Surrogate-guided optimization in quantum networks

Prielinger, Luise; Iñesta, Álvaro G.; Vardoyan, Gayane

DOI
10.1038/s41534-025-01048-3

Publication date
2025

Document Version
Final published version

Published in
NPJ Quantum Information

npj | quantum information

Article

Published in partnership with The University of New South Wales

https://doi.org/10.1038/s41534-025-01048-3

Surrogate-guided optimization in quantum networks

Luise Prielinger [1,2] ✉, Álvaro G. Iñesta [1,2] & Gayane Vardoyan [1,2,3]

When physical architectures become too complex for analytical study, numerical simulation proves essential to investigate quantum network behavior. Although highly informative, these simulations involve intricate numerical functions without known analytical forms, making traditional optimization techniques that assume continuity, differentiability, or convexity inapplicable. We introduce a more efficient computational framework that employs machine learning models as surrogates for the objective function. We demonstrate the effectiveness of our approach by applying it to three well-known optimization problems in quantum networking: allocating quantum memory across multiple nodes, tuning an experimental parameter in every physical link of a quantum entanglement switch, and finding effective protocol configurations in a large asymmetric quantum network. Our algorithm consistently outperforms Simulated Annealing and Bayesian optimization within the allotted time, improving results by up to 29% and 28%, respectively. Our framework will thus allow for more comprehensive quantum network studies, integrating surrogate-assisted optimization with existing quantum network simulators.

Quantum network infrastructure has the potential to be integrated with today's Internet, serving entangled states to users who request them[1]. These networks will enable a number of applications provably beyond the capabilities of classical technologies alone. Examples include verifiably secure communication[2,3], advanced sensing tasks[4,5] and blind quantum computation[6,7], among others. For quantum networks to achieve their intended potential, diverse solutions for all layers of the system stack have been put forward in recent years. At the physical layer, for instance, so-called quantum repeaters have been proposed to assist with quantum information transport over long distances (see e.g., ref. 8 for an in-depth review on quantum repeaters). At the software layer, the primary task involves creating efficient protocols that coordinate and control the intricate physical processes within the quantum network system stack[9]. Extensive studies[10–17] have been conducted on both software and hardware components of quantum networks, utilizing analytical tools and numerical simulations to find feasible architectures and bring quantum network technology a step closer to the real world. These efforts include the use of optimization and machine learning techniques, which have been used, for example, to design new entanglement distribution protocols[18–20]. While these studies are greatly informative, they generally assess only small-sized networks or operate under simplified assumptions, such as ideal hardware models or highly symmetric network typologies.

In this study, our primary goal is to utilize a surrogate-assisted optimization workflow to discover effective protocol and hardware parameter values under realistic conditions. To this end, we extend previous efforts and integrate comprehensive numerical simulators, such as NetSquid[21] and SeQUeNCe[22] in our framework. In essence, a surrogate model[23] is an approximation of a given objective function, which is built using data obtained from evaluations of the latter. Importantly, instead of directly optimizing the computationally expensive objective function—a common approach in global optimization techniques such as Simulated Annealing or genetic algorithms (see ref. 24 for a review of numerical optimization methods)—the surrogate model guides the iterative optimization process.

Common surrogate models like Gaussian processes[25] (used in Bayesian optimization) and neural networks[26] are based on complex theoretical frameworks. Gaussian processes, for example, depend significantly on the choice of a kernel function and they are known to be most effective in smaller, continuous search spaces with fewer than 20 variables[27]. Neural networks, which consist of various specialized layers[28], require substantial data for optimal performance. Additionally, the outputs from these complex models are often more difficult to interpret compared to those from simpler models[29]. As a consequence, we opted for two well-known but less complex models—Support Vector Regression (SVR)[30] and Random Forest Regression (RF)[31]—due to their explainability, computational efficiency, and straightforward evaluation. Surrogate models have been utilized since the late 80s and ever since applied to various scientific domains, such as chemical engineering[32,33] and material science[34]. In quantum sensing, Bayesian optimization has been utilized for quantum detectors[35] and in quantum

[1]QuTech, Delft University of Technolgy, Delft, The Netherlands. [2]EEMCS, Delft University of Technology, Delft, The Netherlands. [3]CICS, University of Massachusetts, Amherst, MA, USA. ✉e-mail: l.p.prielinger@tudelft.nl

networking to calibrate experimental parameters[36]. To the best of our knowledge, surrogates have not yet been applied to any software component of quantum communication.

We apply our framework to three quantum network use cases simulated in NetSquid[21], SeQUeNCe[22] and OptimizingCD[37], respectively. Our contributions can be summarized as follows:

- *Optimization integrating scientific software*: We integrate detailed simulations using established scientific software in our investigated use cases. This approach allows us to avoid reliance on purely theoretical predictions, which are often specific to certain architectural or software stack characteristics like node connectivity or expected protocol behavior[13,18,22].
- *Handling of many parameters*: The scenarios we test involve up to 100 variables. For the largest variable set, reference methods perform comparably to random search, whereas our approach significantly outperforms both. This capability is largely due to the simplicity of the SVR and RF models utilized, which exhibit only linear time complexity with the number of variables[38].
- *Addressing multiple objectives*: In a quantum network where different parties might have their individual quality-of-service goals, our approach allows to determine an empirical estimate of the Pareto frontier. Using this solution set, we can assess the effectiveness of parameter settings in the context of multi-objective optimization.
- *Diverse applicability*: The selected use cases present well-known optimization challenges of distributing entangled states in quantum networks. The solutions we find introduce competitive candidates to existing solutions found in literature. Furthermore, we test the applicability of our approach on a range of different entanglement distribution protocols, including on-demand and continuous-distribution protocols[37,39].We investigate relevant problem scenarios in quantum networking in order to showcase the versatility of our approach, which we term simply *Surrogate*. For this purpose, we evaluate the efficiency of *Surrogate* using different utility functions. In these detailed problem scenarios we compare to another model-based approach, termed Bayesian optimizer, which is developed by Meta (*Meta*)[40]. Furthermore we compare to a traditional global optimization algorithm termed *Simulated Annealing*[41] as well as to a uniform *Random Search* as baseline. In the investigated scenarios, our approach consistently outperforms the selected reference methods by up to 29% within the allotted time limits.

Results

Network utility

Consider a vector function $f(\mathbf{x}) : X \to \mathbb{R}^{m_f}$ of arbitrary size $m_f \in \mathbb{N}$ describing a performance metric of a quantum network (e.g., the average number of entangled states provided to a pair of nodes). The input to this function $\mathbf{x} = \{x_1, x_2, \ldots, x_N\}$ is a set of network parameters, which may include both fixed (non-configurable) as well as tunable features. A variable x_p of a network configuration $\mathbf{x}$ is defined on a domain X_p. Together, the parameter domains form the input space: $X = X_1 \times X_2 \times \cdots \times X_N$. A variable x_p can be of any datatype—for continuous and discrete values, X_p is confined within some minimum-maximum bounds, $x_p^{\min}$ and $x_p^{\max}$, respectively, while ordinal and categorical parameters are defined as value sets.

Based on f, we can formulate an objective function $U(f, \mathbf{x})$ reflecting the way in which a quantum network perceives *utility*[13]. Essentially, while $f(\mathbf{x})$ describes some output of the network, the function $U(f, \mathbf{x}) : \mathbb{R}^{m_f} \times X \to \mathbb{R}^m$, $m \in \mathbb{N}$, assesses how much utility can be derived from this output. Although U can in principle represent any general objective, in our analyzed use cases, one element of U, denoted $U^{(i)}$, consistently represents the utility perceived by a network user i. We will explore some relevant examples of utility functions U based on distillable entanglement[42], request completions[22], and virtual neighborhood size[37].

In a quantum network, many processes are inherently probabilistic. For example, creating entanglement between nodes often requires multiple attempts following a geometric distribution[8]. As a consequence, we assume f and thus also the utility $U(f, \mathbf{x})$ to be *stochastic* functions. We denote the utility perceived by user i with a random variable $U^{(i)}(f, \mathbf{x})$ with expectation value $\mathbb{E}[U^{(i)}]$. As $f(\mathbf{x})$ is evaluated numerically, we retrieve a sample $\{u_j^{(i)}\}_{j=1}^n$ via n simulation runs to estimate the expected behavior $\mathbb{E}[U^{(i)}]$ for each user i with the sample mean $\mathbb{E}[U^{(i)}] \approx \bar{U}^{(i)}(f, \mathbf{x}) = \frac{1}{n}\sum_{j=1}^n u_j^{(i)}$. In this work, our optimization objective is to maximize the aggregated utility of the network, i.e., the sum of all individual user utilities.

Definition I.1. (Utility maximization over configurable variables). We aim to maximize the aggregated utility over all users by configuring variable values within the tunable portion X_{conf} of the domain X, $X_{\text{conf}} \subseteq X$. This results in the objective $\max_{\mathbf{s}} \sum_{i=1}^m \bar{U}^{(i)}(f, \mathbf{s})$, where $\mathbf{s} \in X_{\text{conf}}$. Here, the set $X \backslash X_{\text{conf}}$ denotes all parameters which are fixed.

Surrogate-assisted search

We present our optimization framework depicted in Fig. 1, which progresses through successive cycles $t \in \{0, \ldots, T\}$. It begins by randomly generating k_0 initial input sets $\{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_{k_0}\}$ from the search domain X_{conf}, where each set is a unique configuration of parameter values. Then, for each input set $\mathbf{s}_i$, a quantum network simulation is run n times to produce mean utility outputs $[\bar{U}^{(1)}(f, \mathbf{s}_i), \cdots, \bar{U}^{(m)}(f, \mathbf{s}_i)]$. Each of the n runs lasts T_{sim} simulation time units. The input configurations together with the associated utility outputs form the initial *dataset*, concluding the first cycle, $t = 0$.

In the subsequent cycle, $t = 1$, the algorithm undertakes the first *acquisition process* which involves two stages: model training and model evaluation. Initially, the two machine learning models (SVR, RF) are evaluated using five-fold cross validation; thereby the models are trained on 80% of the dataset (the training set) and evaluated on the remaining data (the test set). Their performance is measured using the mean absolute error between predicted and actual values in the test set. The model with the lower error progresses to the next phase: model evaluation. To this end, *for each* $\mathbf{s}_i^{\text{top}}$ of l so far best performing configurations $\{\mathbf{s}_i^{\text{top}}\}_{i=1}^l$, a number N_t of points in the neighborhood $\{\mathbf{s}_{i,v}^{\text{top}}\}_{v=1}^{N_t}$ are selected. The latter is achieved via sampling from truncated normal distributions centered around each parameter value in $x_p \in \mathbf{s}_i^{\text{top}}$. Formally, parameter values are sampled from $\mathcal{N}_{\text{trunc}}(\mu_p, \sigma_p(t, d))$. These distributions have mean $\mu_p \equiv x_p$, standard deviation $\sigma_p(t, d) = \gamma(t, d)(x_p^{\max} - x_p^{\min})/2$, and are truncated at the bounds $x_p^{\min}$ and $x_p^{\max}$ (see Section "Network utility"). We gradually shift from exploration towards exploitation by progressively narrowing the standard deviation using a monotonically-decreasing transition function $\gamma(t, d)$ depending on the elapsed cycles t and an exploitation degree $d \geq 1$. This adjustment ensures that as each cycle progresses, the focus on exploring unknown points in X_{conf} turns to exploiting known good areas of the search space. Furthermore, the number of sampling points increases incrementally with each cycle t. This approach allows for less costly computation during early cycles, when the models' performance is lower, and progressively dedicating more time as the models improve and accumulate knowledge; see Supplementary Note 1 and Supplementary Note 2 for further details on exploration and acquisition strategies, respectively.

Once the points $\{\mathbf{s}_{i,v}^{\text{top}}\}_{v=1}^{N_t}$ are passed to a model, it predicts utility values based on its current knowledge, and the configuration associated with the *highest* utility is returned. This leads to l new configurations $\{\mathbf{s}_1^{\text{next}}, \ldots, \mathbf{s}_l^{\text{next}}\}$ to be evaluated by the simulation. Finally these parameter sets and associated simulation outputs are added to the dataset, $k_1 = k_0 + l$, which completes the first optimization cycle $t = 1$. This cycle repeats until a time limit or maximum number of iterations, $t = T$ is reached. By default, k_0 and l are chosen according to the number of compute resources available for parallel execution. Table 1 summarizes the framework parameters used.

Every algorithm comes with its limitations: heuristic approaches, like the one described, do not guarantee a globally optimal solution but instead provide approximations once a termination criterion is met[24]. This criterion

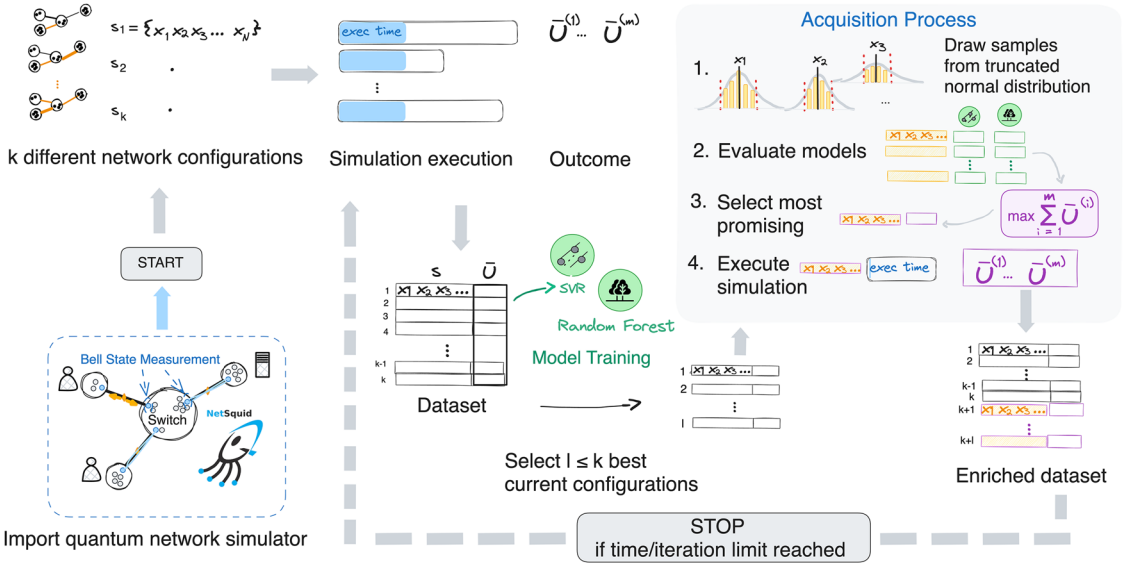

Fig. 1 | Simplified surrogate-assisted workflow. After importing the quantum network simulation, we generate k_0 different network configurations $\{\mathbf{s}_1, \ldots, \mathbf{s}_{k_0}\}$ from the search domain. Execution of the quantum network simulation using the latter yields the initial training data of parameter sets $\{\mathbf{s}_i\}$ along with their objective means $[\bar{U}^{(1)}(f, \mathbf{s}_i), \cdots, \bar{U}^{(m)}(f, \mathbf{s}_i)]$. Machine learning models train on the dataset and their performance is evaluated. In the acquisition process, the best model in the cycle predicts utility values for sampled configurations. Then, the parameter set associated with the highest predicted utility is passed to the simulation, executed, and the outcome appended to the training data. This cycle repeats until the optimization concludes upon reaching a wall-clock time or number of cycles T.

Table 1 | Parameters used in surrogate-assisted search

Symbol	Description
T	Optimization limit, can be specified as wall-clock time limit or maximum number of optimization cycles
t	Elapsed time $t \in [0, T]$ or cycle $t \in \{0, \ldots, T\}$
n	Number of simulation evaluations used to compute $\bar{U}$
l	Number of points explored in an optimization cycle
d	Exploitation degree (see Supplementary Note 1)
k_t	Number of points in the dataset at cycle t, where $k_{t+1} = k_t + l$ and $k_0 = l$
T_{sim}	Simulation time of the quantum network model (specific to use case)

is primarily dictated by the available computational resources. For example, considering the computationally demanding nature of our simulation functions, we set the termination criterion to not exceed $T = 100$ optimization cycles. Further, we must rely on seeds to guarantee reproducibility as both the simulation as well as the machine learning models used in the acquisition process are inherently stochastic. In addition, selecting settings for workflow parameters like the degree of exploitation d is crucial and requires understanding their specific function (see Supplementary Note 1). Finally, the numerical effort to train and evaluate machine learning models makes our approach naturally less suitable for problem instances that are analytically tractable.

In the following section we will investigate different optimization use cases in quantum networking and show applicability as well as demonstrate efficiency of the described approach.

Use cases

A quantum entanglement switch. A quantum entanglement switch (QES) is a canonical example of a simple quantum network[11,13,21] where a set of user nodes is connected to a central hub that distributes entanglement among them. We assume that each node is equipped with memory qubits and is connected to the QES via a fiber optic link to generate so-called link-level entangled states, henceforth referred to simply as *links*. In our setup, one node is designated as a server, while the remaining nodes operate as users, see Fig. 2. The QES' purpose is to create a direct entangled connection between the server and each of the users, which we term *end-to-end link*, in the form of bipartite entanglement. The QES accomplishes this using a process called entanglement swapping[8], during which quantum operations at the switch transform two links between a user and the server into an end-to-end link. After generation, the state is transferred to available memory qubits in each node's buffer of size B. When the buffer is full, the oldest state is discarded. The switch generates end-to-end links in this manner for five simulated seconds $T_{sim} = 5$s.

We assume that link-level entanglement generation uses the single-click scheme[43], wherein each physical link l has a midpoint station with a beamsplitter and two photon detectors. Nodes contain a communication qubit entangled with an emitted photon. When these photons arrive at the midpoint station, they are measured to generate an entangled state between the communication qubits. Under high photon losses, this scheme shows a known tradeoff between entanglement generation rate and the quality of the generated state (fidelity). This tradeoff stems from an experimental parameter, the so-called bright-state population α_l. A higher α_l increases photon emission but at the same time degrades entanglement fidelity. Consequently, the success probability $p_{gen,l}$ of entanglement generation on link l grows with α_l, while the fidelity F_l decreases as $F_l = 1 - \alpha_l$. We approximate this physical behavior at each physical link using NetSquid's depolarizing-error model. A depolarizing error is a type of quantum error that, with a certain probability $p_{\text{depol},l} \propto \alpha_l$, transforms the photon's polarization state into a completely mixed state. For further details on how we model the physical quantum states based on ref. 13 and ref. 21 we refer the reader to Supplementary Note 4.

Our goal is to balance the rate-fidelity tradeoff across all N physical links $\mathbf{s}_\alpha = \{\alpha_i\}_{i=1}^N$, in order to maximize utility served to all users. Following the approach of ref. 13, utility is assessed based on distillable entanglement, which quantifies how many degraded entangled pairs can be restored to a state useful for quantum communication tasks.

Definition I.2. (Utility based on distillable entanglement). A user's utility $U^{(i)}$ is based on the average end-to-end entanglement rate $R^{(i)}$ and fidelity $F^{(i)}$ of end-to-end states established with the server. To calculate a user's utility, the fidelity $F^{(i)}$ is passed to the so-called "yield of the hashing protocol" D_H

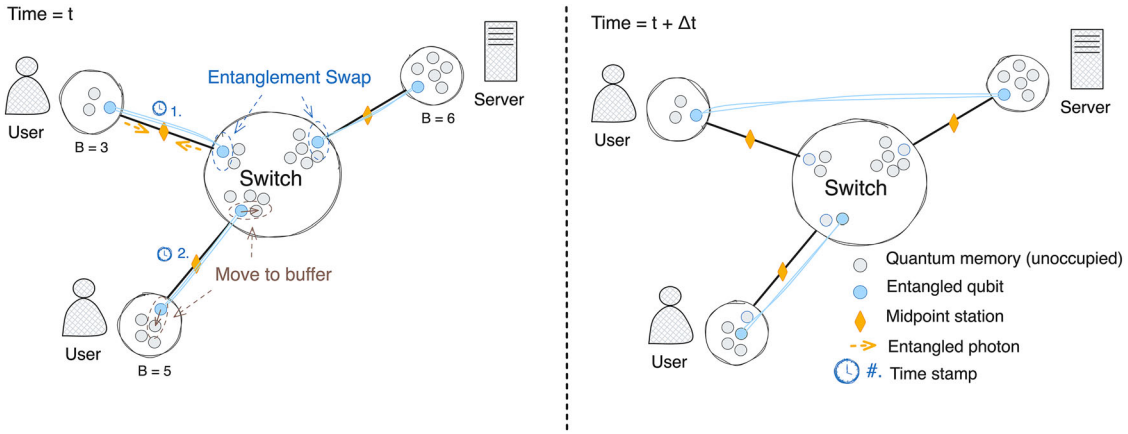

Fig. 2 | Quantum Entanglement Switch (QES) serving two users and a server. Each fiber link is equipped with a midpoint station (drawn as orange diamond) that measures photons entangled with communication qubits located at nodes. After an entangled state is generated, it is immediately transferred to a free memory qubit in the buffer (shown with brown arrows) along with a time stamp (small blue clocks). Should the buffer be full, the oldest state is discarded, and the memory receives the fresh link. The switch can only execute one entanglement swap at a time. By default, the user who holds the oldest link (smallest time stamp) is given priority for the swap.

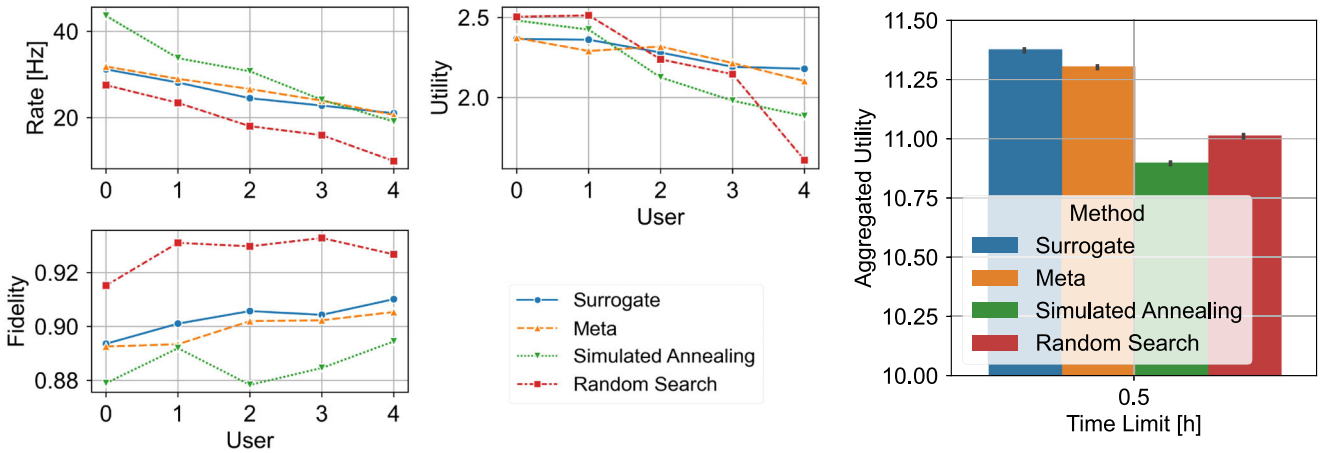

Fig. 3 | Utility maximization results for five users at varying distances from the switch. The left two panels show the average rate and fidelity at which each user receives end-to-end links with the server, as well as the resulting utility values based on distillable entanglement. Each marker presents the mean (and standard error) of $n_{exec} = 10^3$ simulation evaluations (standard error is smaller than marker size). The bars on the right present the aggregated utility over all users. Parameters used in the surrogate workflow: exploitation degree $d = 4$, maximum execution time $T = 30$ min, $n = 20$ simulation runs.

which serves as a lower bound on distillable entanglement[2]:

$$U^{(i)}(\mathbf{s_\alpha}) := \log\left(R^{(i)}(\mathbf{s_\alpha}) \cdot D_H(F^{(i)}(\mathbf{s_\alpha}))\right),$$
$$\text{with } D_H(F) := \max\left(1 + F\log_2 F + (1-F)\log_2 \tfrac{1-F}{3}, 0\right). \quad (1)$$

Using this definition, we can write our objective to maximize the sum of all user utility values $U^{(i)}$ as

$$\max_{\mathbf{s}_\alpha} \sum_{i=1}^{m} U^{(i)}(\mathbf{s}_\alpha), \quad (2)$$

where m denotes the number of users.

Although this problem can be solved analytically for a small number of users[13], the objective function can be non-convex, complicating the application of analytical or even algorithmic optimization techniques, such as gradient ascent. Using simulation in combination with surrogate optimization might thus provide a meaningful solution while mitigating these difficulties. We approximate average end-to-end fidelities $\bar{F}^{(i)}(f, \mathbf{s_\alpha})$ as well as average rates $\bar{R}^{(i)}(f, \mathbf{s_\alpha})$ using n runs of our simulation $f(\mathbf{s_\alpha})$ implemented with NetSquid. We then use these quantities to compute sample means of user utilities $\bar{U}^{(i)}(f, \mathbf{s_\alpha})$.

First, we apply our surrogate workflow to a small QES network serving two users and a server; we thereby recover a selection of results from the analytical study in ref. 13, described in Supplementary Note 4.

Next, we extend the setup to one involving five users located at different distances from the switch: 10, 20, 30, 40, and 50 km. The server is located close to the switch, at a fixed distance of 5 km. We conduct a comparison to reference methods (Section "Baselines") involving ten independent repetitions for each optimization method. From these repetitions, we select the solution that exhibits the best performance according to the objective function. A subsequent simulation of these solutions is then carried out with $n_{exec} = 10^3$ runs each. These additional runs ensure that the collected statistics (e.g., mean of aggregated utility) are sufficiently representative of the distributions that give rise to them. Figure 3a depicts the resulting fidelity $\bar{F}^{(i)}$ as well as the average rate $\bar{R}^{(i)}$ of end-to-end links per user i during the simulated time T_{sim}. Simulated Annealing and random search tend to find solutions with either relatively high rates but low fidelity, or low rates but high fidelity for each user. For example, consider the solution of simulated annealing: user 0 receives on average 42.6 links per second (link-generation rate) which sums up to $42.6[1/s] \times 5[s] \approx 213$ links in total per simulation execution. These states are generated on average

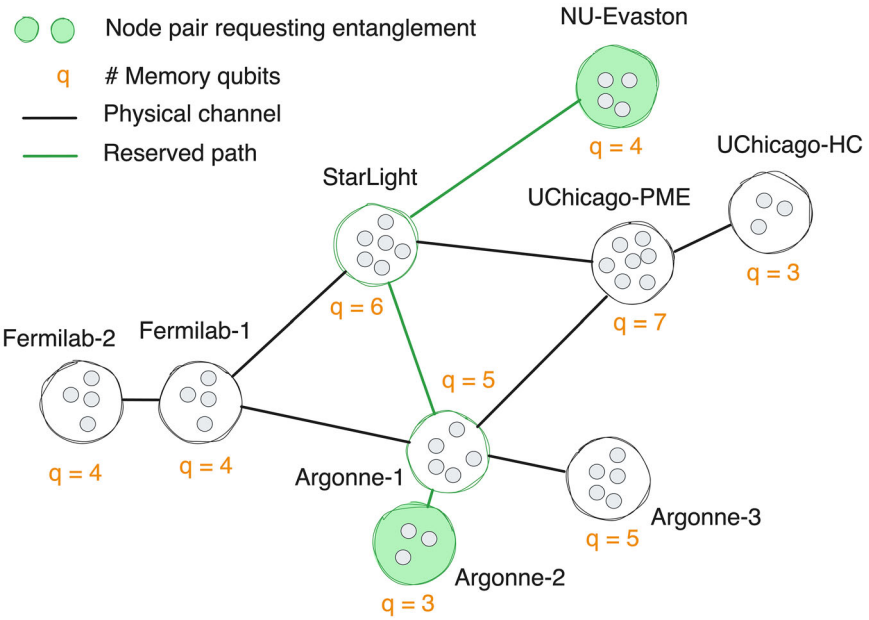

Fig. 4 | Illustration of the metropolitan quantum network studied in ref. 22. Each node i receives a number of memory qubits q_i according to an allocation $\mathbf{q}$ (example values in orange). All nodes, one after the other, request to share entanglement with a randomly chosen other node. The network system stack fulfills the requests sequentially until the simulation time limit T_{sim} is reached.

with a fidelity of 0.88, resulting in a utility of 2.5 for user zero. Doing the same computation for users 1–4 and aggregating results gives 10.9 overall utility, which is the lowest outcome of all methods. In contrast, both the Bayesian optimizer developed by Meta (*Meta*) and our surrogate-assisted approach (*Surrogate*) strike a more meaningful balance between rate and fidelity, resulting in higher aggregated utility, depicted in Fig. 3b. Overall, we observe slightly higher utility outcomes of the surrogate workflow compared to the reference methods: 0.5%, 4.3%, and 3.3% for Meta, Simulated Annealing, and random search, respectively. As expected, Meta and surrogate optimization show similar performance. This similarity arises because Bayesian optimization is a method well-suited for problems involving only continuous variables and fewer than 20 variables[27].

While here our algorithm does not provide a drastic improvement compared to baselines, the goal of this use case is to show that our surrogate workflow enables a straightforward transition from a simple and highly symmetric configuration to a more complex setup, which may no longer be amenable to analytical or exact solutions. In the next use case, we investigate a metropolitan-area quantum network involving a more comprehensive system stack and nine discrete hardware parameters to configure; this leads to a significant divergence between the Meta algorithm and our surrogate approach.

Memory allocation in a metropolitan quantum network. We study another previously investigated quantum network setup: a metropolitan network under construction in Chicago (United States), depicted in Fig. 4. The network we simulate is modeled using the software package SeQUeNCe[22]. All $m = 9$ nodes of the quantum network are users, each possessing at least ten memory qubits $q_i \geq 10$, which are modeled based on single erbium (Er) ions in crystal[44]. Nodes are capable of generating entanglement with their immediate neighbors using dedicated midpoint stations at each optical link. Nodes also have the ability to perform entanglement swapping and execute the BBPSSW purification protocol[45] to enhance fidelity. In this scenario, utility is measured by the network's ability to meet entanglement requests issued by user pairs. One user at a time selects another user uniformly at random and submits a request to its network manager. The network manager identifies the shortest route and announces the request to the nodes' network managers on the route. Each request specifies three key requirements: (1) the desired target fidelity of the end-to-end entangled state chosen uniformly at random between 0.8 and 1; (2) a number of memory qubits q in all nodes along the path, selected uniformly at random from $\{10, \ldots, \min(\frac{q_{\text{init}}}{2}, q_{\text{resp}})\}$, where q_{init} and q_{resp} are the number of qubits at the initiating and responding node, respectively, and (3) a required period between 1 and 2 s to consume the entangled state. If a request fails (for example, due to insufficient end-to-end fidelity) the request is attempted requesting a different randomly selected number of memories until success. Note, that even if all requests failed during simulation time T_{sim}, the outcome still provides valuable information to the optimization process. This outcome indicates that the chosen allocation is poor; and it naturally prompts the algorithm to explore more promising memory allocations. Nevertheless, allocating e.g., zero memories to all users is not meaningful as we already know the number of completed requests will be zero. To prevent such trivial cases, we impose the minimum number of memories per user ($q_i \geq 10$), as stated above. This initial condition ensures that the algorithm starts from a more realistic baseline.

Should a request complete successfully, another user submits. If all users have submitted and completed a request, the process starts anew. For a detailed explanation of the model used and its parameters, we direct the reader to the comprehensive description provided in ref. 22 and Supplementary Note 5. We simulate this process of submitting and fulfilling requests over $T_{\text{sim}} = 20$ s. Crucially, the number of requested memory qubits directly influences the potential for multiple rounds of entanglement purification, thereby increasing the likelihood of achieving the desired fidelity. Thus, similar to ref. 22, we aim to maximize the number of requests the network can serve by distributing a given memory budget of $b = 450$ memory qubits across the network's nodes.

Definition I.3. (Memory Allocation Problem). Let b be a budget of memory qubits and c_i the maximum number of memories node i can hold. The problem is to determine a memory allocation $\mathbf{q} = \{q_1, \ldots, q_m\}$ across all m nodes that maximizes the number of completed requests $U^{(i)}$ aggregated over all users $i \in \{1, \ldots, m\}$. We define the following maximization problem:

$$\max_{\mathbf{q}} \sum_{i=1}^{m} U^{(i)}(\mathbf{q}, b) - P(\mathbf{q}, b), \tag{3}$$

$$\text{where } P(\mathbf{q}, b) := \max\left(0, \sum_{i=1}^{m} q_i - b\right) \quad \text{with} \quad 0 \leq q_i \leq c_i \quad \forall i \in \{1, \ldots, m\}, \tag{4}$$

such that for memories exceeding the budget b, the function $P \geq 0$ adds a scalar penalty.

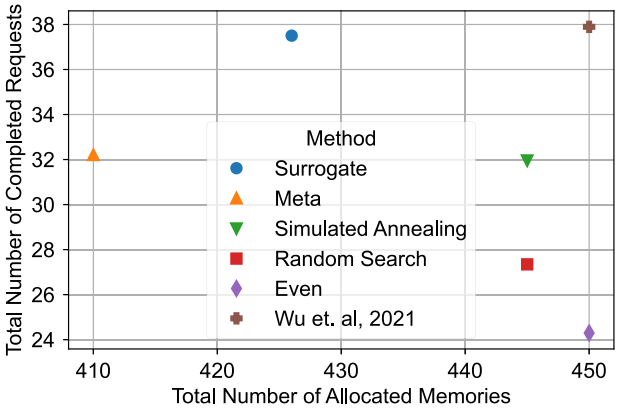

Fig. 5 | Sum of all completed requests over the total number of allocated quantum memories for solutions found by different methods. For the surrogate workflow, we use the following parameter settings: $T = 25$ h, $d = 6$, $n = 5$. Each marker represents the average of $n_{exec} = 10^3$ simulation runs with standard errors smaller than marker size.

As a remark, Definition I.3 is not unique, but one of several possible formulations. It presents a less restrictive optimization problem than the problem solved exhaustively in ref. 22. There, a fixed budget of 450 qubits in total is distributed across network nodes ($\sum_{i=1}^{m} q_i = b$), while we only induce a penalty if $\sum_{i=1}^{m} q_i > b$. Relaxing the fixed budget constraint to the penalty P is a different problem statement than the one solved in ref. 22. However, comparison to solutions of the fixed-budget problem remains valuable, as found solutions can provide insight into the tradeoff between resource (quantum memory) usage and gain in utility.

We utilize the surrogate workflow as well as reference methods (see Section "Baselines" for details), to find solutions to the memory allocation problem, where we compute $\bar{U}^{(i)}(f(\mathbf{q}), b)$ using the adapted network simulation $f(\mathbf{q})$ of ref. 22. Note that the execution time depends on the input values and takes on average 9 min on our system using $n = 5$ simulation runs to calculate sample averages. This leads to long execution times, bringing about $T = 25$ h to be a reasonable time limit for the optimization process. The best found solution from ten independent repetitions per method is presented in Supplementary Table III. Figure 5 displays the associated results: the average number of requests fulfilled, calculated from $n_{\text{exec}} = 10^3$ simulation runs, versus the total number of memories allocated. The solution found by our approach outperforms solutions found by Meta and Simulated Annealing by completing, on average, five additional requests (totaling 37.5). Further, it completes only 0.2 requests less when compared to the allocation derived from an exhaustive graph traversal algorithm, as detailed in ref. 22. Notably, the surrogate optimizer's solution conserves 27 memory qubits compared to this exhaustive method, amounting to 6% resource saving with a minor performance decrease of only 0.5%. The baseline from ref. 22, employing the *Even* allocation where each node possesses 50 memory qubits, exhibits the poorest performance. Using the Even allocation, the network completes 24.2 requests, three requests less on average than the solution found by searching uniformly at random.

With these results, we demonstrate that even though quantum network simulation can come with a heavy computational overhead, surrogates enable the discovery of solutions that are competitive to exhaustive search approaches, e.g., as used in ref. 22.

Continuous distribution of entanglement. Our final use case explores protocols for continuous entanglement distribution in quantum networks. Unlike on-demand protocols that require specific scheduling policies to coordinate node operations based on user demands, continuous-distribution protocols distribute entanglement nonstop throughout the network. This allows for immediate use of entanglement by any node as needed, supporting applications that continuously operate and consume entanglement in the background. In certain scenarios, continuous distribution protocols may be able to allocate resources faster and prevent traffic congestion, especially when it comes to large quantum networks[39]. Here, we consider a continuous-distribution protocol introduced in ref. 37, and we aim to find parameter values that optimize its performance with respect to a measure of network utility. Before introducing this utility function, we first describe the protocol in detail.

The protocol in ref. 37 is carried out in multiple cycles $\Gamma \in \{1, 2, \ldots\}$, where each cycle is discretized into non-overlapping time slots $\tau \in \{0, 1, \ldots, n_\Gamma\}$. We assume as in ref. 37 that each node i holds a number of memory qubits proportional to the number of physical neighbors: $r \cdot d_i$, where $r \in \mathbb{N}$ is a given hardware parameter and d_i is the vertex degree of node i. In each cycle, the protocol carries out the following actions consecutively:

(1) Entangled links that have been existing longer than Γ_{cut} cycles are removed to exclude low-quality entanglement from the network[8]. It takes one time slot to execute this step.
(2) Next, entanglement generation is attempted on each physical link whose end nodes have available (vacant) memories. The attempt takes one time slot to complete, and an entangled link is generated with probability p_{gen}. This process is repeated sequentially (i.e., one physical link at a time, one time slot allocation each) for each physical link with access to vacant memories.
(3) Then, each node i chooses two entangled links $i-j$ and $i-k$ uniformly at random (if they exist), where j and k are non-neighboring nodes. Swap operations are executed on the corresponding memories with probability $q_{\text{swap},i} \in [0, 1]$. A swap succeeds deterministically, consumes both links and produces a $j-k$ link. All swaps are carried out in parallel, and the process takes one time slot to complete. As every swap decreases the quality of the entanglement[8], limiting the number of consecutive swaps to M presents a practical measure to prevent excessive decoherence.
(4) Every node gains information about its own qubits: for each entangled qubit, the age and the number of swaps it took to create the corresponding entangled link are updated. This step is assumed to be instantaneous.
(5) Lastly, pairs of nodes sequentially consume an existing entangled link with probability p_{cons}, with each consumption operation taking up one time slot. This concludes the first cycle $\Gamma = 1$.

For this use case, we consider as in ref. 37 a utility based on each network node's *virtual neighborhood*. This metric provides information about the number of nodes that are able to continuously run background applications. The virtual neighborhood $V^{(i)}(\Gamma)$ is the set of nodes that share at least one entangled link with node i at the end of cycle Γ. The objective is to maximize the virtual neighborhood size $|V^{(i)}(\Gamma)|$ for all nodes, which can be accomplished by tuning the swap probability $q_{\text{swap},i}$ at each node i. We use the vector $\mathbf{q}_{\text{swap}} = [q_{\text{swap},0}, q_{\text{swap},1}, \ldots, q_{\text{swap},N-1}]$ to represent all N nodes' swap probabilities.

Definition I.4. (Utility based on virtual neighborhood size). The virtual neighborhood size $U^{(i)}(\mathbf{q}_{\text{swap}}, \Gamma)$ of a node i at cycle Γ is denoted as

$$U^{(i)}(\mathbf{q}_{\text{swap}}, \Gamma) = |V^{(i)}(\mathbf{q}_{\text{swap}}, \Gamma)|. \tag{5}$$

We evaluate the performance of the protocol by studying the expected size of each node's virtual neighborhood at the steady state, $U^{(i)}(\mathbf{q}_{\text{swap}}) \equiv \lim_{\Gamma\to\infty} \mathbb{E}(U^{(i)}(\mathbf{q}_{\text{swap}}, \Gamma))$. We take the average at the end of the final simulation cycle $\Gamma = T_{sim}$ as an estimate for the expected steady-state value $U^{(i)}(\mathbf{q}_{\text{swap}}) \approx \bar{U}^{(i)}(f(\mathbf{q}_{\text{swap}}, T_{sim}))$, see Supplementary Note 6.

In contrast to previous use cases, our analysis of continuous-distribution protocols goes beyond aggregating objectives: it includes examining the individual node's goal to increase their own number of virtual neighbors. This approach is based on the Pareto frontier, a method of assessing multi-objective optimality. As outlined in Section "Surrogate-assisted Search", the surrogate optimization process naturally collects instances of $\{\mathbf{q}_{\text{swap}}\}$ and their simulation outcomes (objective values) as

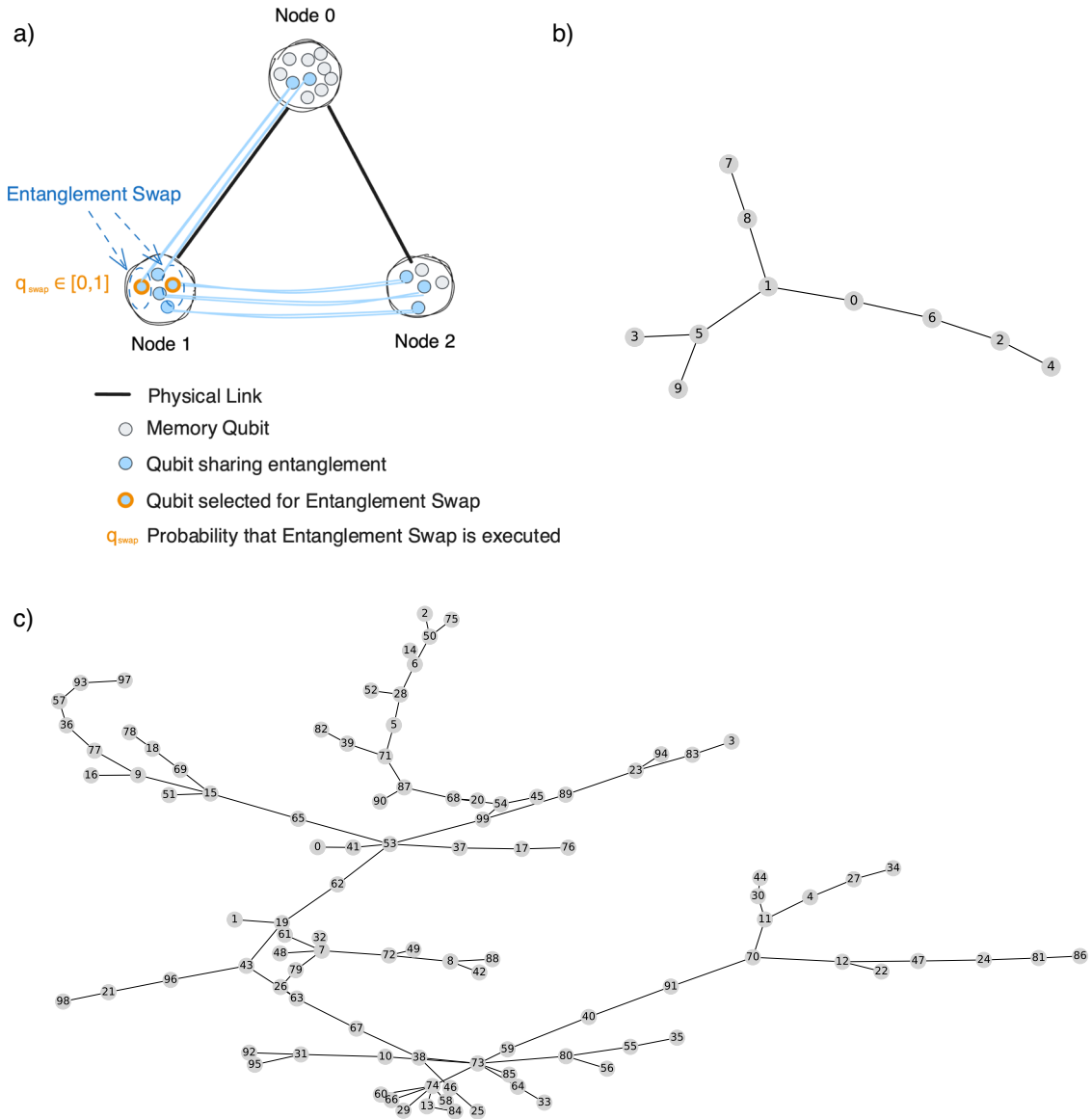

Fig. 6 | Quantum network topologies considered in the continuous distribution use case. a In the three-node network, Node 1 shares entangled links with both other nodes, while Node 0 and 2 each have only one virtual neighbor. An upcoming swap at Node 1 will allow all nodes to reach their maximum of two virtual neighbors. **b**, **c** illustrate larger networks. In all cases, the number of qubits per node is set by multiplying the node degree by $r = 5$. (Note: link lengths are visually scaled for clarity and do not represent actual distances between nodes).

training data. We will refer to the former as the *collected set*$\mathcal{S}$ of solutions. From this data, we can find the dominating set $\mathcal{S}_{\text{dom}} \subseteq \mathcal{S}$, which functions as an empirical estimate of the Pareto optimal set. It is important to note that this analysis of the collected solutions is not exclusive to this use case but could be applied to the other presented use cases as well. For clarity and illustrative purposes, however, we include this analysis only as part of this last use case.

Definition I.5. (Dominating Set $\mathcal{S}_{\text{dom}}$). Let $\mathcal{S}$ represent the set of all solutions collected during the surrogate optimization process, and let $\mathcal{S}_{\text{dom}}$ denote the subset of these solutions that form the dominating set. A solution $\mathbf{q}^*_{\text{swap}} \in \mathcal{S}$ belongs to $\mathcal{S}_{\text{dom}}$ if and only if there does not exist another solution $\mathbf{q}_{\text{swap}} \in \mathcal{S}$ such that $\mathbf{q}_{\text{swap}}$ dominates $\mathbf{q}^*_{\text{swap}}$. A solution $\mathbf{q}_{\text{swap}}$ dominates $\mathbf{q}^*_{\text{swap}}$ if:

$$\forall i \in \{1,\ldots,m\}, U^{(i)}(\mathbf{q}_{\text{swap}}) \geq U^{(i)}(\mathbf{q}^*_{\text{swap}}) \text{ and } \exists j \in \{1,\ldots,m\} : U^{(j)}(\mathbf{q}_{\text{swap}}) > U^{(j)}(\mathbf{q}^*_{\text{swap}}). \quad (6)$$

Thus, each solution in $\mathcal{S}_{\text{dom}}$ is non-dominated with respect to the objectives $\{U^{(1)}, \ldots, U^{(m)}\}$, and any solution in $\mathcal{S} \setminus \mathcal{S}_{\text{dom}}$ is dominated by at least one solution in $\mathcal{S}_{\text{dom}}$.

For an intuitive explanation, see supplementary Figure 9.

First, we investigate a simple three-node quantum network. Next, we extend our study to two larger asymmetric topologies involving 10 and 100 nodes, respectively. All three layouts are depicted in Fig. 6. We explore the different setups by analyzing the dominating set of the collected solutions. We will also demonstrate that as the number of objective functions (i.e., the number of nodes) increases, a larger proportion of solutions is non-dominated, which is a well-known consequence of multi-objective optimization in high dimensional spaces[46]. Lastly, we conduct the already familiar comparison of aggregated objectives found by surrogate optimization to the reference methods in the largest network topology. The parameter settings used in the numerical experiments are based on the previous study[37]: $p_{\text{gen}} = 0.9$, $T_{\text{sim}} = 1000$, $r = 5$, $M = 10$, $p_{\text{cons}} = p_{\text{gen}}/4$, $\Gamma_{\text{cut}} = 28$ cycles.

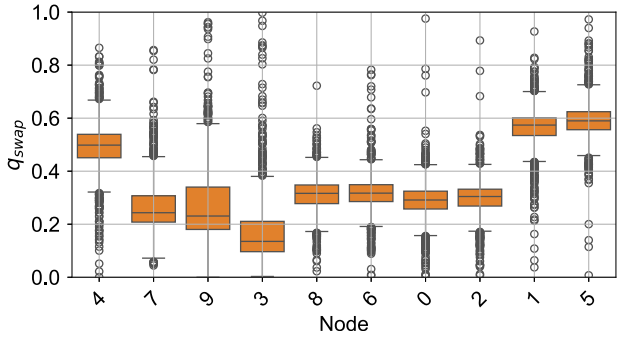

Fig. 7 | Swap probabilities of dominating solutions $\mathcal{S}_{\text{dom}}$. Distributions of swap values across nodes ordered by node degree. Each boxplot represents the solution values present in the dominating set. Execution parameters used in the surrogate workflow: $T = 100$ optimization cycles, $d = 4$, $n = 1000$.

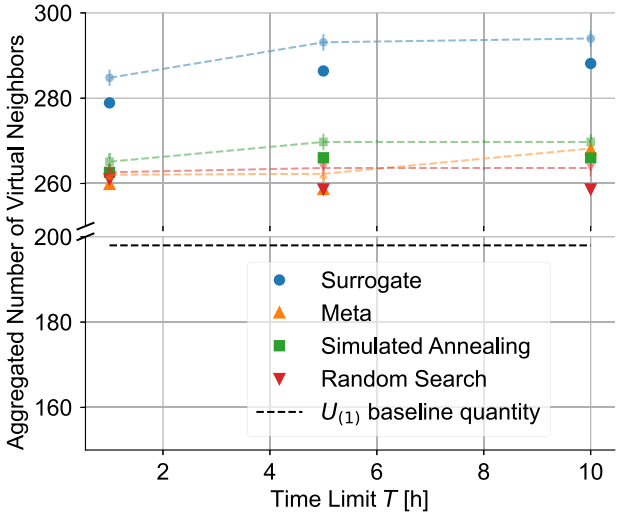

Fig. 8 | Aggregated number of virtual neighbors for best found solution. Light-colored markers (connected with dashed lines) indicate mean values found by the respective optimization methods including standard error bars. Once the best outcome per method is identified, we execute the simulation $n_{exec} = 1000$ times resulting in more accurate estimate of the mean presented by markers with standard error smaller than marker size. The black dashed line refers to the outcome of the basic solution (1) described in the main text. Parameters used in the optimization: $d = 4$, $n = 20$, $T = 1$, $T = 5$ and $T = 10$ h.

Three-node network. Using our surrogate framework, we simulate the continuous-distribution protocol from ref. 37 on a simple three-node network (see Fig. 6a). We aim to optimize the network's virtual neighborhood by tuning swap parameters $q_{swap,i}$ for each node i, where we let $q_{swap,1} = q_{swap,2}$ due to the network's symmetry. Post-optimization, we compute the dominating set[47]. The latter reflects the expected system behavior: we observe a wide-spread distribution at the leaf nodes while the swap probabilities at the central node ($q_{\text{swap},0}$) are concentrated around a low swap probability of 0.2 (see supplementary Fig. 10). These findings can be explained as follows: entanglement swapping at Nodes 1 and 2 only establishes virtual connections that are also directly generated over the physical links. This lessens the impact of such swaps, resulting in a wide-spread distribution of swap probability values for $q_{swap,1}$. Entanglement swapping is indeed only crucial for the entangled link shared between Nodes 1 and 2 (as there is no physical connection). A low swap probability around 0.2 at Node 0 proves effective, while a high swap probability would diminish the overall virtual neighborhood size. Through this case study, we see that even in small networks the dominating set offers non-trivial insights into the performance of continuous-distribution protocols across multiple objectives.

Random tree networks. When scaling to larger networks, the number of node interactions increases, but our objective remains the same. As in the previous setup, we evaluate the dominating set as a post-processing step. In comparison, while the three-node network has 3.5% solutions dominating, in the 10-node network (Fig. 6b) 99% solutions are in $\mathcal{S}_{\text{dom}}$. In addition to the standard deviation, we evaluate variation in the collected swap values using the Kolmogorov-Smirnov (KS) statistic[48], which measures the distance between two distributions. With the latter, we determine if a collected sample of swap values is closer to a uniform or normal distribution.

Figure 7 shows the swap probability values in $\mathcal{S}_{\text{dom}}$ across nodes obtained from our surrogate workflow. Swap probabilities at leaf nodes exhibit a larger standard deviation (>0.1) compared to all other nodes. Additionally, the distributions of nodes 4, 7 and 9 more closely align with a uniform distribution than with a normal distribution according to the evaluated KS measure. These metrics indicate that we should not draw a preference for the swap parameter settings at these nodes. Conversely, nodes with a higher vertex degree—such as nodes 1 and 5—show a more biased distribution with values clustering around a mean of ~0.6. This pattern is anticipated, as swapping at well-connected nodes is vital for enhancing the virtual connectivity of other nodes.

Next, we examine the largest network in this study, depicted in Fig. 6c, involving 100 configurable swap probabilities. We apply our workflow under three different time constraints—1, 5, and 10 h. All solutions are non-dominated, reflecting the already anticipated *curse of dimensionality*[46]. Furthermore, all collected data in the dominating sets reflect more closely a uniform than a normal distribution, with a standard deviation greater than 0.1, meaning that we should refrain from drawing generalized conclusions in this scenario.

In addition to examining the dominating set, we compare the maximum aggregated neighborhood size

$$\max_{\mathbf{q}_{\text{swap}}} \sum_{i=1}^{N} U^{(i)}(\mathbf{q}_{\text{swap}}), \tag{7}$$

to the reference methods (Section "Baselines") under said constraints. Furthermore, we compare the solutions found via numerical methods against four distinct protocol settings that serve as additional baselines: (1) setting all swap probabilities to zero, $\mathbf{q}_{\text{swap}} = \vec{0}$; (2) setting all swap probabilities to one, $\mathbf{q}_{\text{swap}} = \vec{1}$; and (3) setting swap probabilities of leaf nodes to zero while others are set to one. Note that in our configuration the mean virtual connectivity in the steady state of the basic solution (1) results into the sum of all node degrees, $U_{(1)} = \sum_i U^{(i)}(\mathbf{q}_{\text{swap}} = \vec{0}) = 198$. The latter thus serves as reference point for comparing the optimization methods, representing the baseline quantity of link-level entangled states without swapping. We summarize the optimization results for all algorithms in Fig. 8. The Bayesian optimizer developed by Meta only outperforms random search in the largest time limit, while our workflow performs significantly better in all scenarios, achieving as much as 29.4% and 28.2% more virtual neighbors than Simulated Annealing and Meta, respectively, using $U_{(1)}$ as baseline quantity. Meta only outperforms Simulated Annealing in the largest time limit; this is likely because the 100 variables present in this network are far beyond the recommended limit of 20[27]. Conversely, the models used by our surrogate optimizer are less impacted by the increase in variables, showing significant improvement with larger execution times. Note that even though Meta identifies a solution within a 5-h limit—using $n = 20$ simulation runs—that exhibits the same virtual neighborhood size as the solution found in the 1-h limit, the use of $n_{exec} = 1000$ simulation runs of the result depicted in Fig. 8 provides a more accurate estimate of the average, albeit a smaller one.

To conclude this section, we summarize the most relevant findings: across different quantum network sizes, the collected dominating set can provide relevant insights to the behavior of parameters in continuous-distribution protocols when multiple objectives are to be met. Even for a 10-

node setup, swap probability distributions are biased at some nodes, which can guide the configuration of these parameters in a protocol. However, the ratio of non-dominated to dominated solutions increases significantly with the number of objectives; in the largest network studied, this leads to all parameter distributions being closer to a uniform than to a normal distribution. Although the dominating set provided limited insights in the largest setup, we could still find a promising solution to the single-objective case, i.e., maximizing the sum over all objectives. The latter comprises 286.2 virtual neighbors on average achieving 28% larger virtual neighborhood size than the solution found by Meta using $U_{(1)}$ as reference point.

Further benchmarking

We conduct further benchmarking on 54 standardized test functions for global optimization (Rastrigin, Rosenbrock, Schaffer, etc.), provided by the benchmarking platform COCO[49]. In these experiments, we compare our surrogate framework to additional global solvers, such as MIDACO (unlimited version)[50], BFGSN[51], EDA-PSO[52], GA[53], DASA[54] and POEMS[55]. We include the benchmarking results involving up to 20 variables and further details in Supplementary Note 3. From these detailed results, we can conclude that for most of the test functions in the noisy testbed, our approach is competitive with other global optimizers, up to the limit of 20 variables in the considered regime of up to 10^4 function evaluations. However, all test functions provided are *single-valued*, which prevents our method from exploiting one of its key advantages–namely, learning multiple competing objectives simultaneously and optimizing a composite cost function that depends on all of them.

Discussion

We introduced a surrogate-based optimization workflow that integrates detailed quantum network simulations. Through three distinct use cases, we demonstrated the workflow's broad applicability to practical optimization problems and its effectiveness in enhancing the performance of both on-demand and continuous entanglement distribution protocols across various network topologies. Particularly, the application to a five-user quantum entanglement switch and a metropolitan network performing purification protocols, showed our method's efficacy to handle complex and highly asymmetric quantum networking scenarios. Our workflow efficiently handled up to 100 network parameters and achieved approximate solutions that significantly outperformed optimization techniques such as Simulated Annealing (up to 29%) and Bayesian optimization (up to 28%) in tested scenarios. Moreover, finding the dominating set of collected solutions allowed us to derive insights from a multi-objective perspective.

Due to the potentially large computational costs of numerical simulation, our approach is suitable for scenarios where the simulation models an analytically intractable problem, i.e., a problem scenario in which it is justified to invest additional time in numerical optimization.

We note that in addition to the work discussed in the above scenarios, relevant prior studies were carried out by Ferreira da Silva et al.[17], Wallnhöfer et al.[18], Khatri et al.[19] and Haldar et al.[20]. Ref. 17 utilized genetic algorithms combined with simulation of repeater chains in NetSquid. In contrast to genetic algorithms, our approach utilized simple machine learning models to save on extensive evaluation of the expensive objective function, which helped to traverse the search space more efficiently. Further, ref. 17 focused on optimizing for minimum requirements to satisfy target benchmarks, while we investigated diverse utility functions based on different use cases. Similar to our approach, the other studies[18–20] used machine learning techniques to find optimal quantum network protocols. These studies were based on the theoretical framework of decision processes. Ref. 18 introduced for the first time learning agents for quantum networks, which by trial and error manipulate quantum states and thereby construct communication protocols. Refs. 19,20 devised a mathematical framework to optimize link-level entanglement generation in general networks allowing for said learning agents to discover policies. In contrast to our approach, learning agents are able discover complete protocols (devising optimal sequences of actions), while surrogate optimization is limited to finding improved parameter settings *within* protocols. However, our approach offers two distinct advantages: (1) It is applicable to enhancing hardware configurations, such as the allocation of memory qubits per node. In contrast, learning agents operate within a static architectural environment, limiting them to discover only quantum network protocols; (2) We employ detailed quantum network simulations, in contrast to the purely mathematical or overly simplified models used in other studies. This allows us to address complex network topologies beyond, e.g., simple linear repeater chains typically considered in prior work.

Methods

Baselines

In order to evaluate our approach, we utilize the following baselines: (1) *Uniform random search*: this method chooses sets of parameter values uniformly at random from X_{conf} to execute the simulation. We employ this method as our foundational baseline because its constraints are easily managed through time or iteration limits. Uniform sampling prevents being constrained to a specific area of the search domain, a limitation that might be encountered by a time- or iteration-limited exhaustive grid search. (2) *Simulated Annealing*[41]: this conventional global optimization method starts by accepting less optimal solutions with high probability, thus enabling exploration of the search space and escaping from local optima using the so-called Metropolis criterion. We implemented the algorithm using the fast annealing schedule[24] and found objective values of various benchmark functions[56] comparable to the L-BFGS-B method. (3) *Bayesian optimization API by Meta*: Bayesian optimization is a surrogate method based on Gaussian processes, which is mostly used to optimize unknown but continuous functions; empirical studies suggest optimal performance below 20 variables[27]. The Service/Loop API, developed by Meta, incorporates a Bayesian optimization algorithm[40], where we adhere to the default configuration settings recommended in the official documentation (https://ax.dev/docs/bayesopt).

Simulation experiments

For each use case scenario, we compared optimization methods by conducting ten independent repetitions, each with an allotted time limit of T hours. To compute averages $\bar{U}^{(i)}(f,.)$ we used $n = 20$ simulation runs by default. We report on the distribution of the ten repetitions in Supplementary Note 7. From all repetitions, the solution that performed best according to the objective function is selected, and a subsequent simulation involving $n_{exec} = 10^3$ simulation runs were executed. This approach leads to statistically robust solutions, characterized by small standard errors (explicit values given in captions of respective result figures). In the scenarios, where we did not compare to reference methods, T is a set number of optimization cycles and $n = 1000$. Note that in scenarios involving time constraints for comparison between reference methods, the outcomes are specific to our computing system. For instance, let us assume a simulation takes 9 s to execute on our system but 10 s on another system. If we impose a time limit that permits ten executions on our system, the same time limit would only allow nine executions on the slower system, resulting in different outcomes. Since execution times depend on multiple aspects of a computing node (processor speed, RAM, operating system), to ensure exact reproducibility of output data, it is advisable to use a fixed number of iterations; we diverged from this to enable fair comparison between used optimization methods. Each optimization method was allocated ten cores on an Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz and 64 GB of memory.

Data availability

Source data, as well as well as generated machine learning models and time profiling data is available at https://doi.org/10.4121/a07a9e97-f34c-4e7f-9f68-1010bfb857d0.

Code availability

The code used to generate reported data, as well as usage guidelines can be found in the Git repository https://github.com/Luisenden/qnetsur and corresponding documentation https://qnetsur.readthedocs.io.

Received: 13 September 2024; Accepted: 7 May 2025;
Published online: 31 May 2025

References

1. Wehner, S., Elkouss, D. & Hanson, R. Quantum internet: a vision for the road ahead. *Science* **362**, eaam9288 (2018).
2. Bennett, C. H., DiVincenzo, D. P., Smolin, J. A. & Wootters, W. K. Mixed-state entanglement and quantum error correction. *Phys. Rev. A* **54**, 3824 (1996).
3. Ekert, A. K. Quantum cryptography and Bell's theorem. *Quantum Measurements in Optics* Vol. 413 (Springer, 1992).
4. Giovannetti, V., Lloyd, S. & Maccone, L. Quantum-enhanced measurements: beating the standard quantum limit. *Science* **306**, 1330 (2004).
5. Jozsa, R., Abrams, D. S., Dowling, J. P. & Williams, C. P. Quantum clock synchronization based on shared prior entanglement. *Phys. Rev. Lett.* **85**, 2010 (2000).
6. Broadbent, A., Fitzsimons, J., and Kashefi, E. Universal blind quantum computation, in *Proc. 50th Annual IEEE Symposium on Foundations of Computer Science* 517–526 (IEEE, 2009).
7. Fitzsimons, J. F. & Kashefi, E. Unconditionally verifiable blind quantum computation. *Phys. Rev. A* **96**, 012303 (2017).
8. Azuma, K. et al. Quantum repeaters: from quantum networks to the quantum internet. *Rev. Mod. Phys.* **95**, 045006 (2023).
9. Dahlberg, A. et al. A link layer protocol for quantum networks, in *Proceedings of the ACM Special Interest Group on Data Communication, Series and Number SIGCOMM '19* 159–173 (Association for Computing Machinery, 2019).
10. Van Meter, R., Ladd, T. D. & Nemoto, K. System design for a long-line quantum repeater. *IEEE/ACM Trans. Netw.* **17**, 1002 (2008).
11. Vardoyan, G., Guha, S., Nain, P. & Towsley, D. On the stochastic analysis of a quantum entanglement switch. *ACM SIGMETRICS Perform. Eval. Rev.* **47**, 27 (2019).
12. Iñesta, A. G., Vardoyan, G., Scavuzzo, L. & Wehner, S. Optimal entanglement distribution policies in homogeneous repeater chains with cutoffs. *npj Quantum Inform.* **9**, 46 (2023).
13. Vardoyan, G. & Wehner, S. Quantum network utility maximization. In *Proc. IEEE International Conference on Quantum Computing and Engineering (QCE)*, Vol. 1, 1238–1248 (IEEE, 2023).
14. Liao, C.-T., Bahrani, S., da Silva, F. F. & Kashefi, E. Benchmarking of quantum protocols. *Sci. Rep.* **12**, 5298 (2022).
15. Avis, G. et al. Requirements for a processing-node quantum repeater on a real-world fiber grid. *NPJ Quantum Inf.* **9**, 100 (2023).
16. Kozlowski, W., Dahlberg, A. & Wehner, S. Designing a quantum network protocol. In *Proc. 16th International Conference on Emerging Networking Experiments and Technologies* 1–16 (IEEE, 2020).
17. Da Silva, F. F., Torres-Knoop, A., Coopmans, T., Maier, D. & Wehner, S. Optimizing entanglement generation and distribution using genetic algorithms. *Quantum Sci. Technol.* **6**, 035007 (2021).
18. Wallnöfer, J., Melnikov, A. A., Dür, W. & Briegel, H. J. Machine learning for long-distance quantum communication. *PRX Quantum.* https://doi.org/10.1103/prxquantum.1.010301 (2020).
19. Khatri, S. Policies for elementary links in a quantum network. *Quantum* **5**, 537 (2021).
20. Haldar, S., Barge, P. J., Khatri, S. & Lee, H. Fast and reliable entanglement distribution with quantum repeaters: principles for improving protocols using reinforcement learning. *Phys. Rev. Appl.* **21**, 024041 (2024).
21. Coopmans, T. et al. Netsquid, a network simulator for quantum information using discrete events. *Commun. Phys.* **4**, 164 (2021).
22. Wu, X. et al. Sequence: a customizable discrete-event simulator of quantum networks. *Quantum Sci. Technol.* **6**, 045027 (2021).
23. Box, G. E. & Draper, N. R. *Empirical Model-building and Response Surfaces* (John Wiley & Sons, 1987).
24. Kochenderfer, M. J. & Wheeler, T. A. *Algorithms for Optimization* (MIT Press, 2019).
25. Shahriari, B., Swersky, K., Wang, Z., Adams, R. P. & de Freitas, N. Taking the human out of the loop: a review of bayesian optimization. *Proc. IEEE* **104**, 148 (2016).
26. Tripathy, R. K. & Bilionis, I. Deep UQ: learning deep neural network surrogate models for high dimensional uncertainty quantification. *J. Comput. Phys.* **375**, 565 (2018).
27. Frazier, P. I. Bayesian optimization, in *Recent Advances in Optimization and Modeling of Contemporary Problems* 255–278 (Informs, 2018).
28. Weiss, S. M. & Kulikowski, C. A. *Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems* (Morgan Kaufmann Publishers Inc., 1991).
29. Carvalho, D. V., Pereira, E. M. & Cardoso, J. S. Machine learning interpretability: a survey on methods and metrics. *Electronics* **8**, 832 (2019).
30. Gunn, S. R. et al. Support vector machines for classification and regression. *ISIS Tech. Rep.* **14**, 5 (1998).
31. Breiman, L. Random forests. *Mach. Learn.* **45**, 5 (2001).
32. Bhosekar, A. & Ierapetritou, M. Advances in surrogate based modeling, feasibility analysis, and optimization: a review. *Comput. Chem. Eng.* **108**, 250 (2018).
33. Wang, J. Y. et al. Identifying general reaction conditions by bandit optimization. *Nature* **626**, 1025 (2024).
34. Kusne, A. G. et al. On-the-fly closed-loop materials discovery via Bayesian active learning. *Nat. Commun.* **11**, 5966 (2020).
35. Popp, J., Haider, M., Franckié, M., Faist, J. & Jirauschek, C. Bayesian optimization of quantum cascade detectors. *Opt. Quantum Electron.* **53**, 287 (2021).
36. Cortes, C. L. et al. Sample-efficient adaptive calibration of quantum networks using Bayesian optimization. *Phys. Rev. Appl.* **17**, 034067 (2022).
37. Iñesta, Á. G. & Wehner, S. Performance metrics for the continuous distribution of entanglement in multiuser quantum networks. *Phys. Rev. A* **108**, 052615 (2023).
38. Pedregosa, F. et al. Scikit-learn: machine learning in Python. *J. Mach. Learn. Res.* **12**, 2825 (2011).
39. Chakraborty, K., Dahlberg, A., Rozpedek, F., & Wehner, S. Distributed Routing in a Quantum Internet. *APS March Meeting Abstracts* **28** (2019)
40. Balandat, M. et al. Botorch: a framework for efficient Monte-Carlo Bayesian optimization. *Adv. neural Inf. Process. Syst.* **33**, 21524 (2020).
41. Kirkpatrick, S., Gelatt Jr, C. D. & Vecchi, M. P. Optimization by simulated annealing. *Science* **220**, 671 (1983).
42. Rains, E. M. A semidefinite program for distillable entanglement. *IEEE Trans. Inf. Theory* **47**, 2921 (2001).
43. Humphreys, P. C. et al. Deterministic delivery of remote entanglement on a quantum network. *Nature* **558**, 268 (2018).
44. Rančić, M., Hedges, M. P., Ahlefeldt, R. L. & Sellars, M. J. Coherence time of over a second in a telecom-compatible quantum memory storage material. *Nat. Phys.* **14**, 50 (2018).
45. Bennett, C. H. et al. Purification of noisy entanglement and faithful teleportation via noisy channels. *Phys. Rev. Lett.* **76**, 722 (1996).
46. Kukkonen, S. & Lampinen, J. Ranking-dominance and many-objective optimization. In *Proc. IEEE Congress on Evolutionary Computation* 3983–3990 (IEEE, 2007).
47. Kung, H.-T., Luccio, F. & Preparata, F. P. On finding the maxima of a set of vectors. *J. ACM* **22**, 469 (1975).
48. Lilliefors, H. W. On the Kolmogorov-Smirnov test for normality with mean and variance unknown. *J. Am. Stat. Assoc.* **62**, 399 (1967).

49. Hansen, N. et al. COCO: A platform for comparing continuous optimizers in a black-box setting. *Optim. Methods Softw.* **36**, 114 (2021).
50. Schlüter, M., Gerdts, M. & Rückmann, J.-J. A numerical study of midaco on 100 minlp benchmarks. *Optimization* **61**, 873 (2012).
51. Ros, R. Benchmarking the BFGS algorithm on the BBOB-2009 function testbed. *Association for Computing Machinery,* 2409–2414 (2009).
52. El-Abd, M. & Kamel, M. S. Black-box optimization benchmarking for noiseless function testbed using an EDA and PSO hybrid. *Association for Computing Machinery* 2263–2268 (2009).
53. Ryan, C., Nicolau, M. & O'Neill, M. Genetic algorithms using grammatical evolution. *Springer Berlin Heidelberg* (2002).
54. Korošec, P., Šilc, J. & Filipič, B. The differential ant-stigmergy algorithm. *Inf. Sci.* **192**, 82 (2012).
55. Kubalik, J. and Faigl, J. Iterative prototype optimisation with evolved improvement steps. In *European Conference on Genetic Programming* 154–165 (Springer, 2006).
56. Jamil, M. & Yang, X.-S. A literature survey of benchmark functions for global optimisation problems. *Int. J. Math. Model. Numer. Optim.* **4**, 150 (2013).

Acknowledgements

This work is supported by QuTech NWO funding 2020-2024 Part I "Fundamental Research", Project Number 601.QT.001-1, financed by the Dutch Research Council (NWO). We further acknowledge support from *NWO QSC grant BGR2 17.269*. Á.G.I. acknowledges financial support from the Netherlands Organisation for Scientific Research (NWO/OCW), as part of the Frontiers of Nanoscience program.

Author contributions

L.P. and A.G.I. conceived the project. L.P. developed the theory, the algorithm and performed the computations. A.G.I. proposed investigating continuous distribution protocols and G.V. proposed investigating utility maximization of a quantum entanglement switch. A.G.I. and G.V. verified the analytical methods. L.P. wrote the manuscript with input from all authors. G.V. supervised the project.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41534-025-01048-3.

Correspondence and requests for materials should be addressed to Luise Prielinger.