighting that the most significant part of the objective function gains is due to the simulation policy that prioritizes lower cost edges. We also

benchmark the execution time of the various approaches, showing that SG-UCT requires similar amounts of computation to other search-based methods.

## 2. Preliminaries and background

### (a) Markov decision processes

MDPs are widely adopted formalizations of decision-making tasks. The decision maker, usually 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 actions, 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* $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_T$, which continues until a terminal state $S_T$ is reached. In *deterministic* MDPs, there exists a unique state $s'$ s.t. $P(S_{t+1} = s' | S_t = s, A_t = a) = 1$. The tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ defines this MDP, where $\gamma \in [0,1]$ weighs immediate and long-term rewards. The discounted sum of rewards $\sum_{k=0}^{T} \gamma^k R_{t+k+1}$ received from state $t$ onwards is called *return*. We also define a *policy* $\pi(a|s)$, a distribution of actions over states. Given a policy $\pi$, the *value function* $V_\pi(s)$ is defined as the expected return when starting from $s$ and following policy $\pi$.

There exists a spectrum of algorithms for constructing a policy, ranging from model-based algorithms (which assume knowledge of the MDP) to model-free algorithms (which require only samples of agent-environment interactions). If either the full MDP specification or a model is available, a *planning* method such as forward search [22] can be used. Dynamic programming, which is also a model-based method, has the same theoretical underpinning; its goal is to find the optimal value functions that satisfy the Bellman optimality equations. However, since it involves exploring the entire state space, it is very computationally intensive and not applicable to large MDPs. Reinforcement learning methods can be viewed as solving the same equations approximately by sampling, requiring less computation. Beyond this, decision-time planning methods aim to reduce computation even further by considering trajectories that only start at the current state in a problem of interest, rather than attempting to approximate the optimality equations for a larger part of the state space [23].

### (b) Monte Carlo tree search

MCTS is a model-based planning technique that addresses the inability to explore all paths in large MDPs by constructing a policy *from the current state*. It builds a search tree rooted at the current state, in which each child node represents the next state obtained by applying an action to the state at the parent node. It relies on two core principles: firstly, that the value of a state can be estimated by sampling trajectories and, secondly, that the returns obtained by this sampling are informative for deciding the next action at the root of the search tree. We review its basic concepts below and refer the interested reader to [24] for more information. To aid understanding, in figure 2, we illustrate its main steps and their relationship to the goal-directed graph construction of spatial networks.

In MCTS, each node in the search tree stores several statistics such as the sum of returns and the node visit count in addition to the state. For deciding each action, the search task is given a computational budget expressed in terms of node expansions or wall clock time. The algorithm keeps executing the following sequence of steps until the search budget is exhausted:

(i) **Selection**: The tree is traversed iteratively from the root until an expandable node (i.e. a node containing a non-terminal state with yet-unexplored actions) is reached.
(ii) **Expansion**: From the expandable node, one or more new nodes are constructed and added to the search tree, with the expandable node as the parent and each child

corresponding to a valid action from its associated state. The mechanism for selection and expansion is called *tree policy*, and it is typically based on the node statistics.

(iii) **Simulation**: Trajectories in the MDP are sampled from the new node until a terminal state is reached and the return is recorded. The *default* or *simulation policy* dictates the probability of each action, with the standard version using uniform random sampling. The intermediate states encountered when sampling are not added to the search tree.

(iv) **Backpropagation**: The return is backpropagated to the root of the search tree, and the statistics of each node that was selected by the tree policy are updated.

Once the computational budget is exhausted and the search is stopped, the statistics of the root node's children are used to decide the next action $A_t$. The algorithm then constructs a new search tree from the next state $S_{t+1}$, following the above steps. This proceeds iteratively until the algorithm encounters a terminal state $S_T$.

The tree policy used by the algorithm needs to trade-off exploration and exploitation in order to balance actions that are already known to lead to high returns against yet-unexplored paths in the MDP for which the returns are still to be estimated. The exploration–exploitation trade-off has been widely studied in the multi-armed bandit setting, which may be thought of as a single-state MDP. A representative algorithm is the upper confidence bound (UCB) algorithm [25], which computes confidence intervals for each action and chooses, at each step, the action with the largest upper bound on the reward, embodying the principle of optimism in the face of uncertainty. UCT is a variant of MCTS that applies the principles behind UCB to the tree search setting. Namely, the selection decision at each node is framed as an independent multi-armed bandit problem. At decision time, the *tree policy* of the algorithm selects the child node corresponding to action $a$ that maximizes $\mathrm{UCT}(s,a) = R(s,a)/N(s,a) + 2c_p\sqrt{2\ln N(s)/N(s,a)}$, where $R(s,a)$ is the sum of returns obtained when taking action $a$ in state $s$, $N(s)$ is the number of parent node visits, $N(s,a)$ the number of child node visits and $c_p$ is a constant that controls the level of exploration [21].

MCTS has found wide applicability in a variety of decision-making and optimization scenarios. Our own source of inspiration has been the successful applications of the technique to 'connectionist' games such as Morpion Solitaire [26] and Hex [27,28], which share some similarities since they involve deciding which edges to create between nodes placed on a regular grid. Domain knowledge [29] can be integrated with MCTS such that its performance is enhanced, an aspect that we leverage in constructing the reduction and simulation policies of the proposed algorithm. Alternatively, one may use *learned* knowledge based on linear function approximation [30] as well as deep neural networks [31–33]. The latter category of methods has enjoyed substantial interest recently for solving combinatorial optimization problems [34,35].

## (c) Spatial networks and objectives

We define a *spatial network* as the tuple $G = (V, E, f, w)$. $V$ is the set of vertices, and $E$ is the set of edges. $f : V \to M$ is a function that maps nodes in the graph to a set of positions $M$. We require that $M$ admits a metric $d$, i.e. there exists a function $d : M \times M \to \mathbb{R}^+$ defining a pairwise *distance* between elements in $M$. The tuple $(M, d)$ defines a *space*, common examples of which include Euclidean space and spherical geometry. $w : E \to \mathbb{R}^+$ associates a *weight* with each edge, a positive real-valued number. For example, the weight can be interpreted as the capacity of a link in networks that carry traffic.

We consider two global objectives $\mathcal{F}$ for spatial networks that are representative of a wide class of properties relevant in real-world situations. Depending on the domain, there are many other global objectives for spatial networks that can be considered, to which the approach that we present is directly applicable.

### (i) Efficiency

Efficiency is a metric quantifying how well a network exchanges information. It measures how fast information can travel between any pair of nodes in the network on average, and

is hypothesized to be an underlying principle for the organization of networks [20]. Efficiency does not solely depend on topology but also on the *spatial* distances between the nodes in the network. We adopt the definition of global efficiency as formalized in [20], and let $\mathcal{F}_E(G) = (1/N(N-1)) \sum_{i \neq j \in V} (1/sp(i,j))$, where $sp(i,j)$ is the cumulative length (i.e. the summed distances) of the shortest path between vertices $i$ and $j$. To normalize, we divide by the ideal efficiency $\mathcal{F}_E^*(G) = (1/N(N-1)) \sum_{i \neq j \in V} (1/d(i,j))$, and possible values are thus in $[0,1]$.[1] Efficiency is computable in $O(|V|^3)$ by using the 'weighted' version of the Floyd–Warshall shortest path algorithm,[2] in which the spatial distances over the edges are given as weights (note, however, that this differs from the weight function $w$ defined above, which is akin to capacity). The path lengths are provided raw and are not normalized by the straight line distances.

### (ii) Robustness

We consider the property of robustness, i.e. the resilience of the network in the face of removals of nodes. We adopt a robustness measure widely used in the literature [37,38] and of practical interest and applicability based on the largest connected component (LCC), i.e. the component with most nodes. In particular, we use the definition in [15], which considers the size of the LCC as nodes are removed from the network. We consider only the targeted attack case as previous work has found it is more challenging [12,37]. We define the robustness measure as $\mathcal{F}_R(G) = \mathbb{E}_\xi[(1/N) \sum_{i=1}^{N} s(G, \xi, i)]$, where $s(G, \xi, i)$ denotes the fraction of nodes in the LCC of $G$ after the removal of the first $i$ nodes in the permutation $\xi$ (in which nodes appear in descending order of their degrees). Possible values are in $[(1/N), 0.5)$. This quantity can be estimated using Monte Carlo simulations and scales as $O(|V|^2 \times (|V| + |E|))$.

It is worth noting that the value of the objective functions typically increases the more edges exist in the network (the complete graph has both the highest possible efficiency and robustness). However, it may be necessary to balance the contribution of an edge to the objective with its cost. The proposed method explicitly accounts for this trade-off, which is widely observed in infrastructure and brain networks [39,40].

## (d) Existing methods for goal-directed graph construction

Given an objective function such as the ones defined above and a budget of structural modifications, a worthwhile question to consider is how to modify a graph so as to optimize the function. This has been addressed by various prior works from several communities. We refer to this problem as *goal-directed graph construction*, noting that this term includes the extension of an existing graph.

In the network science literature, authors have been particularly interested in improving resilience. The authors of [14] approach this problem by edge addition or rewiring, based on random or preferential degree-based modifications. In [15], the authors propose a greedy modification scheme based on random edge selection and swapping if the resilience metric improves, with the threshold of the swapping criterion being annealed over time. For the problem of *de novo* graph generation, in [41] an algorithm that produces graphs with similarly high values of robustness at a much lower computational cost is proposed. The authors of [17] consider both preferential (degree-based) and greedy (effective graph resistance), finding that greedy strategies are generally more effective but also more expensive to compute.

In the machine learning community, model-free RL techniques have been applied for deriving adversarial examples for graph-based classifiers by changing the network structure [13] and the goal-directed generation of molecular graphs [10]. [12] formulated the goal-directed construction of a graph as an MDP and proposed a method based on RL and graph neural networks, showing

[1]It is worth noting that efficiency is a more suitable metric for measuring the exchange of information than the inverse average path length between pairs of nodes. In the extreme case, where the network is disconnected (and thus some paths lengths are infinite), this metric does not go to infinity. More generally, this metric is better suited for systems in which information is exchanged in a parallel, rather than sequential, way [20].

[2]In practice, this may be made faster by considering dynamic shortest path algorithms, e.g. [36].

some advantages over prior methods in the network science literature in terms of its ability to optimize the objective and the fast runtime of the policy once it has been trained.

Related problems have also attracted the attention of the operations research community. In [42], the authors consider a problem with node upgrade actions with the goal of reducing pairwise shortest path distances, which resembles our case study on efficiency improvement. Work on the Pre-disaster Transportation Network Preparation problem [43,44] considers strengthening a highway network to cope with possible natural disasters such as floods. Another important class of approaches for network design is based on the concept of *spanner of a graph*, i.e. a subgraph of the given graph of in which the length of any path does not exceed a given threshold. Such methods have found particularly successful applications in designing communication networks [45,46]. However, none of the methods are applicable in our setting given their focus on specific problem definitions.

## 3. Proposed method

In this section, we first formulate the construction of spatial networks in terms of a global objective function as an MDP. Subsequently, we propose a variant of the UCT algorithm (SG-UCT) for planning, which exploits the characteristics of spatial networks, for this MDP.

### (a) Spatial graph construction as Markov decision processes

#### (i) Spatial constraints in network construction

Spatial networks that can be observed in the real world typically incur a cost to edge creation. To aid intuition, consider the example of a power grid: the cost of a link depends, among other aspects, on its geographical distance as well as on its capacity. We let $c(i,j)$ denote the cost of edge $(i,j)$ and $C(\Gamma) = \sum_{(i,j)\in\Gamma} c(i,j)$ be the cost of a *set* of edges $\Gamma$. We consider $c(i,j) = w(i,j) * d(f(i),f(j))$ to capture the notion that longer, higher capacity connections are more expensive. We are aware that this is just one of the possible definitions and different notions of cost may be desirable depending on the domain. However, it can be considered representative of several real-world scenarios, in which cost can be modelled in a similar manner. To ensure fair comparisons, we normalize costs $c(i,j)$ to be in $[0,1]$.

#### (ii) Problem statement

Let $\mathbf{G}^{(N)}$ be the set of labelled, undirected, weighted, spatial networks with $N$ nodes. We let $\mathcal{F}: \mathbf{G}^{(N)} \to [0,1]$ be an objective function, and $b_0 \in \mathbb{R}^+$ be a modification budget. Given an initial graph $G_0 = (V, E_0, f, w) \in \mathbf{G}^{(N)}$, the aim is to add a set of edges $\Gamma$ to $G_0$ such that the graph $G_* = (V, E_*, f, w)$ satisfies

$$G_* = \arg\max_{G'\in\mathbf{G}'} \mathcal{F}(G'), \quad \text{where } \mathbf{G}' = \{G \in \mathbf{G}^{(N)} \mid E = E_0 \cup \Gamma \,.\, C(\Gamma) \le b_0\}. \tag{3.1}$$

#### (iii) Markov decision processes formulation

We next define the MDP elements:

*State*: The state $S_t$ is a three-tuple $(G_t, \sigma_t, b_t)$ containing the spatial graph $G_t = (V, E_t, f, w)$, an *edge stub* $\sigma_t \in V$, and the remaining budget $b_t$. $\sigma_t$ can be either the empty set $\varnothing$ or the singleton $\{v\}$, where $v \in V$. If the edge stub is non-empty, it means that the agent has 'committed' in the previous step to creating an edge originating at the edge stub.

*Action*: For scalability to large graphs, an action $A_t$ corresponds to the selection of a *single* node in $V$ (thus having at most $|V|$ choices). We enforce spatial constraints as follows: given a node $i$,

we define the set $\mathcal{K}(i)$ of *connectable* nodes $j$ that represent realizable connections. We let

$$\mathcal{K}(i) = \{j \in V \mid c(i,j) \leq \rho \max_{k \in V.(i,k) \in E_0} c(i,k)\},$$

which formalizes the idea that a node can only connect as far as a proportion $\rho$ of its longest existing connection, with $\mathcal{K}(i)$ fixed based on the initial graph $G_0$. This allows long-range connections if they already exist in the network. Given an unspent connection budget $b_t$, we let the set $\mathcal{B}(i, b_t) = \{j \in \mathcal{K}(i) \mid c(i,j) \leq b_t\}$ consist of those connectable nodes whose cost is not more than the unspent budget. Letting the degree of node $v$ be $d_v$, available actions[3] are defined as

$$\mathcal{A}(S_t = ((V, E_t, f, w), \varnothing, b_t)) = \{v \in V \mid d_v < |V| - 1 \ \wedge \ |\mathcal{B}(v, b_t)| > 0\}$$

and

$$\mathcal{A}(S_t = ((V, E_t, f, w), \{\sigma_t\}, b_t)) = \{v \in V \mid (\sigma_t, v) \notin E_t \ \wedge \ v \in \mathcal{B}(\sigma_t, b_t)\}.$$

*Transitions*: The deterministic transition model adds an edge every two steps. Concretely, we define it as $P(S_t = s' | S_{t-1} = s, A_{t-1} = a) = \delta_{S_t s'}$, where $s' =$

$$\begin{aligned} &((V, E_{t-1} \cup (\sigma_{t-1}, a), f, w), \varnothing, b_{t-1} - c(\sigma_{t-1}, a)), \quad \text{if } 2 \mid t \\ &((V, E_{t-1}, f, w), \{a\}, b_{t-1}), \quad \text{otherwise.} \end{aligned}$$

*Reward*: The final reward $R_T$ is defined as $\mathcal{F}(G_T) - \mathcal{F}(G_0)$ and intermediary rewards are 0. We do not provide intermediate rewards due to the large computational cost (at least cubic in the number of nodes) needed for calculating the objective functions. However, intermediate rewards may be provided for less computationally demanding objectives.

Episodes in this MDP proceed for an arbitrary number of steps until the budget is exhausted or no valid actions remain (concretely, $|\mathcal{A}(S_t)| = 0$). Since we are in the finite horizon case, we let $\gamma = 1$. Given the MDP definition above, the problem specified in equation (3.1) can be reinterpreted as finding the trajectory $\tau_*$ that starts at $S_0 = (G_0, \varnothing, b_0)$ such that the final reward $R_T$ is maximal—actions along this trajectory will define the set of edges $\Gamma$.

## (b) Algorithm

The formulation above can, in principle, be used with any planning algorithm for MDPs in order to identify an optimal set of edges to add to the network. The UCT algorithm, discussed in §2, is one such algorithm that has proven very effective in a variety of settings. We refer the reader to [24] for an in-depth description of the algorithm and its various applications. However, the generic UCT algorithm assumes very little about the particulars of the problem under consideration, which, in the context of spatial network construction, may lead to sub-optimal solutions. In this section, we identify and address concerns specific to this family of problems, and formulate the SG-UCT variant of UCT in algorithm 1. The evaluation presented in §4 compares SG-UCT to UCT and other baselines, and contains an ablation study of SG-UCT's components.

### (i) Best trajectory memoization

The standard UCT algorithm is applicable in a variety of settings, including multi-agent, stochastic environments. For example, in two-player games, an agent needs to re-plan from the new state that is arrived at after the opponent executes its move. However, the single-agent (puzzle), deterministic nature of the problem considered means that there is no need to re-plan trajectories after a stochastic event: the agent can plan all its actions from the very beginning in

[3]Depending on the type of network being considered, in practice there may be different types of constraints on the connections that can be realized. For example, in transportation networks, there can be obstacles that make link creation impossible, such as prohibitive landforms or populated areas. In circuits and utility lines, planarity is a desirable characteristic as it makes circuit design cheaper. Such constraints can be captured by the definition of $\mathcal{K}(i)$ and enforced by the environment when providing the agent with available actions $\mathcal{A}(s)$. Conversely, defining $\mathcal{K}(i) = V \setminus \{i\}$ recovers the simplified case where no constraints are imposed.

**Algorithm 1.** Spatial Graph UCT (SG-UCT).

**Input:** spatial graph $G_0=(V, E_0, f, w)$,
objective function $\mathcal{F}$, budget $b_0$, reduction policy $\phi$
**Output:** actions $A_0, \ldots A_{T-1}$
**for** $i$ in $V$: compute $\mathcal{K}(i)$
compute $\Phi=\phi(G_0)$ % { apply reduction policy $\phi$}
$t=0$, $b_0=\tau * C(E_0)$, $S_0=(G_0, \varnothing, b_0)$
$\Delta_{max}=-\infty$, *bestActs*=*array*(), *pastActs*=*array*()
**loop**
  **if** $|\mathcal{A}_\phi(S_t)|=0$ **then** return *bestActs*
  create root node $v_t$ from $S_t$
  **for** $i=0$ to $n_{sims}$
    $v_l$, *treeActs*= TREEPOLICY($v_t$, $\Phi$)
    $\Delta$, *outActs*= MINCOSTPOLICY($v_l$, $\Phi$) % { cost-sensitive default policy}
    BACKUP($v_l$, $\Delta$)
    **if** $\Delta > \Delta_{\max}$ **then**
      *bestActs*=[*pastActs*, *treeActs*, *outActs*]
      $\Delta_{max}=\Delta$ % { memorize best trajectory}
  *child*= MAXCHILD($v_t$)
  *pastActs.append*(*child.action*)
  $t+=1$, $S_t$=*child.state*

a single step. We thus propose the following modification over UCT: memorizing the trajectory with the highest reward found during the rollouts, and returning it at the end of the search. We name this best trajectory memoization, BTM. This is similar in spirit (albeit much simpler) to ideas used in Reflexive and Nested MCTS for deterministic puzzles, where the best move found at lower levels of a nested search are used to inform the upper level [47,48].

### (ii) Cost-sensitive default policy

The standard default policy used to perform out-of-tree actions in the UCT framework is based on random rollouts. Even though this is free from bias, rollouts can lead to high-variance estimates, which can hurt the performance of the search. Previous work has considered hand-crafted heuristics and learned policies as alternatives, although, perhaps counterintuitively, learned policies may lead to worse results [29]. As initially discussed in §2, the value of the objective functions we consider grows with the number of edges of the graph. We thus propose the following default policy for spatial networks: sampling each edge with probability inversely proportional to its cost. Formally, we let the probability of edge $(i, j)$ being selected during rollouts be proportional to $(\max_{i,j} c(i, j) - c(i, j))^\beta$, where $\beta$ denotes the level of bias. $\beta \to 0$ reduces to random choices, while $\beta \to \infty$ selects the minimum cost edge. This is very inexpensive computationally, as the edge costs only need to be calculated once, at the start of the search.

### (iii) Action space reduction

In certain domains, the number of actions available to an agent is large, which can greatly affect scalability. Previous work in RL has considered decomposing actions into independent sub-actions [49], generalizing across similar actions by embedding them in a continuous space [50], or learning which actions to eliminate via supervision provided by the environment [51]. Existing approaches in planning consider progressively widening the search based on a heuristic [52] or learning a partial policy for eliminating actions in the search tree [53].

Concretely, in this MDP, the action space grows linearly in the number of nodes. This is partly addressed by the imposed connectivity constraints: once an edge stub is selected (equivalently, at odd values of $t$), the branching factor of the search is small since only *connectable* nodes need

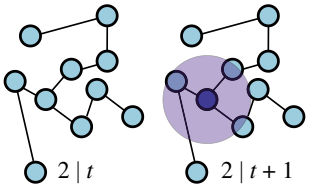


**Figure 3.** Illustration of the asymmetry in the number of actions at even (state has no edge stub) versus odd $t$ (state has an edge stub). Spatial constraints are imposed in the latter case, reducing the number of actions.

to be considered. However, the number of actions when selecting the origin node of the edge (even values of $t$) remains large, which might become detrimental to performance as the size of the network grows (as illustrated in figure 3). Can this be mitigated?

We consider limiting the nodes that can initiate connections to a subset—which prunes away all branches in the search tree that are not part of this set. Concretely, let a *reduction policy* $\phi$ be a function that, given the initial graph $G_0$, outputs a strict subset of its nodes.[4] Then, we modify our definition of allowed actions as follows: under a reduction policy $\phi$, we define

$$\begin{aligned}\mathcal{A}_\phi(S_t) &= \mathcal{A}(S_t) \cap \phi(G_0), \quad \text{if } 2 \mid t \\ &= \mathcal{A}(S_t), \quad \text{otherwise.}\end{aligned}$$

We investigate the following class of reduction policies: a node $i$ is included in $\phi(G_0)$ if and only if it is among the top nodes ranked by a local node statistic $\lambda(i)$. Specifically, we consider the $\lambda(i)$ listed below, where $gain(i,j) = \mathcal{F}(V, E \cup (i,j), f, w) - \mathcal{F}(V, E, f, w)$. Since the performance of reduction strategies may depend on $\mathcal{F}$, we treat them as a tunable hyperparameter.

— *Degree (DEG)*: $d_i$; *Inverse Degree (ID)*: $\max_j d_j - d_i$; *Number of Connections (NC)*: $|\mathcal{K}(i)|$
— *Best Edge (BE)*: $\max_{j \in \mathcal{K}(i)} gain(i,j)$; *BE Cost Sensitive (BECS)*: $\max_{j \in \mathcal{K}(i)} (gain(i,j)/c(i,j))$
— *Average Edge (AE)*: $\sum_{j \in \mathcal{K}(i)} gain(i,j)/|\mathcal{K}(i)|$; *AECS*: $\sum_{j \in \mathcal{K}(i)} (gain(i,j)/c(i,j))/|\mathcal{K}(i)|$.

# 4. Experiments

## (a) Experimental protocol

### (i) Definitions of space and distance

For all experiments in this paper, we consider the unit two-dimensional square as our space, i.e. we let $M = [0,1] \times [0,1]$ and the distance $d$ be Euclidean distance. In case the graph is defined on a spherical coordinate system (as is the case with physical networks positioned on Earth), we use the WGS84 variant of the Mercator projection to project nodes to the plane; then normalize to the unit plane. We opt to project the coordinates on a plane, rather than on a unit sphere, since the networks that we consider are at the geographical scale of a city or at most a country, for which a plane is a reasonable local approximation. Instead, projecting on a unit sphere would introduce more significant distortions. For simplicity, we consider uniform weights, i.e. $w(i,j) = 1 \; \forall \; (i,j) \in E$. The approach can be extended to networks with heterogenous weights by defining the action space at even timesteps as a product between the set of valid nodes and the set of possible weight values, as well as adopting an objective function that incorporates them as a drop-in replacement (e.g. the formalization of efficiency in [54]).

[4]Learning a reduction policy in a data-driven way is also possible; however, obtaining the supervision signal (i.e. node rankings over multiple MCTS runs) is very expensive. Furthermore, since we prioritize performance on specific graph instances over generalizable policies, simple statistics may be sufficient. Still, a learned reduction policy that predicts an entire set at once may be able to identify better subsets than individual statistics alone. Furthermore, a possible limitation of using a reduction policy is that the optimal trajectory in the full MDP may be excluded. We consider these worthwhile directions for future investigations.

**Table 1.** Real-world graphs considered in the evaluation.

| dataset | graph | $\|V\|$ | $\|E\|$ |
|---|---|---|---|
| Internet | Colt | 146 | 178 |
| | GtsCe | 130 | 169 |
| | TataNld | 141 | 187 |
| | UsCarrier | 138 | 161 |
| Metro | Barcelona | 135 | 159 |
| | Beijing | 126 | 139 |
| | Mexico | 147 | 164 |
| | Moscow | 134 | 156 |
| | Osaka | 107 | 122 |

### (ii) Synthetic and real-world graphs

As a means of generating synthetic graph data, we use the popular model proposed by Kaiser and Hilgetag in [55], which simulates a process of growth for spatial networks. Related to the Waxman model [56], in this model the probability that a connection is created is inversely proportional to its distance from existing nodes. The distinguishing feature of this model is that, unlike, e.g. the random geometric graph [57], this model produces connected networks: a crucial characteristic for the types of objectives we consider. We henceforth refer to this model as Kaiser–Hilgetag (KH). We use $\alpha_{KH} = 10$ and $\beta_{KH} = 10^{-3}$, which yields sparse graphs with scale-free degree distributions—a structure similar to road infrastructure networks. We also evaluate performance on networks belonging to the following real-world datasets, detailed in table 1: *Internet* (a dataset of internet backbone infrastructure from a variety of ISPs [58], which display spatial characteristics due to their country-level scales) and *Metro* (a dataset of metro networks in major cities around the world [59]). Due to computational budget constraints, we limit the sizes of networks considered to $|V| = 150$.

### (iii) Set-up

For all experiments, we allow agents a modification budget equal to a proportion $\tau$ of the total cost of the edges of the original graph, i.e. $b_0 = \tau * C(E_0)$. We use $\tau = 0.1$. We let $\rho = 1$ for synthetic graphs and $\rho = 2$ for real-world graphs, respectively. This is because the real-world networks tend to have comparatively less long-range connections than the synthetic ones. Moreover, a value of $\rho$ that is too low could prohibit longer connections entirely in case they do not already exist in the seed network, severely restricting the feasible solution space. Confidence intervals are computed using results of 10 runs, each initialized using a different random seed. Rollouts are not truncated. We allow a number of node expansions per move $n_{\text{sims}}$ equal to $20 * |V|$ (a larger number of expansions can improve performance, but leads to diminishing returns), and select as the move at each step the node with the maximum average value (commonly referred to as MaxChild). Full details of the hyperparameter selection methodology and the values used are provided in the electronic supplementary material.

### (iv) Evaluation metrics

We report the rewards obtained by the approaches taken into consideration, i.e. the difference in $\mathcal{F}$ between the final and initial graphs. This has the advantage that the values are directly interpretable and correspond to the gain in the objective function that can be obtained with a certain budget. We note that the relatively small scale of the values reported in the tables and

figures are due to the fact that they represent differences in objective function values that range between $[0, 1]$ for efficiency and $[1/N, 0.5]$ for robustness, as reported in §2(c).

### (v) Baselines

The baselines we compare against are detailed below and represent several prior methods discussed in §2(d). All of the techniques, including our proposed algorithm, solve the problem approximately, since no methods currently exist to solve the problem exactly beyond the smallest of graphs. We do not consider previous RL-based methods such as [12] directly, since they are unsuitable for training at the scale of the largest graphs considered in this work.

— *Random* ($\mathcal{F}_E, \mathcal{F}_R$): Randomly selects an available action.
— *Greedy* ($\mathcal{F}_E, \mathcal{F}_R$): A local search that selects the edge that gives the biggest improvement in $\mathcal{F}$: formally, it adds the edge $(i, j)$ that satisfies $\text{argmax}_{i,j}\, gain(i, j)$ to the graph structure. This approach builds a shallow search tree of depth two, evaluating the objective function for all leaf nodes. We also evaluate the cost-sensitive variant *GreedyCS*, for which the gain is offset by the cost: $\text{argmax}_{i,j}\, (gain(i, j)/c(i, j))$.
— *MinCost* ($\mathcal{F}_E, \mathcal{F}_R$): Selects edge $(i, j)$ that satisfies $\text{argmin}_{i,j}\, c(i, j)$.
— *LBHB* ($\mathcal{F}_E$): Adds an edge between the node with Lowest Betweenness and the node with Highest Betweenness; formally, letting the betweenness centrality of node $i$ be $g_i$, this strategy adds an edge between nodes $\text{argmin}_i\, g_i$ and $\text{argmax}_j\, g_j$.
— *LDP* ($\mathcal{F}_R$): Adds an edge between the vertices with the *lowest degree product*, i.e. vertices $i, j$ that satisfy $\text{argmin}_{i,j}\, d_i \cdot d_j$.
— *FV* ($\mathcal{F}_R$): Adds an edge between the vertices $i, j$ that satisfy $\text{argmax}_{i,j}\, |\mathbf{y}_i - \mathbf{y}_j|$, where $\mathbf{y}$ is the Fiedler Vector [16,60].
— *ERes* ($\mathcal{F}_R$): Adds an edge between vertices with the highest pairwise effective resistance, i.e. nodes $i, j$ that satisfy $\text{argmax}_{i,j}\, \Omega_{i,j}$. $\Omega_{i,j}$ is defined as $(\hat{\mathcal{L}}^{-1})_{ii} + (\hat{\mathcal{L}}^{-1})_{jj} - 2(\hat{\mathcal{L}}^{-1})_{ij}$, where $\hat{\mathcal{L}}^{-1}$ is the pseudoinverse of the graph Laplacian $\mathcal{L}$ [17].

## (b) Evaluation results

### (i) Synthetic graph results

In this experiment, we consider 50 KH graphs each of sizes $\{25, 50, 75\}$. The obtained results are shown in the top half of table 2. We summarize our findings as follows: SG-UCT outperforms UCT and all other methods in all the settings tested, obtaining 13% and 32% better performance than UCT on the largest synthetic graphs for the efficiency and robustness measures respectively. For $\mathcal{F}_R$, UCT outperforms all baselines, while for $\mathcal{F}_E$ the performance of the Greedy baselines is superior to UCT. Interestingly, MinCost yields solutions that are superior to all other heuristics and comparable to search-based methods while being very cheap to evaluate. Furthermore, UCT performance decays in comparison to the baselines as the size of the graph increases.

### (ii) Real-world graph results

The results obtained for real-world graphs are shown in table 3. As with synthetic graphs, we find that SG-UCT performs better than UCT and all other methods in all settings considered in the evaluation. The aggregated differences in performance between SG-UCT and UCT are 10% and 39% for $\mathcal{F}_E$ and $\mathcal{F}_R$, respectively.

### (iii) Ablation study

Since SG-UCT comprises three components that are active at the same time as defined in §3(b), we conduct an ablation study on synthetic graphs in order to assess their impact. To achieve this, we consider standard UCT and enable each of the three components in turn, measuring the performance of the algorithm. The obtained results are shown in table 4, where the results for each

**Table 2.** Gains in the values of the objective function $\mathcal{F}$ between the optimized and the original graphs obtained by baselines, UCT and SG-UCT on synthetic graphs.

| objective | $\mathcal{F}_E$ | | | $\mathcal{F}_R$ | | |
|---|---|---|---|---|---|---|
| $\lvert V\rvert$ | 25 | 50 | 75 | 25 | 50 | 75 |
| Random | 0.128 ± 0.008 | 0.089 ± 0.005 | 0.077 ± 0.004 | 0.031 ± 0.002 | 0.033 ± 0.002 | 0.035 ± 0.002 |
| Greedy | 0.298 | 0.335 | 0.339 | 0.064 | 0.078 | 0.074 |
| $\text{Greedy}_{\text{CS}}$ | 0.281 | 0.311 | 0.319 | 0.083 | 0.102 | 0.115 |
| LDP | — | — | — | 0.049 | 0.044 | 0.040 |
| FV | — | — | — | 0.051 | 0.049 | 0.049 |
| ERes | — | — | — | 0.054 | 0.057 | 0.052 |
| MinCost | 0.270 | 0.303 | 0.315 | 0.065 | 0.082 | 0.099 |
| LBHB | 0.119 | 0.081 | 0.072 | — | — | — |
| UCT | 0.288 ± 0.003 | 0.307 ± 0.003 | 0.311 ± 0.003 | 0.092 ± 0.001 | 0.112 ± 0.002 | 0.120 ± 0.001 |
| SG-UCT (ours) | **0.305** ± 0.000 | **0.341** ± 0.000 | **0.352** ± 0.001 | **0.107** ± 0.001 | **0.140** ± 0.001 | **0.158** ± 0.000 |

component are separated by the horizontal lines. $\text{SG-UCT}_{\text{BTM}}$ denotes UCT with best trajectory memoization, $\text{SG-UCT}_{\text{MINCOST}}$ denotes UCT with the cost-based default policy, $\text{SG-UCT}_{\phi-q}$ for $q$ in $\{40, 60, 80\}$ denotes UCT with a particular reduction policy $\phi$, and $q$ represents the percentage of original nodes that are selected by $\phi$.

We find that BTM indeed brings a net improvement in performance: on average, 5% for $\mathcal{F}_E$ and 11% for $\mathcal{F}_R$. The benefit of the cost-based default policy is substantial (especially for $\mathcal{F}_R$), ranging from 4% on small graphs to 27% on the largest graphs considered, and increases the higher the level of bias. This is further evidenced in figure 4, which shows the average reward obtained as a function of $\beta$. This illustrates that the cost-based simulation policy yields the largest gains in the value of the objective function out of all the three algorithmic components. In terms of reduction policies, even for a random selection of nodes, we find that the performance penalty paid is comparatively small: a 60% reduction in actions translates to at most 15% reduction in performance, and as little as 5%; the impact of random action reduction becomes smaller as the size of the network grows. The best-performing reduction policies are those based on a node's gains, with BECS and AECS outperforming UCT with no action reduction. For the $\mathcal{F}_R$ objective, a poor choice of bias can be harmful: prioritizing nodes with high degrees leads to a 32% reduction in performance compared to UCT, while a bias towards lower-degree nodes is beneficial.

## (c) Execution time and scalability

An important aspect to consider is the computational time needed to decide which edges to be added. The wall clock time for a complete run of the algorithms on the real-world graphs is shown in table 5 and is measured on a single core of an Intel Xeon E5-2630 v3 (2014) processor.

A distinction between heuristic and search-based methods is immediately apparent. The former methods, which do not evaluate the objective directly but instead rely on local or spectral properties, result in timings that do not exceed a few seconds. The latter category, comprising the greedy and tree search variants, require larger timescales. Within this category, the cost-sensitive methods (GreedyCS and SG-UCT) typically require more time compared to their cost-agnostic counterpart. This is due to longer action sequences resulting in more edges being added and hence more simulations being performed, as well as the overheads introduced for computing pairwise distances and using them to weight edge choices.

In order to further probe the scaling behaviour of the search-based approaches, we carry out an experiment using synthetic KH graphs of sizes between $N = 25$ and $N = 200$. Each column

**Table 3.** Gains in the values of the objective function $\mathcal{F}$ between the optimized and the original graphs obtained by baselines, UCT and SG-UCT on real-world graphs.

| $\mathcal{F}$ | G | graph | Random | Greedy | $\text{Greedy}_{CS}$ | LDP | FV | ERes | MinCost | LBHB | UCT | SG-UCT (ours) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\mathcal{F}_E$ | Internet | Colt | 0.081 ± 0.003 | 0.180 | 0.127 | — | — | — | 0.127 | 0.098 | 0.164 ± 0.003 | **0.199** ± 0.000 |
| | | GtsCe | 0.017 ± 0.007 | 0.123 | 0.089 | — | — | — | 0.082 | 0.014 | 0.110 ± 0.004 | **0.125** ± 0.002 |
| | | TataNld | 0.020 ± 0.004 | 0.106 | 0.082 | — | — | — | 0.078 | 0.015 | 0.102 ± 0.003 | **0.110** ± 0.002 |
| | | UsCarrier | 0.026 ± 0.014 | **0.178** | 0.097 | — | — | — | 0.097 | 0.026 | 0.171 ± 0.005 | **0.178** ± 0.002 |
| | Metro | Barcelona | 0.020 ± 0.005 | 0.071 | 0.063 | — | — | — | 0.063 | 0.003 | 0.067 ± 0.002 | **0.076** ± 0.000 |
| | | Beijing | 0.008 ± 0.003 | 0.036 | 0.033 | — | — | — | 0.028 | 0.003 | 0.041 ± 0.001 | **0.046** ± 0.001 |
| | | Mexico | 0.007 ± 0.002 | 0.037 | 0.035 | — | — | — | 0.032 | 0.011 | 0.037 ± 0.001 | **0.041** ± 0.000 |
| | | Moscow | 0.011 ± 0.003 | 0.052 | 0.042 | — | — | — | 0.038 | 0.007 | 0.043 ± 0.001 | **0.053** ± 0.001 |
| | | Osaka | 0.017 ± 0.005 | 0.097 | 0.082 | — | — | — | 0.082 | 0.010 | 0.093 ± 0.003 | **0.102** ± 0.000 |
| $\mathcal{F}_R$ | Internet | Colt | 0.007 ± 0.004 | 0.034 | 0.075 | 0.005 | 0.006 | 0.009 | 0.075 | — | 0.055 ± 0.003 | **0.089** ± 0.000 |
| | | GtsCe | 0.023 ± 0.011 | 0.064 | 0.101 | 0.048 | 0.017 | 0.031 | 0.099 | — | 0.098 ± 0.005 | **0.155** ± 0.003 |
| | | TataNld | 0.017 ± 0.010 | 0.043 | 0.083 | 0.011 | −0.002 | 0.013 | 0.074 | — | 0.093 ± 0.006 | **0.119** ± 0.002 |
| | | UsCarrier | 0.010 ± 0.004 | 0.078 | 0.060 | 0.035 | 0.038 | 0.033 | 0.041 | — | 0.085 ± 0.007 | **0.125** ± 0.003 |
| | Metro | Barcelona | 0.020 ± 0.007 | 0.057 | 0.073 | 0.010 | 0.009 | 0.036 | 0.071 | — | 0.076 ± 0.004 | **0.115** ± 0.002 |
| | | Beijing | 0.004 ± 0.004 | 0.014 | 0.054 | 0.003 | 0.002 | 0.001 | 0.037 | — | 0.055 ± 0.003 | **0.062** ± 0.001 |
| | | Mexico | 0.007 ± 0.004 | 0.040 | 0.043 | 0.003 | 0.005 | 0.011 | 0.038 | — | 0.051 ± 0.002 | **0.068** ± 0.001 |
| | | Moscow | 0.013 ± 0.005 | 0.072 | 0.057 | 0.033 | 0.042 | 0.034 | 0.031 | — | 0.090 ± 0.003 | **0.109** ± 0.002 |
| | | Osaka | 0.003 ± 0.006 | 0.042 | 0.064 | 0.008 | 0.011 | 0.015 | 0.064 | — | 0.066 ± 0.004 | **0.072** ± 0.001 |

**Table 4.** Ablation study that examines the impact of the three components of SG-UCT by enabling each of them separately in standard UCT. The components are best trajectory memoization, the MINCOST simulation policy, and the various choices of a reduction policy. The results are separated by horizontal lines. Each value represents the gain in the value of the objective function $\mathcal{F}$ between the optimized and the original graphs.

| objective | $\mathcal{F}_E$ | | | $\mathcal{F}_R$ | | |
|---|---|---|---|---|---|---|
| $\|V\|$ | 25 | 50 | 75 | 25 | 50 | 75 |
| UCT | $0.288 \pm 0.003$ | $0.307 \pm 0.003$ | $0.311 \pm 0.003$ | $0.092 \pm 0.001$ | $0.112 \pm 0.002$ | $0.120 \pm 0.001$ |
| SG-UCT$_{\text{BTM}}$ | $0.304 \pm 0.001$ | $0.324 \pm 0.002$ | $0.324 \pm 0.002$ | $0.106 \pm 0.001$ | $0.123 \pm 0.001$ | $0.128 \pm 0.001$ |
| SG-UCT$_{\text{MINCOST}}$ | $0.299 \pm 0.001$ | $0.327 \pm 0.001$ | $0.333 \pm 0.001$ | $0.105 \pm 0.001$ | $0.131 \pm 0.001$ | $0.153 \pm 0.001$ |
| SG-UCT$_{\text{RAND-80}}$ | $0.284 \pm 0.005$ | $0.305 \pm 0.003$ | $0.303 \pm 0.003$ | $0.091 \pm 0.001$ | $0.111 \pm 0.001$ | $0.119 \pm 0.001$ |
| SG-UCT$_{\text{RAND-60}}$ | $0.271 \pm 0.007$ | $0.288 \pm 0.004$ | $0.288 \pm 0.003$ | $0.089 \pm 0.003$ | $0.107 \pm 0.002$ | $0.115 \pm 0.002$ |
| SG-UCT$_{\text{RAND-40}}$ | $0.238 \pm 0.009$ | $0.263 \pm 0.005$ | $0.271 \pm 0.003$ | $0.083 \pm 0.001$ | $0.102 \pm 0.002$ | $0.110 \pm 0.002$ |
| SG-UCT$_{\text{DEG-40}}$ | $0.237 \pm 0.003$ | $0.262 \pm 0.003$ | $0.255 \pm 0.002$ | $0.069 \pm 0.001$ | $0.086 \pm 0.001$ | $0.092 \pm 0.001$ |
| SG-UCT$_{\text{ID-40}}$ | $0.235 \pm 0.002$ | $0.268 \pm 0.001$ | $0.283 \pm 0.001$ | $0.094 \pm 0.001$ | $0.114 \pm 0.001$ | $0.124 \pm 0.001$ |
| SG-UCT$_{\text{NC-40}}$ | $0.234 \pm 0.003$ | $0.268 \pm 0.002$ | $0.262 \pm 0.003$ | $0.071 \pm 0.001$ | $0.087 \pm 0.002$ | $0.092 \pm 0.001$ |
| SG-UCT$_{\text{BE-40}}$ | $0.286 \pm 0.002$ | $0.304 \pm 0.002$ | $0.297 \pm 0.001$ | $0.088 \pm 0.001$ | $0.108 \pm 0.001$ | $0.115 \pm 0.001$ |
| SG-UCT$_{\text{BECS-40}}$ | $0.290 \pm 0.001$ | $0.316 \pm 0.001$ | $0.319 \pm 0.002$ | $0.097 \pm 0.001$ | $0.115 \pm 0.001$ | $0.121 \pm 0.001$ |
| SG-UCT$_{\text{AE-40}}$ | $0.286 \pm 0.002$ | $0.302 \pm 0.003$ | $0.297 \pm 0.002$ | $0.088 \pm 0.001$ | $0.103 \pm 0.001$ | $0.114 \pm 0.001$ |
| SG-UCT$_{\text{AECS-40}}$ | $0.289 \pm 0.001$ | $0.317 \pm 0.001$ | $0.319 \pm 0.002$ | $0.098 \pm 0.001$ | $0.117 \pm 0.001$ | $0.126 \pm 0.001$ |

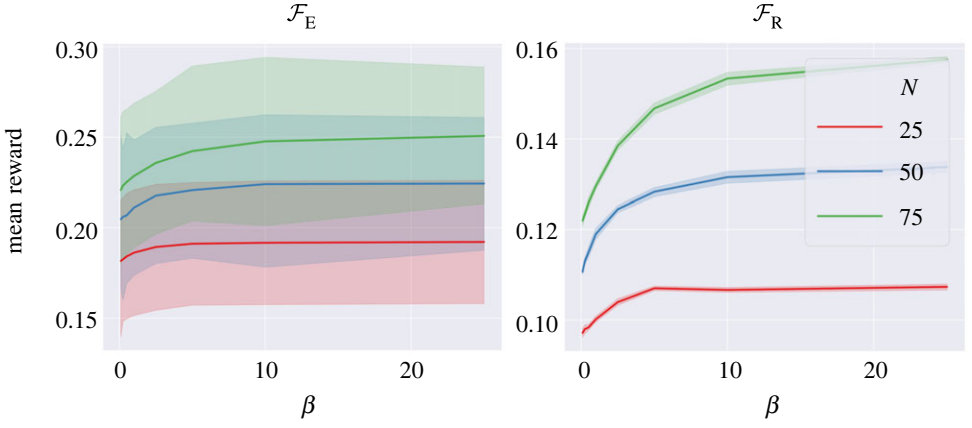


**Figure 4.** Average reward for SG-UCT$_{\text{MINCOST}}$ as a function of $\beta$, which suggests a bias towards low-cost edges is beneficial.

in figure 5 analyses a different metric for the Greedy and UCT variants; namely, the total and mean wall clock time per search step, as well as the total and mean number of objective function evaluations. As shown in the third panel, it is indeed the case that the cost-sensitive searches typically perform more objective function evaluations, an aspect that is reflected in their total wall clock timings (first panel). Furthermore, the UCT variants are noticeably slower on the smaller graphs, which is explained by the increased number of simulations compared to the greedy searches. Nevertheless, an advantage of UCT is that the user is able to use a simulation budget that suits their requirements, different from the $20N$ simulations used throughout our experiments. However, on the largest graphs, the greedy approaches begin performing more objective evaluations per step (fourth panel), and hence the wall clock timings per search step

**Table 5.** Representative wall clock time for the algorithms measured in hours, minutes and seconds on real-world graphs.

| $\mathcal{F}$ | G | Graph | Random | Greedy | GreedyCS | LDP | FV | ERes | MinCost | LBHB | UCT | SG-UCT (ours) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\mathcal{F}_E$ | Internet | Colt | <00.01 | 0.20.50 | 1.19.05 | — | — | — | 0.00.02 | 0.00.02 | 0.51.10 | 1.30.15 |
| | | GtsCe | <00.01 | 0.18.06 | 0.37.23 | — | — | — | 0.00.01 | 0.00.01 | 0.33.35 | 1.01.22 |
| | | TataNId | <00.01 | 0.11.26 | 0.34.25 | — | — | — | 0.00.01 | 0.00.01 | 0.42.12 | 0.53.47 |
| | | UsCarrier | <00.01 | 0.06.08 | 0.14.45 | — | — | — | <00.01 | <00.01 | 0.26.20 | 0.38.11 |
| | Metro | Barcelona | <00.01 | 0.03.22 | 0.07.11 | — | — | — | <00.01 | 0.00.01 | 0.23.05 | 0.26.45 |
| | | Beijing | <00.01 | 0.01.52 | 0.04.09 | — | — | — | <00.01 | <00.01 | 0.15.32 | 0.18.44 |
| | | Mexico | <00.01 | 0.01.58 | 0.03.08 | — | — | — | <00.01 | 0.00.02 | 0.16.44 | 0.24.08 |
| | | Moscow | <00.01 | 0.03.00 | 0.05.44 | — | — | — | <00.01 | 0.00.01 | 0.15.03 | 0.22.57 |
| | | Osaka | <00.01 | 0.01.38 | 0.02.32 | — | — | — | <00.01 | <00.01 | 0.13.23 | 0.14.49 |
| $\mathcal{F}_R$ | Internet | Colt | <00.01 | 0.11.39 | 2.35.10 | <00.01 | <00.01 | <00.01 | 0.00.03 | — | 0.37.19 | 1.27.04 |
| | | GtsCe | <00.01 | 0.08.16 | 1.11.32 | <00.01 | <00.01 | <00.01 | 0.00.01 | — | 0.30.51 | 1.43.39 |
| | | TataNId | <00.01 | 0.07.24 | 1.06.12 | <00.01 | <00.01 | <00.01 | 0.00.01 | — | 1.05.55 | 1.41.43 |
| | | UsCarrier | <00.01 | 0.06.50 | 0.37.12 | <00.01 | <00.01 | <00.01 | <00.01 | — | 0.31.19 | 1.11.30 |
| | Metro | Barcelona | <00.01 | 0.04.00 | 0.16.59 | <00.01 | <00.01 | <00.01 | <00.01 | — | 0.30.40 | 0.50.20 |
| | | Beijing | <00.01 | 0.01.03 | 0.09.55 | <00.01 | <00.01 | <00.01 | <00.01 | — | 0.26.54 | 0.33.05 |
| | | Mexico | <00.01 | 0.02.53 | 0.09.08 | <00.01 | <00.01 | <00.01 | <00.01 | — | 0.31.12 | 0.59.55 |
| | | Moscow | <00.01 | 0.03.23 | 0.13.37 | <00.01 | <00.01 | <00.01 | <00.01 | — | 0.30.11 | 0.31.49 |
| | | Osaka | <00.01 | 0.01.42 | 0.06.30 | <00.01 | <00.01 | <00.01 | <00.01 | — | 0.14.46 | 0.22.22 |

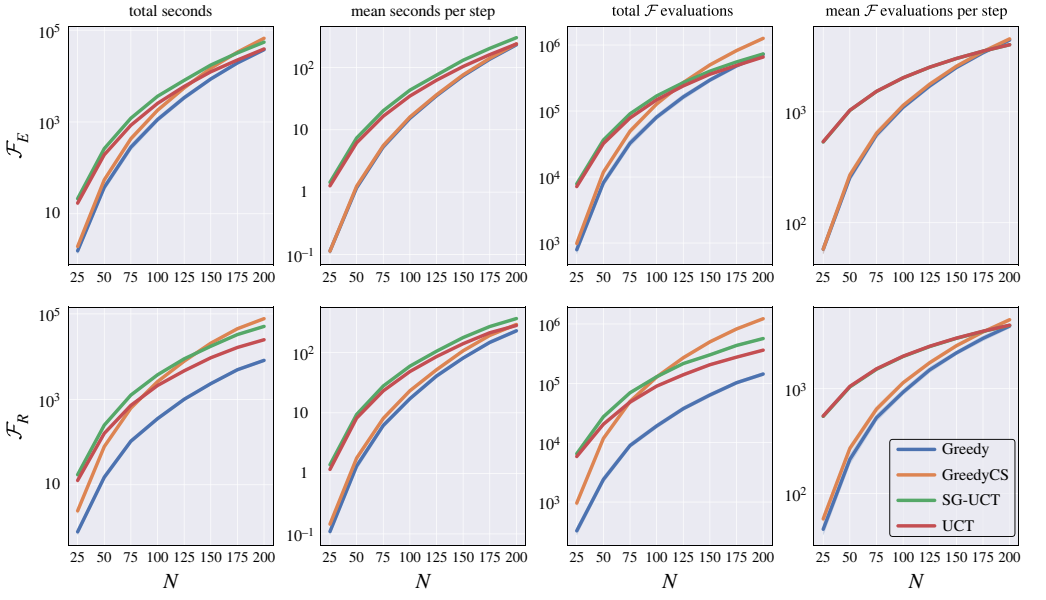


**Figure 5.** Wall clock time (leftmost two columns) and number of objective function evaluations (rightmost two) used by the different algorithms on synthetic graphs.

are also smaller (second panel), becoming nearly identical on the largest of graphs. This is due to the fact that the Greedy approaches need to consider $O(|V|)^2$ actions at each step compared to the $O(|V|)$ required by UCT and SG-UCT, and this difference begins to manifest at larger scales.

Let us now revisit the claim in §1 regarding superior scalability with respect to prior reinforcement learning methods, which require training a model before it can be used to improve a particular network. Namely, Darvariu *et al*. [12] reports a wall clock time that is equivalent to 56 h of a single core of a comparable CPU to train a model on graphs of size $N = 20$ and, due to the complexity of the problem, does not train models directly beyond graphs with $N = 50$. By contrast, on similar computational infrastructure, our proposed SG-UCT requires 11 h on average to optimize a much larger graph with $N = 200$ nodes. Hence, in cases where we are interested in improving a particular network, our proposed method yields an important improvement in scalability by sidestepping the cost of model training altogether.

Given the timings above, we expect our method to be applicable out-of-the-box to graphs with several hundreds of nodes. Apart from code-level optimizations, it is possible to speed up the computation of the objective functions by exploiting their incremental structure (see footnote 2), using cheaper proxy quantities (i.e. the various spectral indicators of resilience, which can circumvent needing to run simulations), or learning an approximate model of such global processes. Root and leaf parallelization in MCTS [24] can also be employed to speed up individual runs. Beyond this scale, it may be necessary to consider different levels of abstraction (for example, by treating the problem hierarchically). Due to the inherent difficulty of combinatorial optimization problems, current solutions are generally limited to this scale.

We also remark that there naturally exists a trade-off between the computational cost of the methods and their ability to improve the values of the given objective function. Namely, heuristics can be evaluated very quickly but yield smaller improvements compared to the search-based methods, which require more computational time. However, the type of infrastructure networks (such as road, rail or wired communication networks) motivating our work are very expensive to build in practice. We argue that finding a solution as close as possible to optimality is worthwhile, since the additional simulation cost would be negligible compared to actions in the real world, such as laying down roads or tracks characterized by sub-optimal layouts.

## 5. Discussion

Given our formulation of goal-directed graph construction in spatial networks as a combinatorial optimization problem, we note that other generic optimization algorithms can in principle be used. The Greedy approaches used for comparison are local searches over a horizon of two actions, in effect constructing a shallow search tree in which the objective function is evaluated for all the leaf nodes. Approaches such as branch-and-bound share some similarities since they also construct a tree of the solution space. However, to the best of our knowledge, no meaningful bounding criteria exist for the considered objectives. In such cases, branch-and-bound degenerates to an exhaustive search, which cannot be performed beyond the smallest of graphs.

Our work is also related to a large body of prior work in network design and optimization [61]. Such works formulate the problem under consideration as a mathematical program on a case-by-case basis, for which known combinatorial solvers can be applied. By contrast, our method is generic and makes no assumptions about the objective at hand. In principle, any objective function defined on a spatial graph is admissible, allowing our method to optimize for complex, nonlinear objectives that may arise in the real world (the resilience of the network to targeted attacks is one example of such a nonlinear objective). Furthermore, our method can be applied to objectives for which no solutions are currently known without the expertise required to cast the problem as a mathematical program. Necessarily, the generic nature of our method means that it may perform worse on certain specific problems with linear constraints and objectives. The trade-offs arising from the use of generic machine learning and decision-making algorithms is a topic of ongoing debate in the combinatorial optimization community [62], and we view these classes of methods as complementary.

## 6. Conclusion

In this work, we have addressed the problem of spatial graph construction: namely, given an initial spatial graph, a budget defined in terms of edge lengths and a global objective, finding a set of edges to be added to the graph such that the value of the objective is maximized. For the first time among related works, we have formulated this task as a deterministic MDP that accounts for how the spatial geometry influences the connections and organizational principles of real-world networks. Building on the UCT framework, we have considered several aspects that characterize this problem space and proposed the SG-UCT algorithm to address them.

Our evaluation results show performance substantially better than UCT (24% on average and up to 54% in terms of a robustness measure) and all the existing baselines taken into consideration, while requiring a computational budget similar to other search-based methods. More broadly, our approach brings an important improvement in scalability with respect to recent work in deep reinforcement learning for goal-directed graph construction, both in terms of network size that the method can operate on as well as reduced computational cost. We hope that our work will be used by infrastructure designers and urban planners as a basis for the design of more cost-effective infrastructure systems such as communication, power, transportation and water distribution networks.

Data accessibility. The Internet dataset is publicly available without any restrictions and can be downloaded via the Internet Topology Zoo website, www.topology-zoo.org/dataset.html. The *Metro* dataset was originally used in Roth *et al.* [59], and was licensed to us by the authors for the purposes of this work. A copy of the Metro dataset can be obtained by others by contacting its original authors for licensing (see www.quanturb.com/data). The data are provided in the electronic supplementary material [63].

Authors' contributions. V.-A.D.: conceptualization, data curation, investigation, methodology, project administration, resources, software, validation, visualization, writing—original draft, writing—review and editing; S.H.: conceptualization, funding acquisition, methodology, supervision, writing—review and editing; M.M.: conceptualization, funding acquisition, methodology, supervision, writing—review and editing.

All authors gave final approval for publication and agreed to be held accountable for the work performed therein.

Conflict of interest declaration. We declare we have no competing interests.
Funding. This work was supported by The Alan Turing Institute under the UK EPSRC grant no. EP/N510129/1.

## References

1. Scarselli F, Gori M, Tsoi AC, Hagenbuchner M, Monfardini G. 2009 The graph neural network model. *IEEE Trans. Neural Netw.* **20**, 61–80. (doi:10.1109/TNN.2008.2005605)
2. 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)
3. Vinyals O, Fortunato M, Jaitly N. 2015 Pointer networks. In *Proc. of the Twenty-ninth Conf. on Neural Information Processing Systems (NeurIPS 2015), Montréal, Canada, 7–12 December 2015*. Red Hook, NY: Curran Associates.
4. Bello I, Pham H, Le QV, Norouzi M, Bengio S. 2016 Neural combinatorial optimization with reinforcement learning. (http://arxiv.org/abs/1611.09940)
5. Khalil E, Dai H, Zhang Y, Dilkina B, Song L. 2017 Learning combinatorial optimization algorithms over graphs. In *Proc. of the Thirty-first Conf. on Neural Information Processing Systems (NeurIPS 2017), Long Beach, CA, USA, 4–9 December 2017*. Red Hook, NY: Curran Associates.
6. Monti F, Bronstein MM, Bresson X. 2017 Geometric matrix completion with recurrent multi-graph neural networks. In *Proc. of the Thirty-fourth Int. Conf. on Machine Learning (ICML 2017), Sydney, Australia, 6–11 August 2017*. PMLR.
7. Ying R, He R, Chen K, Eksombatchai P, Hamilton WL, Leskovec J. 2018 Graph convolutional neural networks for web-scale recommender systems. In *Proc. of the 24th SIGKDD Conf. on Knowledge Discovery and Data Mining (KDD 2018), London, UK, 19–23 August 2018*. New York, NY: ACM.
8. Gilmer J, Schoenholz SS, Riley PF, Vinyals O, Dahl GE. 2017 Neural message passing for quantum chemistry. In *Proc. of the Thirty-fourth Int. Conf. on Machine Learning (ICML 2017), Sydney, Australia, 6–11 August 2017*. PMLR.
9. Jin W, Barzilay R, Jaakkola T. 2018 Junction tree variational autoencoder for molecular graph generation. In *Proc. of the Thirty-fifth Int. Conf. on Machine Learning (ICML 2018), Stockholm, Sweden, 10–15 July 2018*. PMLR.
10. 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 Thirty-second Conf. on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada, 2–8 December 2018*. Red Hook, NY: Curran Associates.
11. Bradshaw J, Paige B, Kusner MJ, Segler MHS, Hernández-Lobato JM. 2019 A model to search for synthesizable molecules. In *Proc. of the Thirty-third Conf. on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada, 8–14 December 2019*. Red Hook, NY: Curran Associates.
12. Darvariu VA, Hailes S, Musolesi M. 2021 Goal-directed graph construction using reinforcement learning. *Proc. R. Soc. A* **477**, 20210168. (doi:10.1098/rspa.2021.0168)
13. 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 Thirty-fifth Int. Conf. on Machine Learning (ICML 2018), Stockholm, Sweden, 10–15 July 2018*. PMLR.
14. 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)
15. 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)
16. Wang H, Van Mieghem P. 2008 Algebraic connectivity optimization via link addition. In *Proc. of the Third Int. ICST Conf. on Bio-Inspired Models of Network, Information, and Computing Systems (BIONETICS 2008), Hyogo, Japan, 25–28 November 2008*. Brussels: ICST.
17. 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)
18. Gastner MT, Newman MEJ. 2006 The spatial structure of networks. *Eur. Phys. J. B* **49**, 247–252. (doi:10.1140/epjb/e2006-00046-8)
19. Barthélemy M. 2011 Spatial networks. *Phys. Rep.* **499**, 1–101. (doi:10.1016/j.physrep.2010.11.002)

20. Latora V, Marchiori M. 2001 Efficient behavior of small-world networks. *Phys. Rev. Lett.* **87**, 198701. (doi:10.1103/PhysRevLett.87.198701)
21. Kocsis L, Szepesvári C. 2006 Bandit based Monte-Carlo planning. In *Proc. of the Seventeenth European Conf. on Machine Learning (ECML 2006), Berlin, Germany, 18–22 September 2006*. Berlin/Heidelberg, Germany: Springer-Verlag.
22. Russell SJ, Norvig P. 2010 *Artificial intelligence: a modern approach*, 3rd edn. Englewood Cliffs, NJ: Prentice Hall.
23. Sutton RS, Barto AG. 2018 *Reinforcement learning: an introduction*. New York, NY: MIT Press.
24. Browne CB *et al.* 2012 A survey of Monte Carlo Tree Search methods. *IEEE Trans. Comput. Intell. AI Games* **4**, 1–43. (doi:10.1109/TCIAIG.2012.2186810)
25. Auer P, Cesa-Bianchi N, Fischer P. 2002 Finite-time analysis of the Multiarmed Bandit problem. *Mach. Learn.* **47**, 235–256. (doi:10.1023/A:1013689704352)
26. Rosin CD. 2011 Nested Rollout policy adaptation for Monte Carlo tree search. In *Proc. of the Twenty-second Int. Joint Conf. on Artificial Intelligence (IJCAI 2011), Barcelona, Catalonia, Spain, 16–22 July 2011*. Palo Alto, CA: AAAI Press.
27. Nash J. 1952 Some games and machines for playing them. Technical Report D-1164 Rand Corporation.
28. Anthony T, Tian Z, Barber D. 2017 Thinking fast and slow with deep learning and tree search. In *Proc. of the Thirty-first Conf. on Neural Information Processing Systems (NeurIPS 2017), Long Beach, CA, 4–9 December 2017*. Red Hook, NY: Curran Associates.
29. Gelly S, Silver D. 2007 Combining online and offline knowledge in UCT. In *Proc. of the Twenty-fourth Int. Conf. on Machine Learning (ICML 2007), Corvallis, OR, USA, June 20–24 2007*. New York, NY: ACM.
30. Silver D. 2007 Reinforcement learning of local shape in the game of go. In *Proc. of the Twentieth Int. Joint Conf. on Artificial Intelligence (IJCAI 2007), Hyderabad, India, 6–12 January 2007*. Palo Alto, CA: AAAI Press.
31. Guo X, Singh S, Lee H, Lewis RL, Wang X. 2014 Deep learning for real-time Atari game play using offline Monte-Carlo tree search planning. In *Proc. of the Twenty-eighth Conf. on Neural Information Processing Systems (NeurIPS 2014), Montréal, Canada, 8–13 December 2014*. Red Hook, NY: Curran Associates.
32. Silver D *et al.* 2016 Mastering the game of go with deep neural networks and tree search. *Nature* **529**, 484–489. (doi:10.1038/nature16961)
33. Silver D *et al.* 2018 A general reinforcement learning algorithm that masters Chess, Shogi, and Go through Self-Play. *Science* **362**, 1140–1144. (doi:10.1126/science.aar6404)
34. Laterre A, Fu Y, Jabri MK, Cohen AS, Kas D, Hajjar K, Dahl TS, Kerkeni A, Beguir K. 2018 Ranked reward: enabling self-play reinforcement learning for combinatorial optimization. (http://arxiv.org/abs/1807.01672)
35. Böther M, Kißig O, Taraz M, Cohen S, Seidel K, Friedrich T. 2022 What's wrong with deep learning in tree search for combinatorial optimization. In *Proc. of the Tenth Int. Conf. on Learning Representations (ICLR 2022), Virtual Event, 25–29 April 2022*. OpenReview.
36. Demetrescu C, Italiano GF. 2004 A new approach to dynamic all pairs shortest paths. *J. ACM (JACM)* **51**, 968–992. (doi:10.1145/1039488.1039492)
37. Albert R, Jeong H, Barabási AL. 2000 Error and attack tolerance of complex networks. *Nature* **406**, 378–382. (doi:10.1038/35019019)
38. Callaway DS, Newman MEJ, Strogatz SH, Watts DJ. 2000 Network robustness and fragility: percolation on random graphs. *Phys. Rev. Lett.* **85**, 5468–5471. (doi:10.1103/PhysRevLett.85.5468)
39. Gastner MT, Newman MEJ. 2006 Shape and efficiency in spatial distribution networks. *J. Stat. Mech: Theory Exp.* **2006**, P01015. (doi:10.1088/1742-5468/2006/01/P01015)
40. Bullmore E, Sporns O. 2012 The economy of brain network organization. *Nat. Rev. Neurosci.* **13**, 336–349. (doi:10.1038/nrn3214)
41. Wu ZX, Holme P. 2011 Onion structure and network robustness. *Phys. Rev. E* **84**, 026106. (doi:10.1103/PhysRevE.84.026106)
42. Dilkina B, Lai KJ, Gomes CP. 2011 Upgrading shortest paths in networks. In *Proc. of the Eighth Int. Conf. on Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming (CPAIOR 2011), Berlin, Germany, 23–27 May 2011*. Berlin/Heidelberg, Germany: Springer-Verlag.

43. Peeta S, Salman FS, Gunnec D, Viswanath K. 2010 Pre-disaster investment decisions for strengthening a highway network. *Comput. Oper. Res.* **37**, 1708–1719. (doi:10.1016/j.cor.2009.12.006)
44. Wu X, Sheldon D, Zilberstein S. 2016 Optimizing resilience in large scale networks. In *Proc. of the Thirtieth AAAI Conf. on Artificial Intelligence (AAAI 2016), Pheonix, AZ, USA, 12–17 February 2016*. Palo Alto, CA: AAAI Press.
45. Mansour Y, Peleg D. 1994 An approximation algorithm for minimum cost network design, Technical Report CS94-22. *Weizmann Institute of Science, Faculty of Mathematical Sciences*.
46. Hassin Y, Peleg D. 2000 Sparse communication networks and efficient routing in the plane (Extended Abstract). In *Proc. of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing (PODC 2000), Portland, OR, USA, 16–19 July 2000*. New York, NY: ACM.
47. Cazenave T. 2007 Reflexive Monte-Carlo search. In *Proc. of the Computer Games Workshop 2007, Amsterdam, The Netherlands, 15–17 June 2007*. Maastricht: Maastricht University.
48. Cazenave T. 2009 Nested Monte-Carlo search. In *Proc. of the Twenty-First Int. Joint Conf. on Artificial Intelligence (IJCAI 2009), Pasadena, CA, USA, 11–17 July 2009*. Palo Alto, CA: AAAI Press.
49. He J, Ostendorf M, He X, Chen J, Gao J, Li L, Deng L. 2016 Deep reinforcement learning with a combinatorial action space for predicting popular reddit threads. In *Proc. of the 2016 Conf. on Empirical Methods in Natural Language Processing (EMNLP 2016), Austin, TX, 1–5 November 2016*. Stroudsburg, PA: Association for Computational Linguistics.
50. Dulac-Arnold G *et al.* 2015 Deep reinforcement learning in large discrete action spaces. In *Proc. of the Thirty-second Int. Conf. on Machine Learning (ICML 2015), Lille, France, 6–11 July 2015*. PMLR.
51. Zahavy T, Haroush M, Merlis N, Mankowitz DJ, Mannor S. 2018 Learn what not to learn: action elimination with deep reinforcement learning. In *Proc. of the Thirty-second Conf. on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada, 2–8 December 2018*. Red Hook, NY: Curran Associates.
52. Chaslot GMJB, Winands MHM, Herik HJVD, Uiterwijk JWHM, Bouzy B. 2008 Progressive strategies for Monte-Carlo tree search. *N. Math. Natural Comput.* **4**, 343–357. (doi:10.1142/S1793005708001094)
53. Pinto J, Fern A. 2017 Learning partial policies to speedup MDP tree search via reduction to IID learning. *J. Mach. Learn. Res.* **18**, 2179–2213.
54. Bertagnolli G, Gallotti R, De Domenico M. 2021 Quantifying efficient information exchange in real network flows. *Commun. Phys.* **4**, 1–10. (doi:10.1038/s42005-021-00612-5)
55. Kaiser M, Hilgetag CC. 2004 Spatial growth of real-world networks. *Phys. Rev. E* **69**, 036103. (doi:10.1103/PhysRevE.69.036103)
56. Waxman B. 1988 Routing of multipoint connections. *IEEE J. Sel. Areas Commun.* **6**, 1617–1622. (doi:10.1109/49.12889)
57. Dall J, Christensen M. 2002 Random geometric graphs. *Phys. Rev. E* **66**, 016121. (doi:10.1103/PhysRevE.66.016121)
58. Knight S, Nguyen HX, Falkner N, Bowden R, Roughan M. 2011 The internet topology zoo. *IEEE J. Sel. Areas Commun.* **29**, 1765–1775. (doi:10.1109/JSAC.2011.111002)
59. Roth C, Kang SM, Batty M, Barthelemy M. 2012 A long-time limit for world subway networks. *J. R. Soc. Interface* **9**, 2540–2550. (doi:10.1098/rsif.2012.0259)
60. Fiedler M. 1973 Algebraic connectivity of graphs. *Czechoslovak Math. J.* **23**, 298–305. (doi:10.21136/CMJ.1973.101168)
61. Ahuja RK, Magnanti TL, Orlin JB, Reddy MR. 1995 Chapter 1 Applications of network optimization. In *Handbooks in operations research and management science*, vol. 7, *Network Models*, pp. 1–83. Elsevier.
62. 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)
63. Darvariu V-A, Hailes S, Musolesi M. 2023 Planning spatial networks with Monte Carlo tree search. Figshare. (doi:10.6084/m9.figshare.c.6350517)