Optimizing entanglement generation and distribution using genetic algorithms

Ferreira Da Silva, Francisco; Torres-Knoop, Ariana; Coopmans, Tim; Maier, David; Wehner, Stephanie

DOI
10.1088/2058-9565/abfc93

Publication date
2021

Document Version
Final published version

Published in
Quantum Science and Technology

Quantum Science and Technology

CrossMark

OPEN ACCESS

RECEIVED
18 November 2020

REVISED
15 April 2021

ACCEPTED FOR PUBLICATION
28 April 2021

PUBLISHED
15 June 2021

PAPER

Optimizing entanglement generation and distribution using genetic algorithms

Francisco Ferreira da Silva[1,2,*], Ariana Torres-Knoop[3], Tim Coopmans[1,2], David Maier[1,2] and Stephanie Wehner[1,2]

[1] QuTech, Delft University of Technology, Lorentzweg 1, 2628 CJ Delft, The Netherlands
[2] Kavli Institute of Nanoscience, Delft University of Technology, Lorentzweg 1, 2628 CJ Delft, The Netherlands
[3] SURF, Utrecht, Postbus 19035, 3501 DA Utrecht, The Netherlands
[*] Author to whom any correspondence should be addressed.

E-mail: f.hortaferreiradasilva@tudelft.nl

Keywords: quantum repeater, entanglement distribution, optimization, repeater chain, quantum network, genetic algorithms, quantum internet

Abstract

Long-distance quantum communication via entanglement distribution is of great importance for the quantum internet. However, scaling up to such long distances has proved challenging due to the loss of photons, which grows exponentially with the distance covered. Quantum repeaters could in theory be used to extend the distances over which entanglement can be distributed, but in practice hardware quality is still lacking. Furthermore, it is generally not clear how an improvement in a certain repeater parameter, such as memory quality or attempt rate, impacts the overall network performance, rendering the path toward scalable quantum repeaters unclear. In this work we propose a methodology based on genetic algorithms and simulations of quantum repeater chains for optimization of entanglement generation and distribution. By applying it to simulations of several different repeater chains, including real-world fiber topology, we demonstrate that it can be used to answer questions such as what are the minimum viable quantum repeaters satisfying given network performance benchmarks. This methodology constitutes an invaluable tool for the development of a blueprint for a pan-European quantum internet. We have made our code, in the form of NetSquid simulations and the *smart-stopos* optimization tool, freely available for use either locally or on high-performance computing centers.

1. Introduction

A quantum internet could be used to perform tasks that are impossible with classical communications alone, the best known example being that of quantum key distribution (QKD) [1, 2]. Beyond QKD, several other applications have been identified, ranging from quantum clock synchronization [3] to distributed quantum computing [4]. The level of network development required is application-dependent, but all of them rely on entanglement generation and distribution [5].

Entanglement generation has been demonstrated at short distances [6], but scaling up has proved very challenging due to the exponential growth of photon losses with the length of fiber covered. Classically, photon loss is overcome by direct amplification, but in the quantum case this is impossible for non-orthogonal states due to the no-cloning theorem. As an alternative, two distant end nodes can be connected by intermediate nodes, known as quantum repeaters [7]. These are devices that can, in theory, enable long-distance entanglement generation. This is done by (i) establishing elementary links between neighboring nodes, i.e. entangled states shared by these nodes and (ii) gluing links together by means of bell state measurements, a process known as entanglement swapping. The simplest possible quantum repeater protocol consists in having nodes constantly trying to generate entanglement and swapping as soon as they hold two entangled qubits, one to each side of the chain. This is known as an SWAP-ASAP protocol. It can

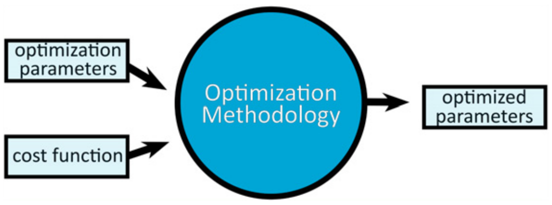


Figure 1. Overview of our optimization process. The user inputs the desired optimization parameters and defines a cost function. Using simulation and optimization tools, our methodology finds a set of parameters optimizing the cost function. For example, the optimization parameters could be parameters defining a quantum repeater model and the cost function could be the inverse of the secret key rate plus a penalty term for parameter values that are much better than a given baseline. The output would then be the values of the parameters defining the quantum repeater model optimizing the cost function.

be enhanced by imposing a cut-off condition, such as a maximum time after which stored entanglement is discarded [8]. For an in-depth introduction to quantum repeaters, see for example [9].

Despite ongoing experimental efforts, a scalable quantum repeater has yet to be demonstrated [10]. Several physical systems are being explored as possible platforms for such a repeater, for example color centers in diamond (e.g. NV centers [11]), atomic ensembles [12] and trapped ions [13], but it is not yet clear which are most feasible in the short-term, nor how imperfections in the physical system would generally affect relevant network performance metrics like end-to-end fidelity and entanglement generation rate.

In the quest for a quantum internet, the question of what the minimal requirements on a quantum repeater are to achieve a certain network performance benchmark is thus fundamental. Furthermore, as we will show in section 2, it is a question that can be framed as an optimization problem. Broadly speaking, two different approaches are being explored in the theoretical study of quantum repeaters: analytical (see e.g. [7, 9, 14]) and simulation-based (see e.g. [15–17]). In the first case, simplifying assumptions are made, such as approximating the states shared between nodes by Werner states or assuming simple topologies for the networks under study, e.g. restricting the analysis to chains of equally spaced nodes. In the second case, accurate and realistic simulations of networks of quantum repeaters are developed, at both the protocol level and the physical level. Each of these approaches offers benefits and drawbacks; opting for an analytical approach enables obtaining analytical expressions for interesting metrics such as the end-to-end fidelity. This means that traditional gradient-based optimization methods can be employed, offering a clear path to an answer. However, this may come at the cost of less detailed predictive power. On the other hand, choosing a simulation-based approach allows the modeling to be as realistic as desired. The downside here is, of course, that analytical results are no longer available, and that the simulations may become very complex, especially taking into account the exponential overhead in simulating quantum systems on classical computers. It also renders optimizing more difficult, as all that is made available to the optimization procedure are the inputs and outputs of the simulation in consideration. For example, having more information about the function landscape, such as number of minima, would render the optimization process simpler. Besides this, a realistic simulation requires a large number of parameters, thus resulting in a large search space to be explored. Nonetheless, if one wants to arrive at a realistic answer, a simulation-based approach seems inevitable.

In this work we propose a methodology based on genetic algorithms (GAs) and simulations of quantum repeaters for optimization of entanglement generation and distribution in quantum networks. This allows us to answer questions such as what are the worst possible repeaters satisfying target benchmarks. Contrasting with previous work on repeater chain optimization [18–23], our methodology constitutes a systematic and modular approach to this problem, successfully integrating simulation and optimization tools, as well as allowing for the use of high-performance computing (HPC) clusters. A high-level overview of how a user interfaces with this process is shown in figure 1.

We performed our simulations using NetSquid [15]. NetSquid can accurately model the effects of time-dependent noise, rendering it well equipped to predict quantum network performance in a physically accurate setting. The tools used in this methodology, which allow for running NetSquid simulations together with an optimization algorithm both locally and on an HPC cluster, are made freely available (see [24]).

We structure the paper as follows. In section 2 we introduce our methodology, together with the required preliminaries. Section 3 concerns the validation of the methodology. This comprises two steps: (i) benchmarking our GA implementation by running it on standard optimization problems and comparing its performance to those found in the literature; and (ii), validating our approach by applying it to a repeater chain where elementary link states are in the Werner form and all noise sources are depolarizing [25]. In this case, analytical results can be found, so we can evaluate how well our optimization method performs.

After validating our methodology, we apply it to some different repeater chain setups, in order to demonstrate its potential usefulness. We present these results in section 4, where we first consider a repeater chain based on real-life fiber data, courtesy of SURF, a classical network provider for Dutch education and research institutions. This showcases the power of our simulation-based approach, as chains of unevenly spaced nodes are hard to study analytically. We further apply our methodology to chains of varying length, internode distance and number of repeaters and we compare the solutions found with our methodology for each of these different setups. This allows us to investigate how the impact of the parameters varies across setups, thus identifying possible bottlenecks and paths toward scalable quantum repeaters.

2. Methodology

In this section we introduce the main contribution of our work, a methodology for the optimization of entanglement generation and distribution. We first present each of the elements that are used in this optimization process. We finalize the section with an overview of how they are integrated to answer the question of what are the minimum requirements on quantum repeaters to achieve a given benchmark.

2.1. Question

We aim to answer the question of what the minimum requirements are on the quality of quantum repeaters to achieve a given benchmark by framing it as an optimization problem. To do so, we must first clarify what we mean by requirements and by quality of a quantum repeater. Let us say that a quantum repeater is described, in a given model, by a set of N parameters $\{x_i\}_{i\in\{1,\ldots,N\}}$. The meaning of x_j is model dependent. For example, if we consider a model of a trapped ion system, x_j and x_k could be single-qubit and two-qubit gate error probabilities. We could also, in a more abstract model, combine these two parameters together to obtain a swap quality that quantifies the noise introduced in an entanglement swap operation, which would then be y_j in this model. The quality of a quantum repeater is then a function of the set of parameters describing it. This also helps clarify what we mean by requirements. Suppose we have some fixed network topology and performance metric. To give a concrete example, the topology could be a repeater chain of 10 equally spaced nodes and the performance metric the end-to-end secret key rate. The requirements on the repeaters are then the worst set of parameters that enable attaining some value of the end-to-end secret key rate over a chain of 10 nodes, i.e. the lowest quality repeaters satisfying this metric. The meaning of repeater quality will be made clear in the following section.

2.2. Cost

Let us say that we have two repeaters described by a set of parameters $\{y_i\}_{i\in\{1,\ldots,N\}}$ and $\{z_i\}_{i\in\{1,\ldots,N\}}$, and that the values of these parameters are the same for all but two of them, i.e. $\{y_i\} = \{z_i\} \forall i \in \{1, 2, \ldots, N\}\setminus\{j, k\}$. Let us further say that y_j is better than z_j, but z_k is better than y_k. Which of these sets of parameters is the better one? To answer this, we will now introduce the quantity to be optimized, the cost function. We emphasize that our method is completely general and could be applied to any cost function, but for concreteness we focus on a particular one from here on out.

We expect that in an experimental setting a given physical parameter becomes harder to improve the closer to its perfect value it is, so we would like our cost function to reflect this. We start by transforming our parameters so that they all live in the $[0, 1]$ interval, with 1 being the perfect value and 0 the worst possible value. We refer to appendix B for details. Denoting x_b as the baseline value of a parameter, i.e. the value from which we are improving, k as the improvement factor and x_new as the new improved value, we claim that the following equation reflects this behavior:

$$x_\mathrm{new}(k) = x_\mathrm{b}^{\frac{1}{k}}. \tag{1}$$

This can be read as: we improve x_b by a factor k to get x_new. To see that equation (1) does in fact reflect the desired behavior, we note the that

$$x_\mathrm{new}(k = 1) = x_\mathrm{b}, \tag{2}$$

$$\lim_{k\to\infty} x_\mathrm{new} = 1. \tag{3}$$

Equation (2) can be read as: improving a parameter by a factor of 1 is equivalent to not improving it all, whereas equation (3) can be taken to mean that in order to improve a parameter to its perfect value we must improve by a factor of infinity, i.e. there is no such thing as a perfect process.

We can then define the cost associated to x_{new} as the factor k by which we must improve the baseline value x_{b} to obtain x_{new}. Therefore, solving equation (1) for k, we get

$$k = \frac{1}{\log_{x_{\text{b}}}\left(x_{\text{new}}\right)}. \tag{4}$$

With this in hand, we can finally define the cost associated to a set of parameters. Let us say our model is described by a set of parameters $\{x_i\}_{i\in\{1,\ldots,N\}}$, and that the current baseline value of each of these parameters is $\{x_{i_{\text{b}}}\}_{i\in\{1,\ldots,N\}}$. A set of values $\{x_{i_{\text{c}}}\}_{i\in\{1,\ldots,N\}}$ is mapped to a cost, C, by equation (5). Intuitively, this can be seen as taking the average of the cost associated to each of the parameters.

$$C\left(x_{1_{\text{c}}},\ldots,x_{N_{\text{c}}}\right) = \sum_{i=1}^{N} \frac{1}{\log_{x_{i_{\text{b}}}}\left(x_{i_{\text{c}}}\right)}. \tag{5}$$

Note that with this definition, the minimum parameter cost is N, with N being the number of parameters in the model under consideration. Since this cost function is meant to be used for comparing the relative cost of parameter sets of the same model, N is the same for all parameter sets under consideration, and hence it is nothing but a constant shift in each set's cost. One could, for instance, divide the cost by N to normalize it or subtract N from it, making the minimum cost 0. This would however have no impact on the results obtained, since the relative ordering of parameter sets according to their cost would remain the same.

There is still the matter of how the network's target performance metrics are taken into account. Throughout this work we will focus on fidelity F of the end-to-end state with the ideal Bell state and entanglement generation rate R, but we stress that our method is not limited to optimizing for these quantities. More concretely, we will try to answer the question of what are the minimum requirements on repeaters to concurrently achieve certain values of F and R. We are then faced with a multi-objective problem, as we want to optimize multiple quantities simultaneously, namely end-to-end fidelity, entanglement generation rate and parameter cost. Furthermore, there are trade-offs between these goals. For example, improving the memory lifetime of nodes in a chain has a positive contribution toward end-to-end entanglement fidelity, but a negative one toward parameter cost. There is a multitude of possible ways of approaching such problems [26]. We chose to map our multi-objective optimization problem to a single-objective one by assigning weights to the different objectives and adding them, a process known as scalarization [27]. In this way, the total cost function T_{C} to minimize becomes a weighted sum of the parameter cost and the thresholds on end-to-end rate and fidelity:

$$T_{\text{C}}\left(p_{1_{\text{c}}},\ldots,p_{N_{\text{c}}},F_{\min},R_{\min}\right) = w_1\Theta(F_{\min}-F) + w_2\Theta(R_{\min}-R) + w_3 C\left(x_{1_{\text{c}}},\ldots,x_{1_N}\right), \tag{6}$$

where the w_i are the weights of each objective, Θ is the Heaviside function and $F_{\min}$ and $R_{\min}$ are, respectively, the minimum required end-to-end fidelity and end-to-end entanglement generation rate. Using step functions reflects the idea that we are looking for solutions that satisfy performance benchmarks, with no reward given for surpassing them. The weights in equation (6) are hyperparameters of our method, meaning that they are not determined by some algorithm but must instead be chosen. This choice can be of any real number, and it has an impact on which sets of parameters have the lowest costs and hence on the solutions found by the method. For example, if we assign very high values to w_1 and w_2 and a low value to w_3 the best sets of parameters will be those that satisfy the requirements on the end-to-end fidelity and rate without much regard for how costly it is to achieve them. To give a concrete example of what the hyperparameter values might be, for the applications we present in section 4, we set w_1 and w_2 to 20 000 and w_3 to 1. The parameter cost term, defined in equation (5), depends on the baseline values of the parameters, which must also be chosen. Typically, for the use cases we consider, these will be chosen to reflect what is currently achievable experimentally.

Optimal solutions to this single-objective optimization problem are then solutions to the multi-objective optimization problem.

2.3. Abstract model

In order to explore and better understand the methodology we propose, we believe it to be wise to employ a relatively simple model whose behavior we understand. We must however again emphasize that our methodology is completely general in terms of the model used for the quantum repeater hardware.

We consider a simplified five-parameter model for a quantum repeater, the five parameters being denoted by $[F_{\text{EL}}, p_{\text{suc}}, s_{\text{q}}, T_1, T_2]$. We assume that elementary links states have fidelity F_{EL} with the ideal Bell state upon generation, and that they are generated with a success probability p_{suc}. We assume also that each

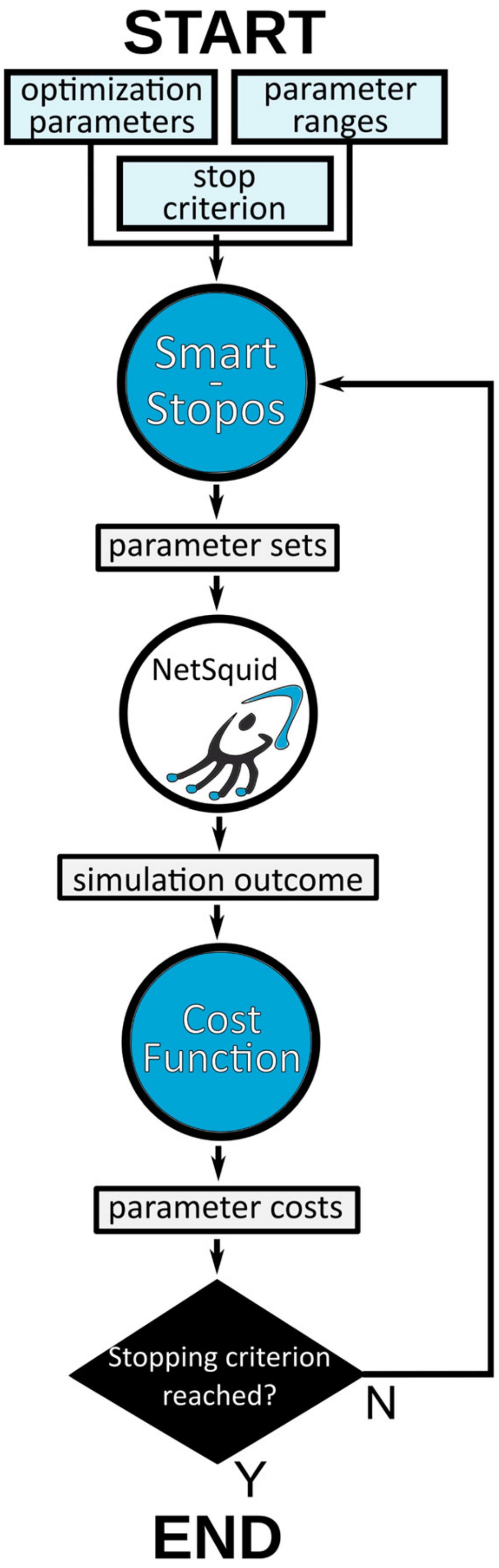


Figure 2. Overview of our optimization process. The user inputs the desired optimization parameters, their ranges and a stopping criterion. *Smart-stopos* generates sets of parameters in the allowed range and feeds them to the NetSquid simulation. The outputs of the simulation are used to compute the cost associated to each parameter set, which in turn is used by *smart-stopos* to generate new parameter sets. This process is repeated until the stopping criterion is reached. In our particular case, the optimization parameters are the parameters defining the abstract repeater model introduced in section 2.3, the relevant simulation outputs are the fidelity and generation rate of end-to-end entangled states and the cost function is the one defined in section 2.2.

swap introduces depolarizing noise parametrized by a swap quality s_q and that memory decoherence is described by a T_1, T_2 process, with T_1 (T_2) being the memory's relaxation (dephasing) time. In simple terms, this means that T_2 determines how fast the off-diagonal components of the density matrix decay, whereas T_1 defines how long it takes for a quantum system to relax to its lowest energy state. For more details on T_1, T_2 noise processes see appendix C, where our parametrization of depolarizing noise is also clarified. We further assume that entanglement swapping, although noisy, is deterministic. We note that this model is quite abstract. It could, in principle, describe the behavior of any repeater of the processing node type, examples being NV centers and trapped ions. By this we mean that it is possible to map the parameters in a physically accurate model of an NV center or trapped ion to this smaller set of more abstract parameters. In fact, we did exactly this for NV centers in order to validate this model, as laid out in appendix C. It is important to note that in this mapping we considered induced dephasing noise instead of the usual memory dephasing. Furthermore, atomic ensemble based repeaters could be described by considering non-deterministic entanglement swaps and enriching the model with a swap success probability parameter, but this lies beyond the scope of this work.

We again highlight that this model was chosen for demonstrative purposes, and that our methodology could just as well be applied to more realistic hardware models, as discussed in section 5.

2.4. Genetic algorithms

Evolutionary algorithms (EAs) have been shown to have an advantage over conventional gradient-based methods in finding global minima in multimodal functions whose search space is not well known [28], although we stress that this is not guaranteed. They are also robust to noise in data and easy to parallelize. There are multiple approaches within the umbrella of EAs, with prominent examples being GAs [29], evolution strategy [30], differential evolution [31] and particle swarm optimization [32, 33]. In this work we have used GAs, a search heuristic inspired by the theory of evolution. We limit ourselves to a high-level overview of GAs. For a comprehensive introduction, we direct the interested reader to [34].

We start with a population of randomly generated individuals. In our case, each individual in a population is a set of values for the parameters of the abstract model introduced in section 2.3. The GA generates new individuals in an iterative process, with each iteration being known as a generation. In each generation, the cost function is evaluated for every member of the population, the resulting value being known as the fitness. A subset of the population is then selected according to a fitness-dependent rule, in which higher-fitness solutions are more likely to be chosen. New individuals are then generated through random crossover and mutation operations. The new population is used for the following iteration of the algorithm, meaning that the simulation is run with the new individuals (i.e. sets of abstract model parameters) as input and the cost function is computed using the simulation outputs. The algorithm can terminate after a set number of generations or once some predefined condition is attained. For the examples given in this work, we have chosen to use the first condition and let the algorithm terminate after a preset number of generations, typically 150. Exploration of the search space is assured by the crossover and mutation-driven recombination of solutions, whereas fitness-based selection ensures exploitation of minima.

GAs come in several different flavors. See appendix B for details on our particular implementation.

2.5. Smart-stopos

The simulation tools we use are computationally heavy and produce large amounts of data. In order to make good use of them and extract useful information from said data, we need a systematized way of feeding input parameters to the simulations in batches, run the simulations on a HPC cluster using *stopos* [35], feed the outputs to the optimization algorithm and iterate this procedure. To these ends, we made use of *smart-stopos* (freely available at [24]), a set of tools we developed to allow for parameter exploration and optimization, both locally and in an HPC setting. *Smart-stopos* can be seen as an addition to *stopos*, extending its capabilities by allowing for the seamless integration of simulation and optimization tasks. We used GAs in this work but in principle any other algorithm could be plugged in, provided that it can be run with only simulation inputs and outputs. Furthermore, we note that we used NetSquid but our methodology could also be made to work with any other quantum network simulator. For more details on the use of *smart-stopos*, we direct the interested reader to appendix A.

2.6. Process overview

We will now show how the tools we introduced can be pieced together to answer the question of what the minimum requirements are on the quality of quantum repeaters. To that end, we show in figure 2 a diagram of the workflow of our methodology.

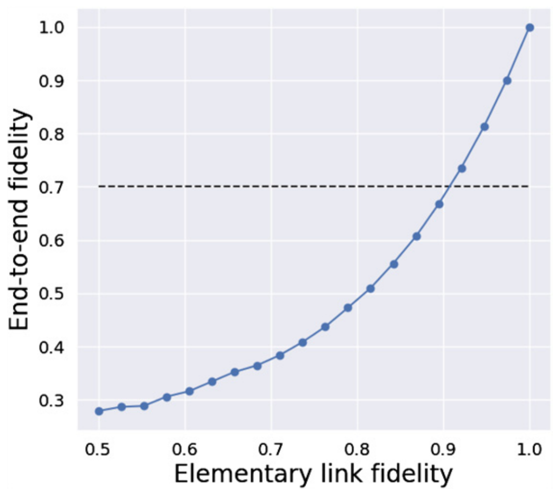


Figure 3. Variation of end-to-end fidelity across five equally spaced nodes as the elementary link fidelity is varied and the other four parameters are kept at their perfect values. The value of the elementary link fidelity that results in an end-to-end fidelity of 0.7 is at the intersection of the two lines in the plot, being just above 0.9 in this case.

The process is started by defining the parameters to be optimized and their allowed range of values. This information, together with a termination criterion, is passed to *smart-stopos*, which then randomly generates sets of parameters within the defined ranges. Each of these sets of parameters is fed to the NetSquid simulation, which outputs an end-to-end entangled state and the time its generation took, allowing us to compute the fidelity with the ideal Bell state and the entanglement generation rate. Note that these quantities are stochastic, so throughout this work we average them over multiple runs of the same setup. These metrics, together with the parameter values and the baseline values, are used to compute the cost function, as defined in equation (5). This process is then repeated for each set of parameters. The ensemble of parameter sets and respective costs are given as input to *smart-stopos*, which generates new sets of parameters using our GA. The process repeats until the termination criterion is reached. The final output is the minimum value of the cost function found by the algorithm, which in this case corresponds to an answer to the question of what are the minimum requirements on a quantum repeater.

Figure 2 makes the modularity of our approach clear. Any of the building blocks of our process, namely the optimization algorithm used by *smart-stopos*, NetSquid simulation and cost function, can be swapped out without changes to the overall workflow. For example, if we wanted to apply our methodology to a simulation of a repeater chain of trapped ions, all we would have to do would be to replace our abstract model NetSquid simulation for an appropriate trapped ions simulation. Similarly, to answer a different optimization question one just has to redefine the cost function.

2.7. Challenges in applying GAs to quantum systems

We came across some challenges when applying GAs to simulations of quantum systems. Some of these were of a practical nature, and others were more fundamental. We will now give an overview of what these issues were, and how we overcame them.

2.7.1. Practical challenges

We came across two practical challenges: (i) the size of the parameter space and (ii) the amount of data generated. (i) Is due to the complexity of quantum repeater modeling. In general the search space may be big, but in our illustrative example of the abstract model introduced in section 2.3 it is manageable. We nevertheless introduced a pre-processing procedure for restricting the parameter space, as we believe it would be useful when considering use cases with larger parameter spaces. This procedure consists of performing sensitivity analysis for each of the five parameters individually, i.e. holding four parameters constant and running simulations varying the fifth one from its baseline value to its perfect one. As an example of how this can reduce the search space, we show in figure 3 the variation of the end-to-end fidelity with the elementary link fidelity when all other parameters are kept at their perfect values. The optimal set of parameters for this setup will certainly contain less-than-perfect values, so the elementary link fidelity of this set will be higher than the one found using this sensitivity analysis, so we can safely restrict the search space for this parameter in GA optimizations runs to the interval $[f_{\text{perf}}, 1]$, where f_{perf} is the elementary link fidelity that results in an end-to-end fidelity of 0.7 when all other parameters are perfect.

Another practical challenge is the sheer amount of data that is produced. For each setup we consider we run our simulations for hundreds of different sets of parameters at each optimization step, with each set of parameters being in turn run a hundred times. In order to systematically and efficiently process all of this data, we developed *smart-stopos*, as detailed in section 2.5.

2.7.2. Fundamental challenges

Fundamental challenges occur due to the fact that quantum systems produce inherently non-deterministic outputs. This can be problematic if the cost function has terms that are step functions, which is our case. For a concrete example, let us say that in generation 34 of the optimization procedure, the GA found a set of parameters that result in an entanglement generation rate of 1.05 Hz, just above the desired threshold. In generation 35, this parameter set would again be fed into the simulation. However, this time around, due to statistical fluctuations, the simulation outputs an entanglement generation rate of 0.99 Hz, just below the threshold. Since the cost function defined in equation (6) assigns a very high cost to any solution that does not attain the performance metrics, this solution would in generation 35 have a very high cost function value. This means that it would almost certainly not be chosen as a parent for the following generation, and the algorithm would effectively lose it. This is a problem, as it results in the algorithm losing a good solution and potentially wasting computation time finding it again.

There are several possible solutions to this problem. The one we chose, due to its simplicity, was to run the simulation multiple times for each set of parameters and compute the value of the cost function using the average end-to-end fidelity and entanglement generation rates. Running the simulations multiple times provides some security against statistical fluctuations, although it increases the computation time. We found empirically that running the simulation 100 times for each set of parameters represents a good trade-off between minimizing fluctuations and keeping computation times feasible.

Another possible solution that we also explored was to use a smoother function, such as a sigmoid, instead of a sharp step function. This would in principle address the problem we mentioned of a set of parameters being heavily penalized because its metrics dipped just below the targets due to statistical fluctuations. For a smoother function, such fluctuations would lead to small fluctuations in the value of the cost function. There are however some issues with this solution. Since the function is smoother, it no longer acts as a hard constraint, which is the behavior we are looking for. What we mean by this is that a solution whose performance metrics are slightly below the targets will only be lightly penalized. It might thus have a lower cost function value than a solution with better, i.e. more expensive, parameters that attains the performance metrics. In less technical terms, this translates as the cost function not being well aligned with the stated optimization goal.

This concludes the introduction of the optimization methodology we propose. The rest of the paper concerns itself with two questions: (i) is our methodology valid, addressed in section 3 and (ii) what results do we get when we apply it, addressed in section 4.

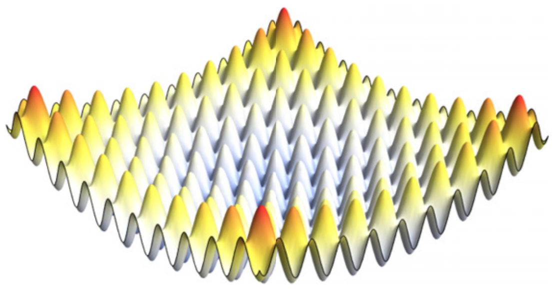
3. Validation

As we stated in the previous section, before we apply our methodology we must validate it. By this we mean that we must verify that the methodology we propose for applying GAs to simulations of quantum networks can produce meaningful results. We can see this validation as being split into two different steps. One, benchmarking the GAs i.e., evaluating how well they perform and two, validating that the methodology is sound. The first step will be accomplished by applying our specific implementation of GAs to the optimization of common benchmarking functions and comparing their performance to that of implementations found in the literature. The second step will consist of applying our methodology to a chain of evenly-spaced nodes generating Werner states, for which analytical expressions for the end-to-end fidelity and entanglement generation rate in terms of repeater parameters can be found. Having these expressions, we can compute what are the repeater parameters that minimize the cost function. If our GA approach is capable of finding this solution, we have compelling evidence that our methodology would also perform well when applied to the more realistic cases we are interested in, for which analytical results cannot be readily derived.

We have also validated the abstract model we use in our simulations against a more physically accurate model of NV center-based repeaters. These results are shown in appendix C.

3.1. Benchmarking genetic algorithms

In order to evaluate the performance of GAs and how it is affected by the algorithm's hyperparameters, several benchmarking functions have been defined [36]. These are designed to test how well each GA implementation handles cost functions with given properties. For example, if we expect the function we


(a) Rastrigin's function.

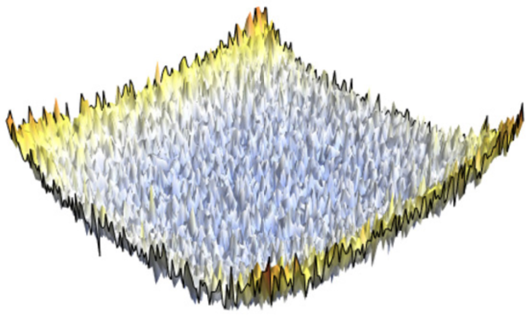
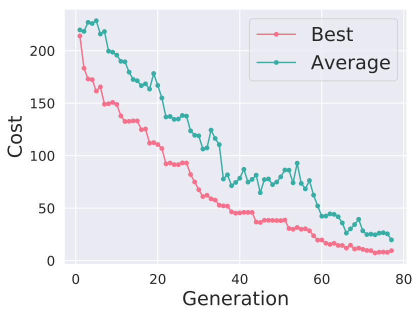
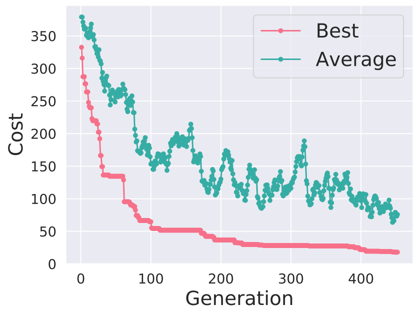

(b) Quartic function.

Figure 4. Plot of the two-dimensional versions of (a) Rastrigin's function and (b) quartic function. The multiple minima of Rastrigin's function and the noisy landscape of the quartic function can be clearly seen.



(a) Quartic function.



(b) Rastrigin's function.

Figure 5. Evolution of the cost of best solution (red) and population average (green) for the (a) quartic function and (b) Rastrigin's function over 75 and 400 generations, respectively. The data used in (a) ((b)) was acquired in roughly 1h30 (26h) on consumer-market hardware (Intel Core i7-8665U and 8 GB RAM). These runtimes can be significantly reduced via parallelization and use of HPC clusters. All costs approach zero, the global minimum of both cost functions, with the average cost being consistently higher than the best cost, as expected. This indicates that our GA implementation is capable of finding good solutions for said functions.

want to optimize to be noisy, i.e. to have the output for a given input randomly oscillate each time the function is called, we should benchmark the GA against a noisy function, such as the quartic function, defined in equation (7)

$$f_{\mathrm{q}}(\mathbf{x}) = \sum_{k=1}^{30} \left(kx_k^4 + \mathcal{N}(0,1)\right) \qquad -1.28 \leqslant x_k \leqslant 1.28, \tag{7}$$

where $\mathcal{N}(0,1)$ is a normal distribution with mean 0 and standard deviation 1. This function, plotted in the bottom half of figure 4, is a unimodal function padded with Gaussian noise. Therefore, a GA that performs poorly on it will also perform poorly on any function with noisy outputs.

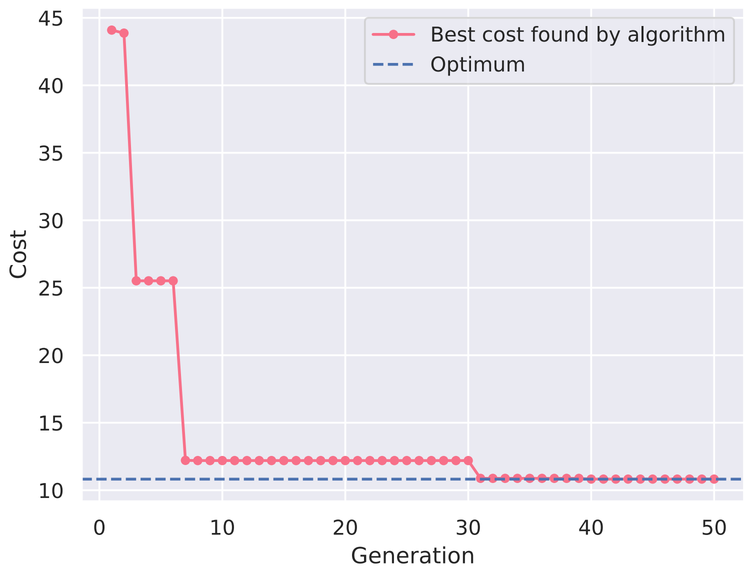


Figure 6. Evolution of the lowest value of the cost function over 50 generations. After little more than 30 generations the algorithm finds the parameter set that optimizes the cost function. This optimum is marked in the figure by a blue dashed line.

Taking this into account, we chose two functions to benchmark our GA implementations. This choice was made by taking into account which of the functions best represented the cost landscape we expect our problem to have. Since the quantum nature of our simulations implies that they will necessarily be noisy in the above-defined sense, we will choose the quartic function as a benchmarking function. Furthermore, we expect that the landscape of the cost function defined in equation (5) will have multiple local minima, corresponding to different sets of parameters that satisfy the imposed constraints on end-to-end fidelity and entanglement generation rate. With this in mind, we also chose Rastrigin's function, defined in equation (8)

$$f_r(\mathbf{x}) = 200 + \sum_{i=1}^{20} \left(x_i^2 - 10\,\cos\left(2\pi x_i\right)\right), \quad -5.12 \leqslant x_i \leqslant 5.12 \tag{8}$$

For illustrative purposes, the two-dimensional version of Rastrigin's function is shown on the top half of figure 4. It can be seen that it has a very bumpy landscape, with a global minimum at **0**, in the center of the plotted region. Its many local minima render it a challenging benchmark for GAs. We applied our GA implementation to both of these functions, with the results being plotted in figure 5. The hyperparameters used for these optimization runs were chosen according to the guidelines given in [36] and population selection was done using the Roulette Wheel method [37]. For an explanation of the Roulette Wheel method we point the interested reader to appendix B. By best value we mean the lowest value of the cost function achieved by any of the parameter sets in the population. Similarly, by average value we mean the average of the costs of all parameter sets in the population. We see that, for both functions, the average cost and the best cost at each generation approach their global minimum, 0. Furthermore, the performance of our implementation is in line with that of those in [36], which indicates that our GA is capable of handling both noisy and multimodal functions. We note that convergence requires significantly more generations for Rastrigin's function than for the quartic function. This reflects the well-known fact [36] that multimodal functions are challenging for GAs. We must also note that we could, by further tuning some of the algorithm's hyperparameters, obtain a marginally better performance on these benchmarking functions. However, since our goal is only to verify that our implementation is correct and performs reasonably well for the type of cost landscapes that we expect to encounter, we abstain from doing so.

3.2. Validating on Werner chains

The previous section focused on benchmarking the performance of the GA, but the question of whether applying GAs to repeater chain optimization problems can produce good results remains. In order to answer it, we consider the simple scenario of a chain of 3 nodes generating Werner states, and we pose the question of what are the worst parameters that can deliver an end-to-end entangled pair of fidelity 0.6 every second. Similarly to the abstract model presented in earlier sections, the nodes in the chain generate elementary links of fidelity F_{EL} with success probability p_{suc} and depolarizing noise parametrized by s_{q} is applied after entanglement swaps. This is a problem for which we can analytically find expressions for the

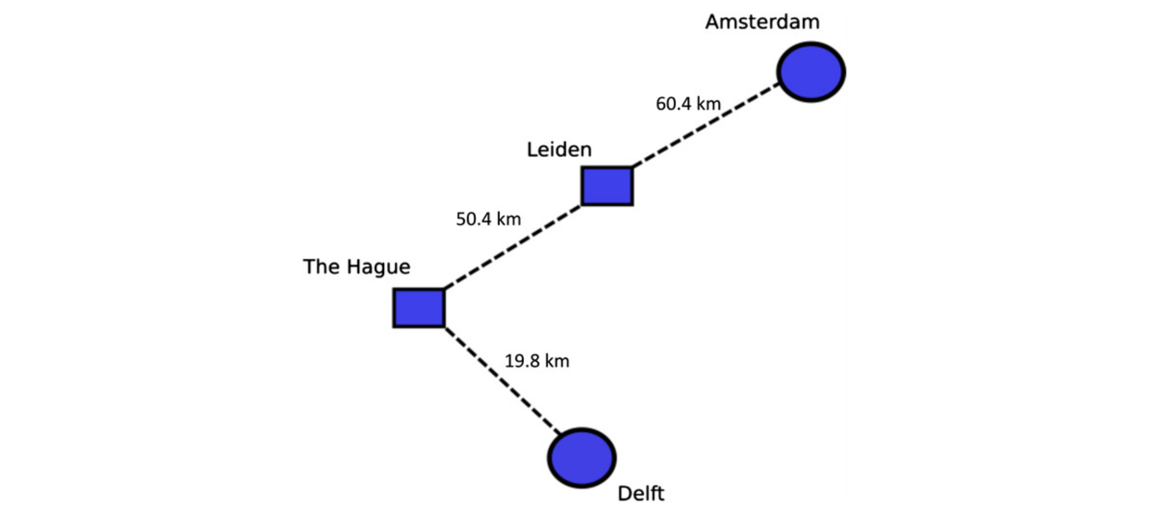


Figure 7. Visualization of the quantum network we will consider. The end nodes, represented by circles, are placed in Delft and Amsterdam. The repeater nodes, represented by squares, are placed in The Hague and Leiden. The placement of the nodes roughly approximates their actual geographical location and the length of the fibers connecting them is included for reference.

end-to-end rate and fidelity, and thus for the ideal value of the cost function. We expect that the structure of this problem is similar to that of the one we want to tackle. By this we mean that we expect its cost landscape to show some of the same features as our target problem, namely multiple minima and noisiness. Therefore, despite being simpler, good performance in this problem should indicate that our approach is valid. For details of how we derived analytical results for this setup we defer the interested reader to appendix D.

In figure 6, we show the evolution of the cost of the best individual in each generation obtained by applying the GA-based method to the setup we described. Also present in the plot, in a dashed line, is the optimum cost.

The cost function drops to the global optimum at around the 30 generation mark, indicating that the algorithm is capable of finding the worst set of repeater parameters satisfying the benchmarks we set. This is then a good indicator that our methodology is well-suited to the optimization of entanglement generation in repeater chains.

4. Evaluation: use cases

Having validated our methodology, we applied it to two use cases demonstrating its power and potential usefulness. In the past decade, NV centers have been demonstrated to be capable of generating remote entanglement between matter memories with long coherence times [6, 11, 38], establishing them as promising candidates for the realization of scalable quantum repeaters [10]. A better understanding of hardware requirements would then be useful in illuminating the path toward scalable NV-based quantum repeaters. We thus used the abstract model of NV-type states that we introduced in section 2.3 in the simulations of all use cases. We furthermore chose to consider, for simplicity, SWAP-ASAP protocols with no memory cut-offs. More precisely, we simulate the protocol introduced in appendix E2 of [15], which proceeds as follows: we assign indices to each node going from left to right in the chain and starting with 1. Even-numbered nodes are called initiators, whereas odd-numbered nodes are called responders. As the name implies, initiators are responsible for initiating the process of entanglement generation, which they do by sending a request for entanglement generation to their left-hand neighbors and waiting for a response. Once the responders respond, the process of entanglement generation begins. This is simulated by sampling the time taken to generate entanglement according to the success probability parameter and the cycle time, which determines how long a single entanglement generation attempt takes. Once entanglement is successfully generated, the initiator proceeds to attempt to generate entanglement with the right-hand neighbor and the process unfolds in the same way. Whenever a node holds two entangled qubits in hand, it performs an entanglement swap by measuring them in the Bell basis. The simulation stops once the end nodes of the chain share an entangled pair.

Another roadblock in the way of the quantum internet is that even when quantum repeater technology is at deployment stage, it is expected that it will be very costly. One way of rendering the implementation of quantum networks more cost-effective is to take advantage of preexisting infrastructure by using previously

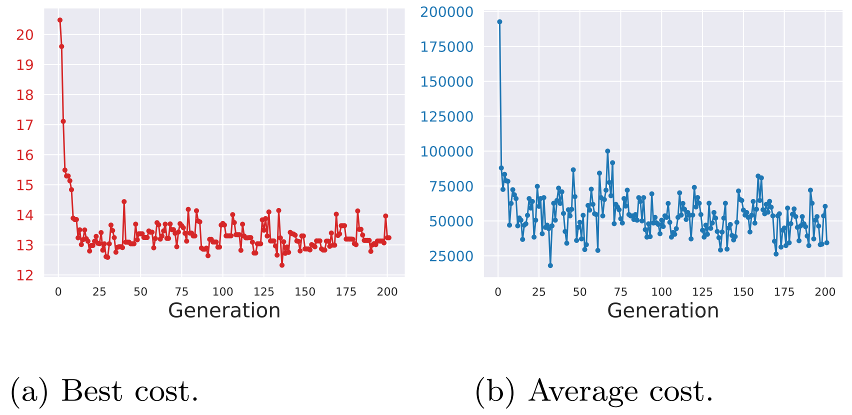


Figure 8. Evolution of the (a) best and (b) average values of the total cost function for the use case discussed in section 4.1.1. The best value converges to roughly 13, whereas the average oscillates around 50 000 (note the different scales). The high values of the average cost throughout the optimization process are due to the mutation process, which sometimes produces solutions that do not fulfill the target metrics. Our simulation was run 100 times for each individual.

deployed optical fiber networks [39]. With this in mind, we used real-life fiber data of the Netherlands. This was made available to us by SURF, a classical network provider for Dutch education and research institutions. We considered a repeater chain with nodes in Delft, The Hague, Leiden and Amsterdam, as depicted in figure 7, as this is an example of a possible near-term quantum network in the Netherlands. We use real fiber length and attenuation in our simulations. We chose Delft and Amsterdam as the end nodes of the chain as out of these four cities they are the most distant pair. The baseline values used for computing the value of the cost function for each set of parameters were obtained from actual state-of-the-art experimental results using NV centers. The process through which we converted these experimental results to our abstract model parameters is described in detail in appendix E1. We set as performance targets end-to-end fidelity $F_{\text{min}} = 0.7$ and end-to-end entanglement generation rate $R_{\text{min}} = 1$ Hz. The value of F_{min} was chosen to ensure that we remain in the regime where the agreement between the abstract model and the detailed NV model is good (see appendix C for details). Besides this practical argument, there is no strong reason to pick a particular number for the fidelity or the rate. These numbers are simply examples, meant to show how our methodology can find the minimal hardware requirements satisfying them.

In order to study the effects of internode distance, chain length and number of repeaters we further applied our methodology to chains of equally spaced nodes with varying numbers of repeaters. In one case, we kept the internode distance fixed, and in the other we kept the total length fixed as we varied the number of repeaters. More concretely, we considered (i) a chain of equally spaced nodes spanning 800 km and (ii) a chain with an internode distance of 100 km. For each of these, we considered the cases of 3, 5, 10 and 12 repeater nodes. The baseline parameter values are computed in the same manner as in the previous use case, so we again defer to appendix E1 for details. We also consider the same target performance metrics as in the previous use case.

4.1. Results

4.1.1. Real network

We will now show the main results obtained by applying our methodology to the network introduced in figure 7.

In figure 8 we show the best and average values of the total cost function (equation (6)) as a function of the optimization step. Contrasting with figure 5, we see that (i) the average value of the cost function remains significantly higher than the best value and that (ii) the best value per generation oscillates. The first observation is explained by the combination of the inherent randomness of the GA and the fact that we used step functions for the cost. A GA generates new candidate solutions through a process of mutation and recombination, as detailed in appendix B. While these processes allow for a thorough exploration of the parameter space, they may also produce solutions that fall outside the defined target metrics. The step functions in the cost ensure that such solutions will be heavily penalized, explaining the high average values of the cost function in figure 8. The second observation is also explained by a combination of two factors, namely the already mentioned step functions in the cost and the non-deterministic nature of our simulations. Since across different simulations for the same set of parameters there are fluctuations in the values of the end-to-end metrics, it might happen that these sometimes dip below the predefined targets. Due to the step function, the cost associated to this particular set of parameters will become much higher,

Table 1. Experimentally-derived baseline parameter values and values of the best solution found using our methodology for the use case discussed in section 4.1.1. The biggest relative increases happen for T_2 and p_{suc}, suggesting that improving these parameters is key for achieving scalable NV-based repeaters.

	F_{EL}	p_{suc}	s_{q}	T_1	T_2
Baseline	0.9698	0.004 600	0.8590	10 h	4.9 ms
Solution	0.9806	0.097 70	0.9414	10.23 h	22.79 ms

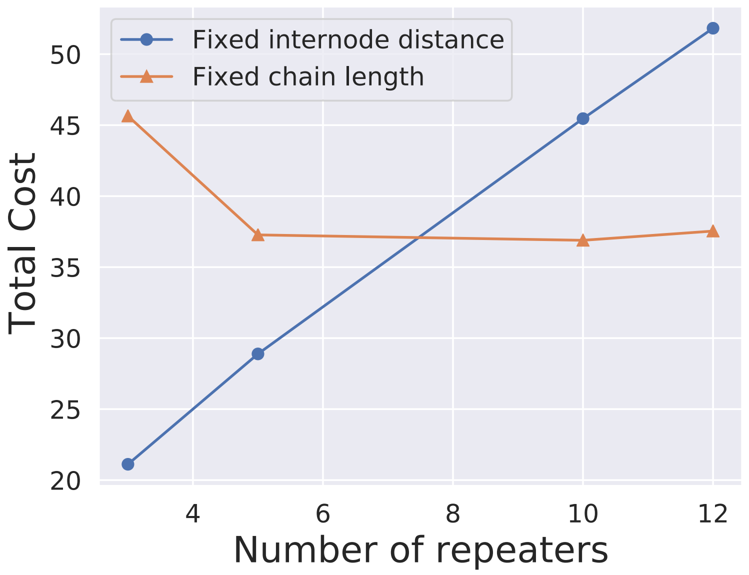


Figure 9. Total cost, as defined in equation (6), of the best solutions found by our GA for setups with varying number of repeaters. The cost grows linearly for FID. There is no discernible pattern for FCL. Each data point corresponds to the best solution found after 200 generations, with 150 population individuals per generation and 100 simulation runs per individual.

meaning that it will no longer be the best solution. This effect can be minimized by running our simulations multiple times for each set of parameters, as discussed in section 2.6.

In table 1 we show the parameters of the best solution found using our methodology. For comparison purposes, we also show the baseline values we considered. The biggest relative increases are in p_{suc} and T_2, suggesting that induced dephasing noise is the biggest hurdle in the way of NV-based repeater technology. On the other end of the spectrum, the solution's T_1 value is barely higher than that of the baseline, indicating that T_1 coherence times in NV centers are already long enough.

4.1.2. Equally spaced nodes

We now show the main results obtained by applying our methodology to repeater chains of equally spaced nodes with different numbers of repeaters. To study how the overall length of a chain and the internode distance affect the solutions found, we considered two cases: (a) fixed chain length (FCL) and (b) fixed internode distance (FID). For both FCL and FID we applied our methodology to chains of 3, 5, 10 and 12 repeater nodes. We note that each data point in the plots shown in this section corresponds to the best solution found after 200 generations, with 150 population individuals per generation and 100 simulation runs per individual. Running our optimization procedure once with these parameters takes roughly 46 hours locally using consumer-market hardware (Intel Core i7-8665U and 8 GB RAM), underlining the need for access to HPC centers. In fact, by using such a center, the computation time can be reduced to 2 hours (using 2 nodes of the HPC center, each endowed with 64 GB of memory and 24 cores with CPU E5-2690). We note that the vast majority of this time is taken by quantum repeater simulations, with the time needed by the GA being negligible in comparison. We further note that, as shown in figure 9 of [15], the runtime of repeater chain simulations using NetSquid grows linearly with the number of nodes in the chain. This implies that our method remains applicable for chains that are significantly longer than the ones considered in this work.

In figure 9 we show how the total cost of the best solution found varies with the number of repeaters in both cases. We observe a linear growth of the FID cost with the number of repeaters, which is not

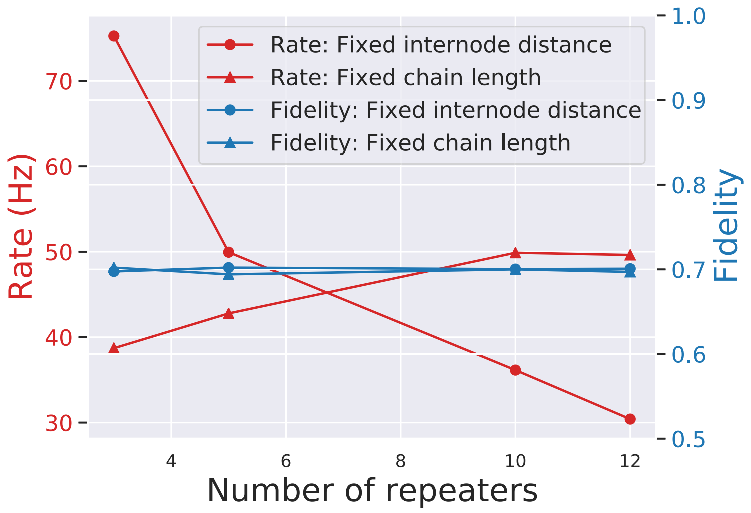


Figure 10. Comparison of the metrics characterizing the best solutions found by our GA for each of the different setups. The end-to-end fidelity is very close to the goal of 0.7 we defined, for both FID and FCL. On the other hand, the end-to-end entanglement rate is well above the 1 Hz goal for both cases. For FID, it decreases from roughly 80 Hz in the 3 repeater node setup to about 30 Hz in the 12 repeater node setup. For FCL, it increases slightly from 40 Hz in the 3 repeater node setup to 50 Hz in the 12 repeaters setup. Each data point corresponds to 100 runs of the simulation. The error bars are smaller than the markers.

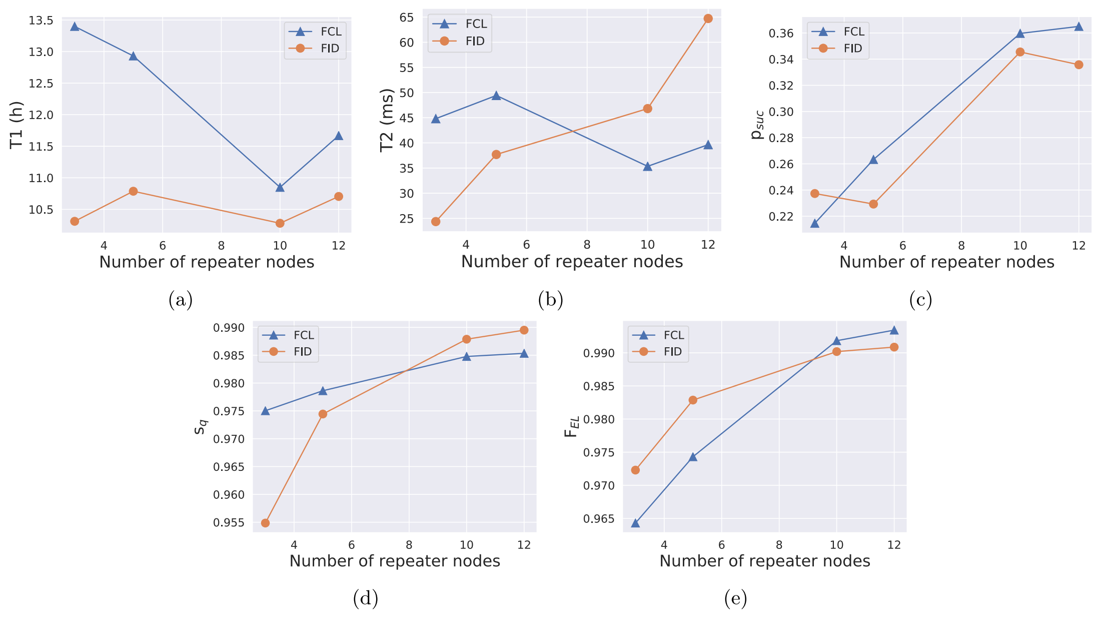


Figure 11. Parameters of the best solutions found for FCL and FID with different numbers of repeaters. Each data point corresponds to the best solution found after 200 generations, with 150 population individuals per generation and 100 simulation runs per individual. For a detailed discussion of these results, see the text in section 4.1.2.

surprising: fixing the internode distance but increasing the number of repeater nodes corresponds to increasing the total length covered. In fact, the leftmost data point in figure 9 corresponds to a chain spanning 400 km, whereas the rightmost is associated to a chain spanning 1300 km. We would expect connecting end nodes that are further apart to be a greater challenge due to the exponential growth in photon losses, which necessitates repeater parameters of higher quality. This does not apply to the FCL use case. All the data points in the associated curve correspond to a repeater chain that spans 800 km and we observe in figure 9 that the cost is slightly higher for the three-repeater setup. It was not *a priori* obvious that this would be the case. A smaller number of repeaters implies that the swap quality and fidelity of the elementary link do not need to be as good, as there will be fewer swaps and hence less fidelity loss. On the other hand the elementary links are longer than in a setup with many repeaters, so the associated baseline values are worse (see appendix E1 for details). Any improvement then requires a higher parameter cost, as per equation (5).

To further explore how the solutions found vary, we plot in figure 10 the end-to-end fidelity and entanglement generation rate of these solutions against the number of repeaters in the chain. We see that, for both use cases and all numbers of repeaters, the end-to-end fidelity is very close to 0.7. On the other hand, the rate decreases from around 80 Hz to 30 Hz as the number of repeater nodes increases from 3 to 12 at FID and it increases slightly from 40 Hz to 50 Hz with the number of repeaters at FCL. While the fidelities obtained are what we expected, since the limit we imposed via the cost function was 0.7, the same is not true for the rates. The penalty term we added to the cost function only comes into effect if the rate drops below 1 Hz, so there is no benefit in terms of the cost to have a solution that results in a rate of e.g. 50 Hz versus one of 1 Hz. We would thus expect the best solutions to have rates close to 1 Hz, which was not the case.

In order to explain this, we note that T_2 and p_{suc} are inextricably linked. T_2 reflects the intensity of the induced dephasing effect, (see appendix C) with a higher value of T_2 corresponding to a weaker induced dephasing effect, and vice-versa. This type of noise is applied every time entanglement generation is attempted. Therefore, its intensity heavily depends on p_{suc}: a lower success probability implies more entanglement generation attempts and thus more dephasing. One would naively think that the GA would always converge toward a solution with lower rate (R) up until the limit of 1 Hz we defined, as that would allow for lower values of p_{suc} and hence a lower value of the parameter cost. However, due to the connection between p_{suc} and T_2, a lower value of the former necessitates a higher value of the latter. This then implies that solutions whose R is closer to the established requirement of 1 Hz, with their lower values of p_{suc}, might actually have higher costs than solutions with higher R, accounting for why the ideal solutions have such high rates.

To conclude our analysis of the solutions found with our optimization procedure, we present in figure 11 the values of each of the parameters in the solutions found for each setup. Starting with the top row, we note that the relative variations of T_1 for different setups are small when compared to the ones of T_2. Similarly to what we saw in the use case of section 4.1.1, this indicates that T_1 is not a crucial parameter to improve for NV center-based repeaters. We note also that for FID, T_2 grows with the number of repeaters, whereas it remains roughly constant for FCL. This is again explained by the fact that in the first case the total distance covered increases with the number of repeaters, so one expects that longer coherence times will be required. Regarding p_{suc}, we observe that it tends to be higher for chains with more repeaters, reflecting the fact that in order to achieve similar end-to-end rates across longer chains, one cannot afford to spend as much time generating elementary links as in shorter chains.

We move now to the bottom row, whose plots concern F_{EL} and s_{q}. Both increase with the number of repeaters, approaching 1. This was to be expected, as a higher number of repeaters implies more entanglement swaps and hence more decay in fidelity. Therefore, to reach the same end-to-end fidelity one needs better elementary links and swaps. We further note that for few repeaters, F_{EL} is higher and s_{q} is lower at FID than at FCL. The opposite is true for many repeaters. We believe this may be explained by the length of the elementary links in the FCL case. For few repeaters, the FCL elementary links are longer than the FID elementary links (133–200 km vs 100 km), with the situation being reversed for many repeaters (73–89 km vs 100 km). A longer elementary link translates into a worse baseline value of F_{EL}, as detailed in appendix E1, and thus more expensive improvements. On the other hand, the baseline value of s_{q} is the same irrespective of the elementary link length, and thus so is the cost of improving it. Therefore, for few repeaters the less costly solution at FCL has a lower elementary link fidelity and higher swap quality than the the less costly solution at FID. The opposite is true for many repeaters, explaining the observed behavior.

5. Conclusions

We have introduced a methodology for the optimization of entanglement generation and distribution in repeater chains using GAs. In contrast with previous work in this area [18–22], our methodology is systematic, modular and broadly applicable. We validated it by benchmarking our GAs on functions commonly used for this purpose and by applying it to a repeater chain generating Werner states. We can derive analytical results for such a chain and thus gauge how well our methodology performs. Having validated our methodology, we applied it to three use cases. First, we considered a repeater chain built using real-life fiber data, thus demonstrating that our methodology can go beyond simple network topologies. The other two use cases consisted of chains of equally spaced nodes for which we varied the number of repeaters. In one we kept the internode distance constant, and in the other we fixed the total chain length. By applying our methodology to these use cases we found what are the worst parameters achieving end-to-end fidelity and rate of at least 0.7 and 1 Hz, respectively, in different scenarios. Even though this was the question we focused on answering in this work, we must note that our methodology is more general and can be applied to a variety of problems, given that they can be restated as optimization problems and that an appropriate cost function is designed.

On a similar note, we must again stress that even though we have here focused on a simplified five-parameter repeater model, in no way is our methodology restricted to such a model. In fact, one interesting application of our methodology would be to consider a more realistic hardware model, such as the one proposed in [40] for NV-center based repeaters. Such models are described by a very large number of parameters, on the order of 30 in this case, which means that the initial search space is too large for a direct application of our methodology. To practically apply our methodology to such a large parameter space, one could opt for a two-stage optimization process. The first stage would be similar to what was shown in this work, i.e. applying the methodology to a simpler model that can be mapped to the more accurate one. This step would allow us to both reduce the search space by finding minimal requirements on parameters and to identify which of these parameters have a bigger impact on the target metrics. With this knowledge in hand, we could apply the methodology to a select subset of parameters in the more detailed model, performing the optimization procedure in a reduced, more feasible search space. The outcome of this two-step procedure would then be a realistic picture of what kind of hardware improvements are required to achieve long-range entanglement, constituting a useful guide for experimental groups working on repeater technology. This establishes the methodology we have proposed as an invaluable tool for the development of a blueprint for the quantum internet.

Acknowledgments

We thank Guus Avis, Kaushik Chakraborty, Axel Dahlberg, Hana Jirovská, Rob Knegjens and Julian Rabbie for insightful comments and discussions. We thank also Julian Rabbie and Matthew Skrzypczyk for critical reading of the manuscript. We further thank SURF for sharing data regarding their fiber network. This work was supported by the QIA-project that has received funding from the European Union's Horizon 2020 research and innovation program under Grant Agreement No. 820445, and also by an ERC Starting grant and NWO Zwaartekracht QSC.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: https://gitlab.com/FranciscoHS/optimizing-entanglement-generation-and-distribution-using-genetic-algorithms-data.

Appendix A. Smart-stopos

In figure 12, we present a detailed overview of the *smart-stopos* workflow. The user must provide a script, entitled program.py in figure 12, that runs the simulation and an input_file.ini that contains information about the optimization procedure, such as the number of iterations and parameter specifications. Given these inputs, *smart-stopos* generates sets of parameters for which the simulation will be run according to the specifications given in input_file.ini. The outputs of the simulation are then used to generate a new set of parameters for the next iteration. This generation is done in an algorithm-dependent way. We used GAs in this work but in principle any other algorithm could be plugged in, provided that it can be run with only simulation inputs and outputs.

Appendix B. Genetic algorithms

In this appendix we give a detailed view of the GA implementation we used for the simulations described in this work.

We started by transforming all parameters to be in the $[0, 1]$ range. This is trivial for the elementary link fidelity, success probability and swap quality. For T_1 and T_2, which usually live in the $[0, \infty]$ range, we performed the following transformation:

$$T' = \begin{cases} \dfrac{1}{T+1} & \text{if } \mathrm{T} > 0 \\ 0 & \text{o.w.,} \end{cases} \tag{B1}$$

which results in $T' \in [0, 1]$, as required. A chromosome, i.e. a set of parameters constituting a candidate solution, is thus a set of 5 real numbers in the $[0, 1]$ interval.

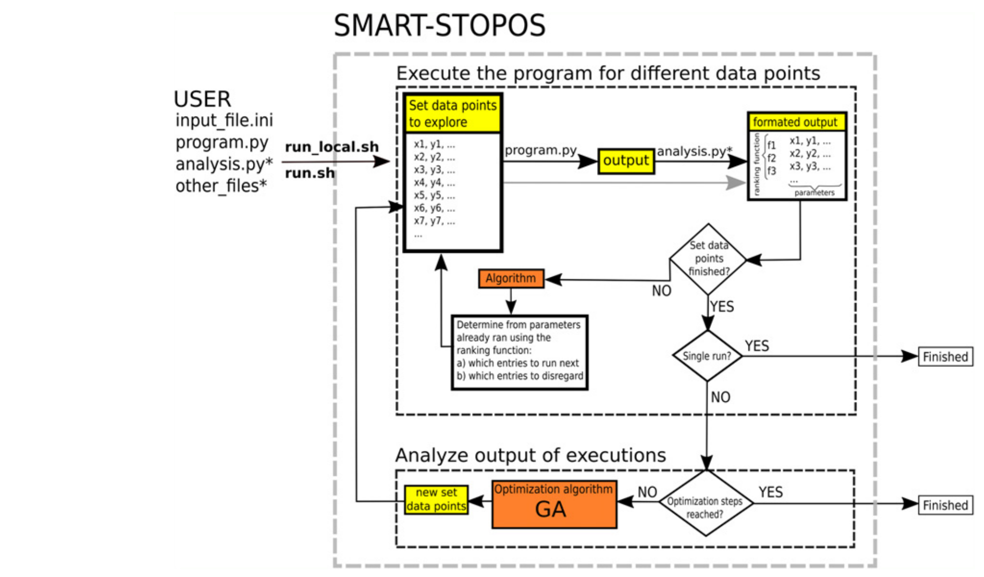

SMART-STOPOS
USER
input_file.ini
program.py
analysis.py*
other_files*
run_local.sh
run.sh
Execute the program for different data points
Set data points to explore
program.py
output
analysis.py*
formated output
ranking function
parameters
Set data points finished?
NO
YES
Algorithm
Determine from parameters already ran using the ranking function: a) which entries to run next b) which entries to disregard
Single run?
YES
NO
Finished
Analyze output of executions
new set data points
Optimization algorithm GA
Optimization steps reached?
NO
YES
Finished

Figure 12. Diagram of the *smart-stopos* workflow for parameter optimization. input_file.ini is used to define optimization parameters such as algorithm to be used, parameters to optimize and allowed range of values. The execution of a simulation (program.py) and the optional post-processing of resulting data (analysis.py) can be done locally (run_local.sh) or using HPC facilities (run.sh). The stopos [35] job manager tool is required for running on HPC facilities. Files marked with $*$ are optional.

We used populations of 150 individuals, as the literature suggests that numbers of this order of magnitude are enough to get adequate parameter space exploration while still being computationally feasible [34].

After the cost function is computed for all members of the population, we select 10 of them, 20% of the total population, according to the Roulette Wheel method [34]. Again, the literature indicates that the percentage of selected individuals should be of this order of magnitude and we empirically verified that this value produced the best results for our particular use case. One of the major challenges in GA-based optimization is to balance exploration of the search space with exploitation of known minima. If the algorithm performs selection in a purely random manner, it is no different than random search. On the other hand, if it simply selects the best individuals in a given generation, the population will tend to get stuck in local minima and be vulnerable to premature convergence. The Roulette Wheel selection method is a well-known approach to this problem, balancing exploration and exploitation by assigning selection probabilities to individuals biased, but not completely determined, by their fitness value. Applying this method to a maximization problem, the probability p_i of individual i being selected is given by:

$$p_i = \frac{f_i}{\sum_j f_j}, \tag{B2}$$

with f_j being the value of the fitness function for individual j. The probability of selection is then proportional to how a big of a share of the total fitness the individual's fitness represents, i.e. how good it is in comparison to its peers. Our problem is, however, one of minimization, not maximization. Therefore, we adapted this method by simply inverting the values of the fitness function.

Crossover is subsequently applied on the 10 selected members of the population, known as parents. This is done by randomly choosing two of the parents, sampling a crossover point, and mixing the two accordingly. To give a concrete example, if the chromosomes of the two parents are given by $[a_1, a_2, a_3, a_4, a_5]$ and $[b_1, b_2, b_3, b_4, b_5]$ and the crossover point was 2, the resulting child would have chromosome $[a_1, a_2, b_3, b_4, b_5]$. The number of children generated in this way is given by the crossover parameter, a hyperparameter of the algorithm defining how often crossover happens, times the desired population size.

The parents plus the children resulting from the crossover process are then mutated. In this process, all chromosomes of a given member of the population are randomly changed by some value that keeps them inside their range. For the mutation probability of a given parent, we implemented the adaptive scheme introduced in [41], which was shown the reduce the likelihood of corrupting a high-quality solution and enhance the exploratory properties of the algorithm. In this scheme, the probability of parent k being mutated is given by:

$$p_{\mathrm{m}} = \begin{cases} 0.5 & \text{if } c_k > \bar{c} \\ 0.5\dfrac{c_k - c_{\min}}{\bar{c} - c_{\min}} & \text{o.w.,} \end{cases} \tag{B3}$$

where c_k is the value of the cost function for parent k, $\bar{c}$ is the value of the cost function averaged over the previous generation's population and $c_{\min}$ is its minimum value. For the children generated in the crossover process, for which there is no cost value yet, the mutation probability is a hyperparameter of the algorithm. Previous work suggests that a high cross over parameter and low mutation probability produce good results [34], so we used a crossover parameter of 0.7 and a mutation probability of 0.02 to obtain the results showed in this work.

Since generation of new individuals is to some extent probabilistic, the size of a generation can vary. To keep our population size fixed, we either randomly remove elements or add some of the best members of the previous generation. We also implement a form of elitism, meaning that the best element of the previous generation is always preserved in the following generation, in order to prevent the algorithm wasting time searching for solutions it has already found [42].

We have empirically determined that 200 generations are usually enough to achieve satisfying solutions while still being computationally feasible on a cluster.

Appendix C. Abstract model validation

In this appendix we show how we validated the abstract model against a physically-accurate NV model.

C1. Matching to NV model

In order to ensure that the simulations of the abstract model can contribute to our understanding of actual physical implementations of quantum repeaters, we must verify that this abstract model captures the relevant physics to a reasonable extent. To do so, we will compare the results of simulations of a repeater chain in the abstract model with those of a repeater chain running a physically accurate model. For this purpose, any model of a physical system being studied as a possible platform for quantum repeaters would do. We will thus focus on one such system, namely NV centers, modeled as described in [43]. This is a very detailed model that accurately captures the physics of NV centers, including for instance modeling the photon emission, capture and detection processes as well as differentiating between communication and memory qubits, with all the restrictions that entails. In contrast, the simplified model we consider abstracts away all of the subtleties of photon emission and detection into an overarching success probability and treats all qubits as equal. Another key difference is that in the NV model the parameters are not mutually independent e.g. there is a relation of inverse proportionality between the fidelity of the generated entangled states and the rate at which they are generated due to the fact that both of these parameters depend on the bright state population. On the other hand, in the abstract model we make the simplifying assumption that all parameters are independent from one another. We must however emphasize that this does not reflect a limitation of our method. Taking the constraints arising from interparameter dependence into account would be possible, but we chose not to consider any such constraints in this preliminary study.

More concretely, we will perform the validation of the abstract model by taking a set of parameters describing an NV center in the model, converting it to the five parameter set that defines our model, running both simulations, and checking how the end-to-end fidelity and entanglement generation rate compare.

We start by proposing a mapping from the NV model in [43] to the five-parameter abstract model we introduced in section 2.3. We assume that elementary link states generated in the abstract repeater chain are of the form:

$$|\phi\rangle\langle\phi| = F_{\mathrm{EL}}|\psi\rangle\langle\psi| + (1 - F_{\mathrm{EL}})|\uparrow\uparrow\rangle\langle\uparrow\uparrow|, \tag{C1}$$

where $|\psi\rangle\langle\psi|$ is the ideal Bell state, F_{EL} is the elementary link fidelity and $|\uparrow\uparrow\rangle\langle\uparrow\uparrow|$ is given by:

$$|\uparrow\uparrow\rangle\langle\uparrow\uparrow| = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

The overlap between $|\psi\rangle$ and $|\uparrow\uparrow\rangle$ is 0, so F_{EL} is in fact the elementary link fidelity, the sole parameter defining elementary link states. To map states from one model to another we compute the fidelity of the NV state described in the appendix of [43] and use the result to define the abstract model state as in equation (C1). The probability of successfully generating these elementary links is obtained in an identical manner.

We take into account any errors that might occur in an entanglement swap, which include gate errors, measurement errors and initialization errors by modeling them all as depolarizing channels, with parameters $\{p_i\}$ and multiplying them to obtain a single parameter, s_{q}, as shown in equation (C2).

$$s_{\mathrm{q}} = \prod_i (1 - p_i) \tag{C2}$$

$1 - s_{\mathrm{q}}$ is then used to parameterize a depolarizing channel that is applied after an ideal Bell state measurement. The action of this channel Φ on a given state ρ as a function of s_{q} is given by

$$\Phi(\rho, s_{\mathrm{q}}) = \left(\frac{1 + 3s_{\mathrm{q}}}{4}\right)\rho + \frac{1 - s_{\mathrm{q}}}{4}(X\rho X + Y\rho Y + Z\rho Z). \tag{C3}$$

This implies that s_{q} is a measure of the quality of an entanglement swap, and it is thus named swap quality.

The two remaining parameters in the abstract model are T_1 and T_2. An NV center's qubits can be either electrons, used as communication qubits, or carbons, used as memory qubits, each of them having different coherence times. This subtlety is lost when going to the abstract model, in which all qubits are created equal. We expect that decoherence will be more relevant in the memory qubits than in the communication qubits, so we ignore it for the latter. Besides this, one of the major sources of noise in NV centers is induced dephasing, the dephasing applied to the memory qubits whenever the communication qubit attempts to generate entanglement [44]. This noise source can also be accurately modeled by a T_1, T_2 noise model. In such a model, one applies dephasing noise with probability given by

$$p = \frac{1 - \mathrm{e}^{-t(1/T_2 - 1/2T_1)}}{2}, \tag{C4}$$

with t being the relevant time period. This is formalized by means of a dephasing channel Φ_{d} whose action on a given state ρ is given by

$$\Phi_{\mathrm{d}}(\rho, p) = (1 - p)\rho + pZ\rho Z. \tag{C5}$$

On the other hand, the noise introduced in an NV center's carbon atoms over n entanglement generation attempts can be modeled by a dephasing noise process of probability

$$p_n = \frac{1 + \left(2(1 - p_1) - 1\right)^n}{2}, \tag{C6}$$

with p_1 being the probability of a single attempt inducing dephasing noise, which can be experimentally determined [44]. If we assume that a node is always trying to generate entanglement through its electron, we can write n as a function of time:

$$n = \frac{t}{T_{\mathrm{cycle}}}, \tag{C7}$$

with T_{cycle} being the time it takes the NV to go through one entanglement generation attempt.

Matching the probability in equation (C4) to the one in equation (C6) and solving for T_2, we find:

$$T_2 = \frac{1}{1/2T_1 - \log(1 - 2p_1)/T_{\mathrm{cycle}}}. \tag{C8}$$

This allows us to account for the effect of induced dephasing in our simulations by modeling it as a T_2 noise process. We note that, in order to more closely capture induced dephasing, this noise should only be simulated when nodes are attempting entanglement generation.

In summary, we have two important sources of noise that can be modeled by T_1, T_2 processes: induced dephasing and memory decoherence. Since we want to restrict our model to 5 parameters, we must restrict ourselves to account for one of the two. In order to make an informed decision regarding which noise source to model, we run repeater chain simulations using the abstract model and the NV model introduced in [43]. For simplicity, we ignore distillation and consider an SWAP-ASAP protocol where the nodes can only attempt entanglement generation, wait or perform an entanglement swap. In order to obtain a better agreement between the entanglement generation rates of both models, we impose that nodes in the abstract model simulation can only generate entanglement with one neighbor at a time, as is the case for NV centers.

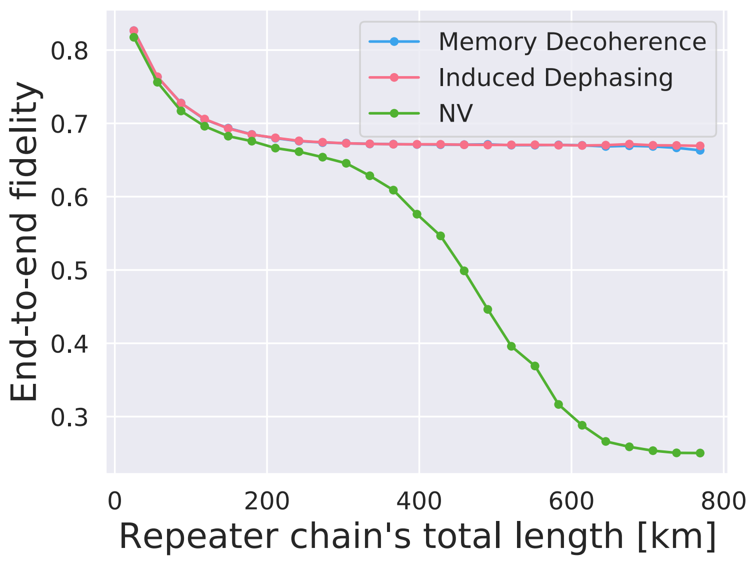


Figure 13. Variation of the end-to-end fidelity of states generated by a chain of five equally spaced nodes as the chain's length is varied in the NV model (green, triangles) and in both abstract model mappings: memory decoherence (blue, circles) and induced dephasing (red, inverted triangles). The curves overlap for short chains, but as the internode distance grows the fidelity of the NV chain falls faster. The results of the two mappings are virtually identical. The error bars are smaller than the markers.

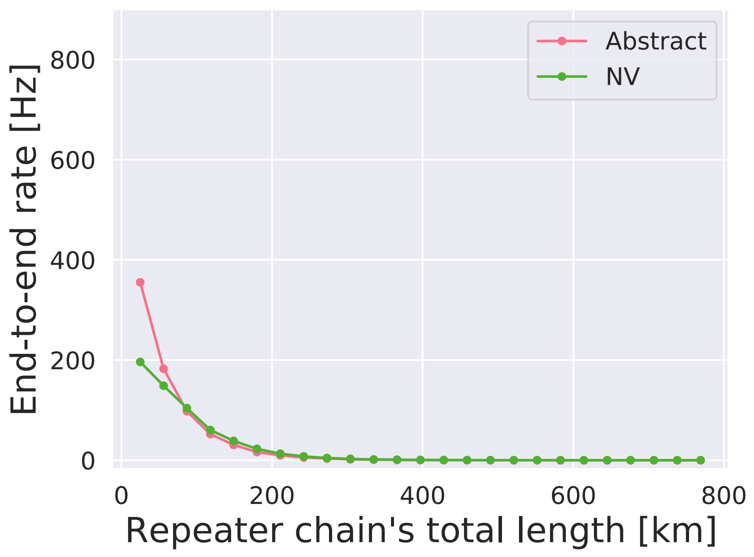


Figure 14. Variation of the end-to-end entanglement generation rate in a chain of five equally spaced nodes as the chain's length is varied in the NV model (triangles) and in the abstract model (circles). At short lengths, the rates achieved are higher in the abstract model by a factor of almost 2. As the distance increases, the two curves overlap.

C2. Comparison of NV and abstract models

We will look into how the internode distance affects the metrics we are interested in, namely end-to-end fidelity and entanglement generation rate, in the two models. To do so, we will focus on chains of equally spaced nodes, for which varying the internode distance is equivalent to varying the total length of the repeater chain.

In figure 13, we plot the end-to-end fidelity of the states generated by a chain of five equally spaced nodes as the chain's length is varied in the NV model and in both abstract model mappings. The three fidelity curves are very similar for shorter chains, roughly overlapping in chains of up to 200 km. At this point the curve for the NV model starts to diverge, dropping abruptly.

Overall, the difference between the two mappings is small. They both show very good agreement at short chain lengths, and they both perform poorly as the distances grows. This indicates that for longer distances or, alternatively, for lower fidelities, ignoring either of the noise sources results in poor agreement

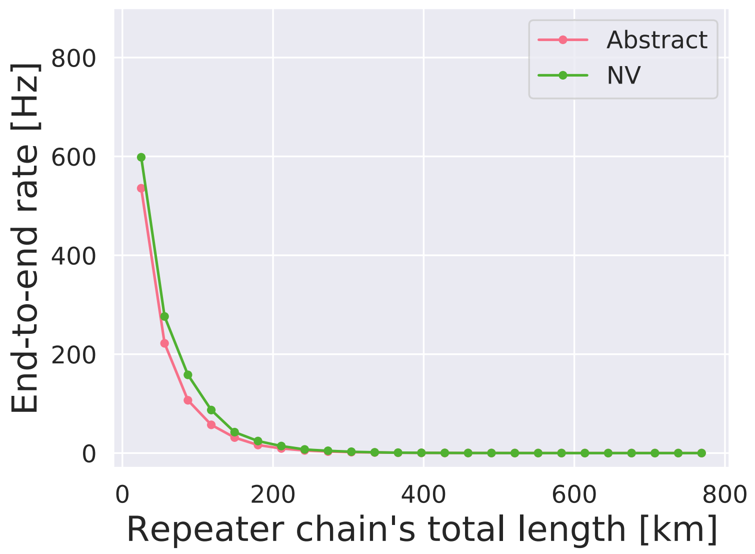


Figure 15. Variation of the end-to-end entanglement generation rate in a chain of five equally spaced nodes as the chain's length is varied in the NV model (triangles) and in the abstract model (circles) with instantaneous SWAP gates. The two curves are very close for all simulated chain lengths. The error bars are smaller than the markers.

with the NV model. In this work we will focus on scenarios where the obtained fidelities are high, above 0.7, so that the agreement is good. We will consider the induced dephasing mapping.

We turn our attentions now to the other metric of interest, the end-to-end entanglement generation rate. In figure 14, we plot the end-to-end rate against the total chain length for the same setup in both models. The behavior of the two curves is similar, although the rates in the abstract model are significantly higher. We believe that this is due to the fact that, since NV centers have only one communication qubit, they must swap established entanglement from it to a memory qubit as soon as it is generated. This does not happen in the abstract model, and thus there is no time spent on swapping the entangled states around qubits, allowing for a higher entanglement generation rate. The difference between the two curves becomes smaller as the distance increases, which could be explained by the fact that at long distances, the majority of the time is spent on generating elementary links, as success probabilities become low. The duration of local node operations become negligible in comparison, and the time taken by internal swaps is not as important in this regime.

In order to verify this, we reran the NV simulation with the internal swap being performed instantly. The results are shown in figure 15. The curves overlap over all distances the simulation covered, corroborating our hypothesis.

We conclude that the entanglement generation rates attained by the two models are similar across the board, with the the biggest difference, which happens at short internode distances, being a factor of roughly 1.8. At longer distances, the rates are the same up to statistical fluctuations.

Appendix D. Werner chains

In this appendix we give details about our approach for validating our GA-based optimization approach by applying it to a repeater chain generating Werner states.

The crux of this validation procedure is that we are able to find the optimum value of the cost function by a method other than the GA-based one we proposed. In order to do so, we require closed-form expressions for end-to-end fidelity and entanglement generation rate as functions of the input parameters, elementary link fidelity, success probability and swap quality.

Consider first, for simplicity, a three-node chain. The nodes establish elementary links whose states are of the form

$$\rho(x) = x\left|\psi^+\right\rangle\left\langle\psi^+\right| + (1-x)\frac{\mathbb{I}}{4}, \tag{D1}$$

where $|\psi^+\rangle = 1/\sqrt{2}(|01\rangle + |10\rangle)$ is the ideal Bell state and $\mathbb{I}$ is the identity. x is the Werner parameter and is related to the fidelity f of the Werner state with the ideal Bell state by $f = (1+3x)/4$. Performing an ideal BSM on two of these states, both of parameter x, results in a Werner state of parameter x^2, i.e. the post-BSM

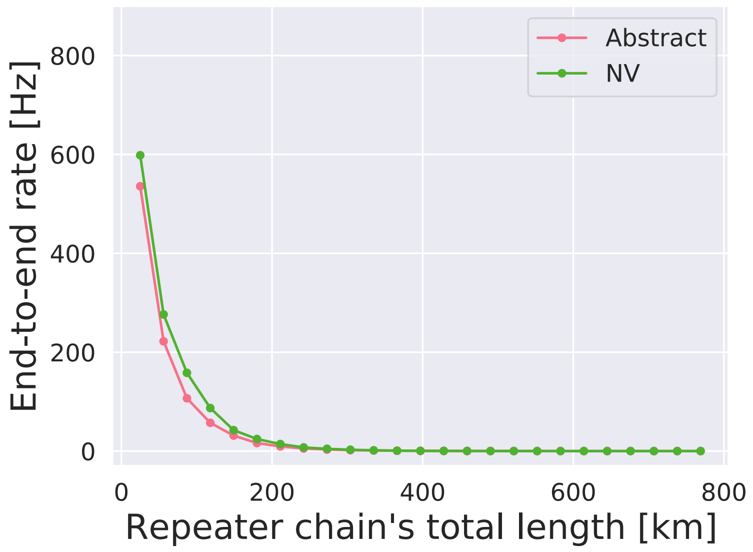


Figure 16. End-to-end fidelity of states generated by a chain of ten nodes as the swap quality is varied, for an elementary link fidelity of 0.99. The analytical and simulation curves perfectly overlap.

state ρ_{BSM} is given by

$$\rho_{\mathrm{BSM}} = x^2 \left|\psi^+\right\rangle \left\langle\psi^+\right| + (1 - x^2)\frac{\mathbb{I}}{4}. \tag{D2}$$

To simulate a noisy BSM, we then apply noise via two single-qubit depolarizing channels, one on each of the two qubits involved in the BSM. Both of these channels are parametrized by the swap quality s_q, as defined in equation (C3). The resulting Werner state has fidelity F with the ideal Bell state:

$$F(f, s_q) = \frac{1}{4} + s_q\left(\frac{1}{2} + \frac{s_q}{4}\right)\left(\frac{4f-1}{3}\right)^2. \tag{D3}$$

Iterating this process, one arrives at the following expression for the end-to-end fidelity

$$F(N, f, s_q) = \frac{1}{4} + s_q^N\left(\frac{1}{2} + \frac{s_q^N}{4}\right)\left(\frac{4f-1}{3}\right)^{N+1}, \tag{D4}$$

where N is the number of repeater nodes in the chain. As a sanity check, we ran simulations of a 10-node chain for a fixed f while varying s_q and compared the obtained end-to-end fidelity with the values obtained with equation (D4). These results are shown in figure 16.

An attentive reader might notice that equation (D4) slightly differs from the well-known result first derived in [45] in how it accounts for the effect of imperfect operations in the end-to-end fidelity. This is due to the fact that we have here parametrized the depolarizing noise in a slightly different manner, through two single-qubit channels.

We shift now our focus to the computation of the end-to-end entanglement generation rate across a three-node repeater chain. We note that this quantity is simply the inverse of the waiting time, which we denote by T. Let us start with the generation of elementary links. Since we model elementary link generation attempts as processes succeeding with a fixed probability p_{suc}, T_0 is a discrete random variable following a geometric distribution. Its expected value is then given by:

$$\mathbb{E}(T_0) = \frac{1}{p_{\mathrm{suc}}}T_{\mathrm{cycle}}, \tag{D5}$$

where $\mathbb{E}$ denotes the expected value and T_{cycle} is the cycle time, i.e. the time a single entanglement generation attempt takes. We consider a sequential repeater chain, i.e. one in which nodes can only attempt entanglement generation with one of their neighbors at a time. Therefore, the end-to-end waiting time is given by:

$$\mathbb{E}(T) = 2\mathbb{E}(T_0) + T_{\mathrm{SWAP}}, \tag{D6}$$

where T_{SWAP} is the time an entanglement swap takes. This holds because the repeater node has to generate elementary links with both its neighbors, and it can only start generating the second once it has finished

Table E1. Values considered for NV model parameters. See e.g. [10, 40, 43] for detailed explanations of parameters.

Parameter	Value
Visibility	0.90
σ Phase drift	0.35 rad
$p_{\text{doubleexcitation}}$	0.06
$p_{\text{electron measureerror}}$	0.025
$p_{\text{electron1qubiterror}}$	0
$F_{\text{carbonZrot}}$	0.999
F_{EC}	0.97
T_{1carbon}	10 h
p_{det}	0.000 13
$p_{\text{darkcount}}$	2.5×10^{-6}
$\text{N}_{1/e}$	1400
$p_{\text{losslength}}$	0.5 dB km^{-1}

generating the first. Furthermore, after having generated these links, it must swap them. We then define the entanglement rate R as the inverse of the expected waiting time:

$$R = \frac{1}{\mathbb{E}\,(T)}. \tag{D7}$$

With equations (D4) and (D7) in hand, we can compute the end-to-end fidelity and entanglement generation rate using only the input parameters f, s_q and p_{suc} and the simulation parameters T_{cycle} and T_{SWAP}. This implies that we can also directly compute the cost function, as we have analytical expressions for every term appearing in the cost function defined equation (6). We then used the Basin-hopping algorithm [46] to find the global minimum of this cost function for a target fidelity $f_{\text{min}} = 0.6$ and a target entanglement generation rate $r_{\text{min}} = 1$ Hz over a chain of equally spaced nodes. We took as baseline values $f_b = s_{q_b} = 0.5$ and $p_{\text{suc}_b} = 10^{-10}$. The Basin-hopping algorithm is available in the SciPy library.

Appendix E. Computing baseline values in the abstract model

E1. Uniform spacing

In order to use a realistic and up to date set of baseline values, we considered the latest results achieved in Ronald Hanson's Lab at QuTech, in Delft [47]. The values for T_1 and T_2 can be directly computed from experimental values. The same is true for s_q, which can be derived from entanglement swap experiments. This does not hold for the elementary link-related parameters, namely the fidelity F_{EL} and success probability p_{suc}. Their values are heavily distance-dependent, and to date entanglement generation experiments using NV centers have only been realized at distances on the single kilometer scale [6]. We therefore use instead the model proposed in [43] with the experimental values we obtained from the Hanson group as inputs to compute the baseline values for F_{EL} and p_{suc} for the elementary link lengths we consider. In table E1 we list the values used as inputs to the NV model to compute the baseline abstract parameter values. Explaining the physical meaning of each of these parameters would require a detailed exposition of the NV model, which is beyond the scope of this work. This can instead be found in [40, 43]. We note that although these parameter values have all been measured in actual laboratory experiments, they are not absolute truths. Different setups might achieve slightly different performances, and even in the same NV center not all nuclear spins are identical nor do they couple in exactly the same way to the electron spin. These nonetheless provide a valuable picture of the current state of the art.

The bright state population α is also a required parameter in the model. We chose not to include it in table E1 as this parameter is not defined by the quality of the hardware but can instead be chosen. It represents the fraction of the NV electron spin that is in the bright state, i.e. the state that emits photons. It therefore has a direct effect on the success probability of establishing elementary links, as a bigger α results in a higher photon emission probability. On the other hand, increasing α also increases the fraction of terms orthogonal to the Bell basis in the entangled state, decreasing the elementary link fidelity. There is thus a trade-off between elementary link fidelity and success probability when varying an NV center's bright state population [43]. However, in our simplified abstract model we ignore any correlations between parameters, so such a trade-off is not present. We therefore chose to ignore the existence of the trade-off in NV centers when computing the baseline value. Our process for computing these values consisted of performing a parameter scan over α with the NV model and choosing the highest achievable elementary link fidelity and success probability. In practice, this means that the baseline values considered for the

Table E2. Baseline values of the abstract model parameters for the different elementary link lengths considered.

	73 km	89 km	100 km	133 km	200 km
T_1			10 h		
T_2			4.9 ms		
s_q			0.8459		
F_{EL}	0.95	0.94	0.90	0.80	0.52
p_{suc}	1.3×10^{-4}	7.0×10^{-5}	1.5×10^{-5}	2.2×10^{-6}	9.6×10^{-8}

Table E3. Baseline values for the links (DH stands for Delft–The Hague, HL for The Hague–Leiden and LA for Leiden–Amsterdam) and at negligible fiber length (NL).

	p_{suc}	F_{EL}
DH	0.002 588	0.9683
HL	0.000 9187	0.9643
LA	0.000 9082	0.9642
NL	0.004 600	0.9698

elementary link fidelity were obtained with very low values of α and, conversely, the baseline values of the elementary link success probability were obtained with the highest values of α. We note that we restricted the parameter scan to the $[0, 0.5]$ interval, because for $\alpha > 0.5$ entanglement is impossible even for perfect parameters.

Taking all of this into account, we show in table E2 the baseline values we obtained for the abstract model parameters. The distances in the table correspond to the elementary link lengths we considered in the two uniform spacing use cases.

E2. Real network

The way we arrive at the baseline values used in this use case is identical to what was described in the previous section, with the exception of F_{EL} and p_{suc}. We will now explain why these values must be computed in a different manner, as well as the process we employed to do so.

In order to arrive at realistic baseline values for the network we introduced in figure 7 we used real-life fiber data that was made available to us by SURF. Although we cannot share this data, we used both the physical length of the fibers connecting the locations indicated in the figure 7 and their measured attenuation values. These two quantities then have an impact on the baseline values we consider for F_{EL} and p_{suc}, resulting in three different sets of baseline values, one for each of the links in the network. This raises some questions about how the value of the cost function introduced in equation (5) should be computed, as this function takes as input only one set of baseline values and a respective set of improved values. There are multiple ways to address this. We will now explain the approach we took.

We start by computing four sets of baseline values: one for each of the links in the network plus one at negligible fiber length (NL). By this we mean that the length we use as an input to the model in [43] is such that the impact of losses in the fiber are negligible. The cost associated with a given set of parameters is computed with respect to the set of baseline values at NL. One can then think of this set of parameters as the improved parameters at NL. In order to obtain the sets of parameters that will be used in our simulation we start by obtaining the improvement factor, defined in equation (4), for each of the parameters. These improvement factors are then applied to the baseline values of each of the links according to equation (1). The resulting three sets of values, one for each of the links, are finally the ones fed into our simulation. We reiterate that this process only applies to F_{EL} and p_{suc}. The baseline values of the remaining parameters, not being dependent on fiber length, are computed in the same way as described in the previous section. In table E3 we present the baseline values we arrived at through the aforementioned process.

Appendix F. Search space reduction using previous runs

We can use previous optimization runs to limit the search space of new runs and hence increase the probability of a good solution being found. As an example of how this can be done, suppose we have performed an optimization run over a repeater chain of 5 uniformly spaced nodes spanning some distance L. This resulted in a solution that achieves an end-to-end entanglement generation rate of $R = 1$ Hz with an elementary link success probability of p_{suc_5}, the subscript being here used to denote the number of nodes in the chain. Say we now want to apply our optimization method to a chain of 7 uniformly spaced nodes spanning the same distance L. As more elementary links need to be established and more entanglement swaps need to be performed, we know with certainty that, in order to achieve the same R a higher elementary link success probability will be needed, i.e. $p_{\text{suc}_7} > p_{\text{suc}_5}$. We can thus impose a lower bound of p_{suc_5} on the search space, reducing it.

These considerations are easy to make for the case of the elementary link success probability. Since we hold the operation times constant and implement no cut-off, it is the only parameter influencing the end-to-end entanglement generation rate. The same is not true for the other metric of interest, the end-to-end fidelity. As a concrete example, assume that the best solution found for a repeater chain of 5 uniformly spaced nodes had an elementary link fidelity $F_{\text{EL}} = 0.96$ and a swap quality $s_q = 0.98$, resulting in an end-to-end fidelity of 0.75. One could be inclined to, in a future optimization run, upper bound the search space of F_{EL} by 0.96 to help lead the algorithm to a solution with an end-to-end fidelity closer to the target value of 0.7. However, it might be that there is a solution with $F_{\text{EL}} > 0.96$ and $s_q < 0.98$ that results in a lower cost function value than any solution with $F_{\text{EL}} < 0.96$. Therefore, by imposing this upper bound we could be preventing the algorithm from ever finding the ideal solution.

ORCID iDs

Francisco Ferreira da Silva https://orcid.org/0000-0003-3642-4350
Ariana Torres-Knoop https://orcid.org/0000-0001-8976-2965
Tim Coopmans https://orcid.org/0000-0002-9780-0949
Stephanie Wehner https://orcid.org/0000-0002-8433-0730

References

[1] Bennett C H and Brassard G 2020 Quantum cryptography: public key distribution and coin tossing (arXiv:2003.06557)
[2] Ekert A K 1991 Quantum cryptography based on Bell's theorem *Phys. Rev. Lett.* **67** 661
[3] Kómár P, Kessler E M, Bishof M, Jiang L, Sørensen A S, Ye J and Lukin M D 2014 A quantum network of clocks *Nat. Phys.* **10** 582–7
[4] Buhrman H and Röhrig H 2003 Distributed quantum computing *Int. Symp. on Mathematical Foundations of Computer Science* (Springer) pp 1–20
[5] Wehner S, Elkouss D and Hanson R 2018 Quantum internet: a vision for the road ahead *Science* **362** eaam9288
[6] Hensen B *et al* 2015 Loophole-free bell inequality violation using electron spins separated by 1.3 kilometres *Nature* **526** 682–6
[7] Briegel H-J, Dür W, Cirac J I and Zoller P 1998 Quantum repeaters: the role of imperfect local operations in quantum communication *Phys. Rev. Lett.* **81** 5932
[8] Collins O, Jenkins S, Kuzmich A and Kennedy T 2007 Multiplexed memory-insensitive quantum repeaters *Phys. Rev. Lett.* **98** 060502
[9] Munro W J, Azuma K, Tamaki K and Nemoto K 2015 Inside quantum repeaters *IEEE J. Sel. Top. Quantum Electron.* **21** 78–90
[10] Rozpędek F, Yehia R, Goodenough K, Ruf M, Humphreys P C, Hanson R, Wehner S and Elkouss D 2019 Near-term quantum-repeater experiments with nitrogen-vacancy centers: overcoming the limitations of direct transmission *Phys. Rev.* A **99** 052330
[11] Humphreys P C, Kalb N, Morits J P J, Schouten R N, Vermeulen R F L, Twitchen D J, Markham M and Hanson R 2018 Deterministic delivery of remote entanglement on a quantum network *Nature* **558** 268–73
[12] Yu Y *et al* 2020 Entanglement of two quantum memories via fibres over dozens of kilometres *Nature* **578** 240–5
[13] Zwerger M, Lanyon B P, Northup T E, Muschik C A, Dür W and Sangouard N 2017 Quantum repeaters based on trapped ions with decoherence-free subspace encoding *Quantum Sci. Technol.* **2** 044001
[14] Sangouard N, Simon C, De Riedmatten H and Gisin N 2011 Quantum repeaters based on atomic ensembles and linear optics *Rev. Mod. Phys.* **83** 33
[15] Coopmans T *et al* 2020 NetSquid, a discrete-event simulation platform for quantum networks (arXiv:2010.12535)
[16] Van Meter R, Ladd T D, Munro W J and Nemoto K 2008 System design for a long-line quantum repeater *IEEE/ACM Trans. Netw.* **17** 1002–13
[17] Wu X, Kolar A, Chung J, Jin D, Zhong T, Kettimuthu R and Suchara M 2020 Sequence: a customizable discrete-event simulator of quantum networks (arXiv:2009.12000)
[18] Wallnöfer J, Melnikov A A, Dür W and Briegel H J 2020 Machine learning for long-distance quantum communication *PRX Quantum* **1** 010301
[19] Muralidharan S, Li L, Kim J, Lütkenhaus N, Lukin M D and Jiang L 2016 Optimal architectures for long distance quantum communication *Sci. Rep.* **6** 20463
[20] Jiang L, Taylor J M, Khaneja N and Lukin M D 2007 Optimal approach to quantum communication using dynamic programming *Proc. Natl Acad. Sci.* **104** 17291–6
[21] Santra S, Jiang L and Malinovsky V S 2019 Quantum repeater architecture with hierarchically optimized memory buffer times *Quantum Sci. Technol.* **4** 025010
[22] Goodenough K, Elkouss D and Wehner S 2020 Optimising repeater schemes for the quantum internet (arXiv:2006.12221)
[23] Krastanov S, Albert V V and Jiang L 2019 Optimized entanglement purification *Quantum* **3** 123
[24] Torres-Knoop A, Coopmans T, Maier D and Silva F 2020 Smart-stopos https://gitlab.com/aritoka/smart-stopos
[25] Werner R F 1989 Quantum states with Einstein–Podolsky–Rosen correlations admitting a hidden-variable model *Phys. Rev.* A **40** 4277
[26] Gunantara N 2018 A review of multi-objective optimization: methods and its applications *Cogent Eng.* **5** 1502242

[27] Schaffer J D 1986 Some experiments in machine learning using vector evaluated genetic algorithms (artificial intelligence, optimization, adaptation, pattern recognition) *PhD Thesis* Vanderbilt University

[28] Vikhar P A 2016 Evolutionary algorithms: a critical review and its future prospects *2016 Int. Conf. on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)* (IEEE) pp 261–5

[29] Holland J H 1992 *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence* (Cambridge, MA: MIT Press)

[30] Beyer H-G and Schwefel H-P 2002 Evolution strategies–a comprehensive introduction *Nat. Comput.* **1** 3–52

[31] Storn R and Price K 1997 Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces *J. Global Optim.* **11** 341–59

[32] Kennedy J and Eberhart R 1995 Particle swarm optimization *Proc. of ICNN'95-Int. Conference on Neural Networks* vol 4 (IEEE) pp 1942–8

[33] Shi Y and Eberhart R 1998 A modified particle swarm optimizer *1998 IEEE Int. Conf. on Evolutionary Computation Proc. IEEE World Congress on Computational Intelligence (Cat. No. 98TH8360)* (IEEE) pp 69–73

[34] Goldberg D E 2006 *Genetic Algorithms* (Chennai: Pearson Education India)

[35] SURF 2019 Stopos https://gitlab.com/surfsara/stopos

[36] Digalakis J G and Margaritis K G 2001 On benchmarking functions for genetic algorithms *Int. J. Comput. Math.* **77** 481–506

[37] Goldberg D E 1989 *Genetic Algorithms in Search, Optimization and Machine Learning* 1st edn (Reading, MA: Addison-Wesley)

[38] Bradley C, Randall J, Abobeih M, Berrevoets R, Degen M, Bakker M, Markham M, Twitchen D and Taminiau T 2019 A ten-qubit solid-state spin register with quantum memory up to one minute *Phys. Rev.* X **9** 031045

[39] Rabbie J, Chakraborty K, Avis G and Wehner S 2020 Designing quantum networks using preexisting infrastructure (arXiv:2005.14715)

[40] Dahlberg A *et al* 2019 A link layer protocol for quantum networks *Proc. of the ACM Special Interest Group on Data Communication* pp 159–73

[41] Srinivas M and Patnaik L M 1994 Adaptive probabilities of crossover and mutation in genetic algorithms *IEEE Trans. Syst. Man Cybern.* **24** 656–67

[42] Luke S 2013 *Essentials of Metaheuristics* 2nd edn (Morrisville, NC: Lulu) http://cs.gmu.edu/\ignorespaces&tnqx223c;sean/book/metaheuristics/

[43] Kalb N *et al* 2017 Entanglement distillation between solid-state quantum network nodes *Science* **356** 928–32

[44] Kalb N, Humphreys P C, Slim J and Hanson R 2018 Dephasing mechanisms of diamond-based nuclear-spin memories for quantum networks *Phys. Rev.* A **97** 062330

[45] Dür W, Briegel H-J, Cirac J I and Zoller P 1999 Quantum repeaters based on entanglement purification *Phys. Rev.* A **59** 169

[46] Wales D J and Doye J P K 1997 Global optimization by basin-hopping and the lowest energy structures of Lennard–Jones clusters containing up to 110 atoms *J. Phys. Chem.* A **101** 5111–6

[47] Hermans S 2020 private communication