Identifying network topologies via quantum walk distributions

Claudia Benedetti[1] and Ilaria Gianani[2]

[1]*Dipartimento di Fisica "Aldo Pontremoli", Università degli Studi di Milano, 20133, Milan, Italy*
[2]*Dipartimento di Scienze, Università degli Studi Roma Tre,*
Via della Vasca Navale 84, 00146, Rome, Italy[*]

Control and characterization of networks is a paramount step for the development of many quantum technologies. Even for moderate-sized networks, this amounts to explore an extremely vast parameters space in search for the couplings defining the network topology. Here we explore the use of a genetic algorithm to retrieve the topology of a network from the measured probability distribution obtained from the evolution of a continuous-time quantum walk on the network. Our result shows that the algorithm is capable of efficiently retrieving the required information even in the presence of noise.

Networks are a fundamental model to understand the underlying properties of complex systems. They are invaluable tools to describe phenomena happening at different scales ranging from social interactions [1, 2], to biological processes [3–7], from the configurations of molecules [8, 9], to the structure of internet [10–13] and physical systems alike [14–18]. In the context of quantum technologies, networks constitute the prime structure of communication and computation protocols [19–23]. Understanding how quantum information can be reliably transmitted between distant nodes of a network, or routed among different computational units, is a key step and requires a full characterization of the network's structure. While a direct control may not be attainable with the required accuracy and precision, a straightforward strategy to provide such characterization is that of probing the network with a walker that gathers information on its topology by undergoing an evolution which depends on the network's structure. This is the case of continuous-time quantum walks (CTQWs) [24–33], which thus emerge as a natural paradigm for tackling this task.

Two different scenarios may present: the topology of the network may be known, but an accurate estimation of the coupling strengths between each node may be required. This is tantamount to estimating multiple parameters, and can be address in quantum metrological terms [34–37]. It might otherwise be the case that the topology of the network is not known in advance. Whether one is interested in characterizing a physical network or a simulated one, this will be relying on an experimental platform controlled with set of experimental parameters $\mathbf{\Lambda^{exp}}$. These need to be mapped to the associated set of parameters describing the CTQW happening on the network, $\mathbf{\Lambda^{QW}}$, i.e. the Hamiltonian parameters of the quantum walk which, assuming all coupling strengths are fixed to unity and on-site energies to zero, coincide with the adjacency matrix identifying the topology of the network. In order to asses the evolution of the probe, one has to address an observable, such as the spatial probability distribution on the network, which will strongly depend on the network's topology. However, since an analytical description of this distribution for CTQWs is often unattainable, and furthermore the relation between the QW Hamiltonian and its probability distribution is highly non-linear, performing a direct inversion can be involved. At the same time, the parameter space in this instance becomes exceedingly large for this to be treated as an estimation problem. An alternative solution is to cast the issue in terms of a search problem. Having access solely to the initial state of the probe and to the measured experimental distributions at fixed times, the task becomes that of finding an adjacency matrix that matches the evolution. Here we tackle this matter by using a genetic algorithm. We use the algorithm to successfully retrieve different topologies in the ideal case as well as when the measured probabilities are affected by noise.

We consider a CTQW with zero on-site energies, defined by the couplings $\mathbf{\Lambda^{QW}} = \{J_{xy}\}$ between two nodes of the network x and y, such that its Hamiltoian is:

$$H(\mathbf{\Lambda^{QW}}) = \sum_{xy} J_{xy} |x\rangle\langle y| . \tag{1}$$

We assume that the couplings J_{xy} can take only two values: $J_{xy} = 0$ if the link between two nodes is off, or $J_{xy} = 1$ if the link is on, so that each edge is bound to have the same strength. The Hamiltonian thus coincides with the adjacency matrix of the network, hence, determining its parameters amounts to determining the network's topology. The evolution of a walker in the initial state $|\psi_0\rangle$ is described by the unitary operator e^{-iHt}. The probability of occupying a site x at a time t is then:

$$p_x(t, \mathbf{\Lambda^{QW}}) = |\langle x| e^{-iH(\mathbf{\Lambda^{QW}})t} |\psi_0\rangle|^2. \tag{2}$$

Given an undirected graph of n sites, our objective is that of retrieving the couplings $\mathbf{\Lambda^{QW}} = \{J_{12}, \ldots, J_{(n-1)n}\}$, i.e. a binary string of length $n_c = n(n-1)/2$, having access only to the initial state of the network and to the probabilities $p_x(t_k, \mathbf{\Lambda^{QW}})$ measured at times t_k. We tackle this challenge by means of a genetic algorithm (GA). GAs are versatile iterative search algorithms inspired by natural selection and have been extensively
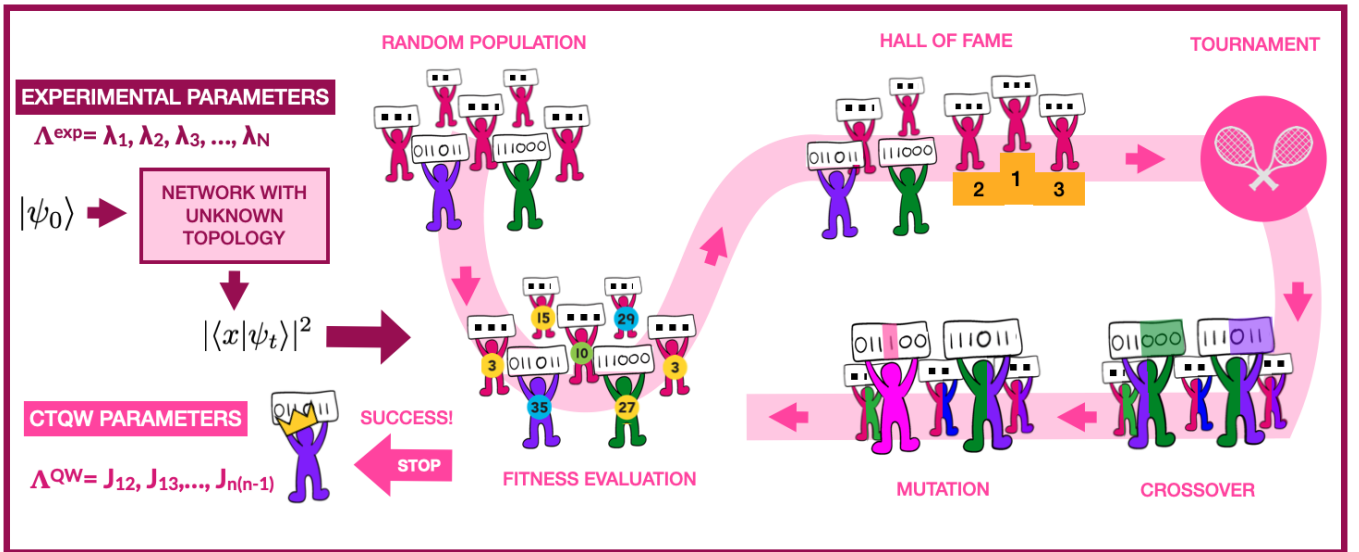
FIG. 1. **Conceptual scheme.** Given an initial probe state $|\psi_0\rangle$ and a network with unknown topology controlled by a set of experimental parameters, we aim at retrieving the topology of the network measuring the probability distributions of the the probe evolved with a CTQW. This is achievede through a genetic algorithm in which the probability distributions are employed to evaluate the fitness score, as described in the main text.

employed for quantum tasks [38–41]. They rely on the evolution of a population of individuals, each defined by a chromosome string and a fitness score, which breed new individuals replacing the previous population at each iteration. By promoting the reproduction of the fittest individuals while introducing various mechanisms to ensure enough genetic variability, GAs allow to efficiently retrieve the optimal solution[42, 43].

We encode the chromosomes as binary strings Λ_i of length n_c, so that each gene constituting the chromosome is a coupling J_{xy}. The fitness of each individual is evaluated as follows: Λ_i is used to evolve the initial state of the probe up to selected times t_k obtaining the probability distributions $p_x(t_k, \Lambda_i)$. For practical purposes, we concatenate the probabilities at different times in a single array that we call $\pi_x(\{t_k\}, \Lambda_i)$. Using multiple times allows to remove eventual ambiguities and to mitigate the effects of local minima, thus improving the performance of the algorithm. We then check the distance between these probabilities and the measured ones $\pi_x(\{t_k\}, \Lambda^{\mathbf{QW}})$ e.g. by using the Kullback-Leibler divergence. When the distance is null, $\Lambda_i = \Lambda^{\mathbf{QW}}$. The value of the distance will be the fitness score of each individual. Thus, in our case, the more fit an individual, the smaller its fitness score. The correct couplings will be those having a fitness score equal to 0.

The algorithm scheme is shown in Fig. 1 and operates as follows: An initial random population of size n_p is generated, and its fitness is evaluated as described above. An *elitist* function selects a small percentage p_e of individuals with the best fitness scores to constitute the *hall of fame*, which will be cloned in the next generation. The whole population is then entered in a tournament where k in-

dividuals at the time compete to be selected for breeding the next generation. This is achieved through a crossover strategy in which the chromosomes of the selected parents are mixed with a probability p_c. The size of the population is kept constant through each generation, so that each selected pair of individuals will produce two children. In order to ensure genetic diversity, with a small probability p_m, children can undergo mutations, consisting in bit flips. The new born children together with the cloned hall of fame individuals become the next generation, and the algorithm continues iteratively, stopping either when the optimal individual is found (i.e. fitness score equals to zero), or when a maximum number of generations n_g is reached. A full depiction of the algorithm and of the implementation of the genetic operations are reported in the Supplementary Information.

The initial state of the probe as well as the evolution times at which the probabilities are measured play a fundamental role towards the success of the algorithm: for instance, choosing a localized state may result in discarding part of the network, if composed by two or more disjoints subnetworks; evolving the state for too short a time in a large network, may preclude the state to reach the whole network. While we do not perform a full optimization of the initial state, we choose one that allows to explore a large variety of different topologies and network sizes. Also all the hyperparameters defining the algorithm (population size n_p, elitist population p_e, individuals involved in each tournament k, crossover probability p_c, mutation probability p_m, max number of generations n_g) can be optimized in accordance with the task at hand and specifically with the network size. In our analysis we vary the network size to explore how the
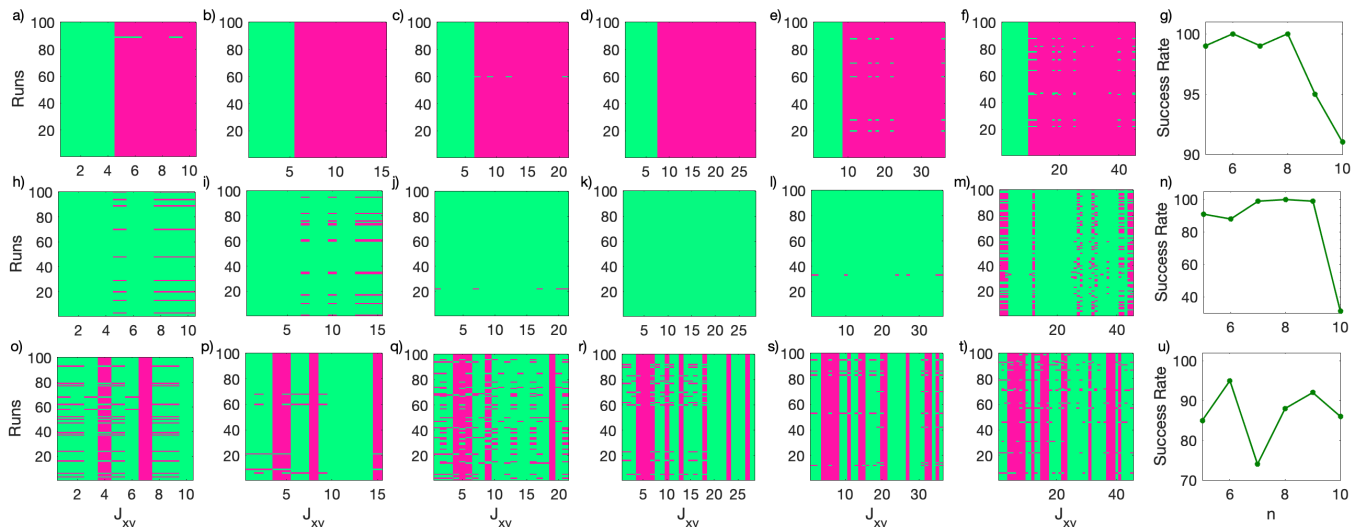
FIG. 2. **Results without noise** Retrieved couplings for N runs of the algorithm for star (a-f), complete (h-m), and random (o-t) networks for varying network sizes: (a,h,o) n=5, (b,i,p) n=6, (c,j,q) n=7, (d,k,r) n=8, (e,l,s) n=9 (f,m,t) n=10. (g,n,u) success rates as a function of network size. Green indicates a coupling equal to 1, fuchsia a coupling equal to 0.

algorithm scales with an increasing number of couplings, but for the sake of simplicity we have chosen to keep all hyperparameters fixed aside from the population size n_p. Our results hence are but a lower bound to the achievable performance attainable by fine-tuning for a fixed network size.

Here we report the results obtained with a star graph, a complete graph and a graph with an arbitrary topology. This last network is a simplified version of the graph in Ref [44] describing the relations between the characters in the novel Les Misérables [45]. Results for additional topologies (line and circle) and further details on the generation of the Les Misérables graph can be found in the Supplementary Information. In order to test the algorithm, we inspect networks with nodes from n=5 to n=10, thus we search for binary strings with length $n_c = 10$ to $n_c = 45$. We measure the probability distributions at two distinct times, $t_1 = 0.5$ and $t_2 = 0.6$. As mentioned above, all hyperparameters are kept fixed (see Supplementary Information), aside from the population size n_p which we scale as $n_p = 2 \cdot n_c^2$. This ensures a trade-off between computation time and performance, and allows us to provide a controlled comparison for the performance at different sizes. We fix the maximum number of generations to $n_g = 100$, and, for each configuration, we run the algorithm N=100 times.

We first consider the ideal case in which the probability distributions are noiseless. The results are reported in Fig 2, which shows the couplings values (green = 1, fuchsia=0) obtained for each run of N for the star (a-f), complete (h-m) and Les Misérables graph (o-t), as well as the success rate in each instance (g, n, u). Fig. 2 highlights how most of the times when the algorithm fails it returns the same (wrong) couplings. This effect is due to the algorithm getting stuck in the same local minima
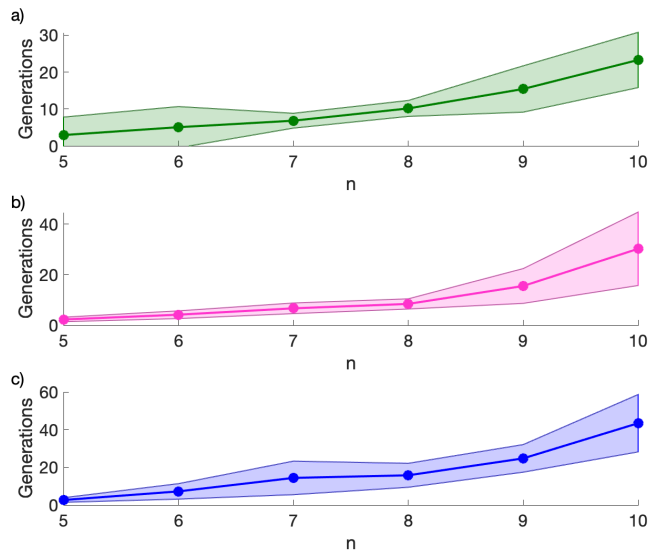

FIG. 3. **Algorithm convergence.** Average numbers of generations required for convergence over N runs of the algorithm for a star (a), complete (b), and random (c) network as described in the main text. The shaded region is the standard deviation error over the N runs.

because for the chosen evolution times there are multiple configurations leading to probabilities which are very close to the true one. The most affected network is the complete, whose success rate, for n=10, drops to 31%. However, it is sufficient to run the algorithm including also a third probability measured at time $t_3 = 1$, and a success rate of 73% is recovered (see the Supplementary Information). In Fig. 3 we report the number of generations needed for convergence as a function of n, which predictably increases with the number of network
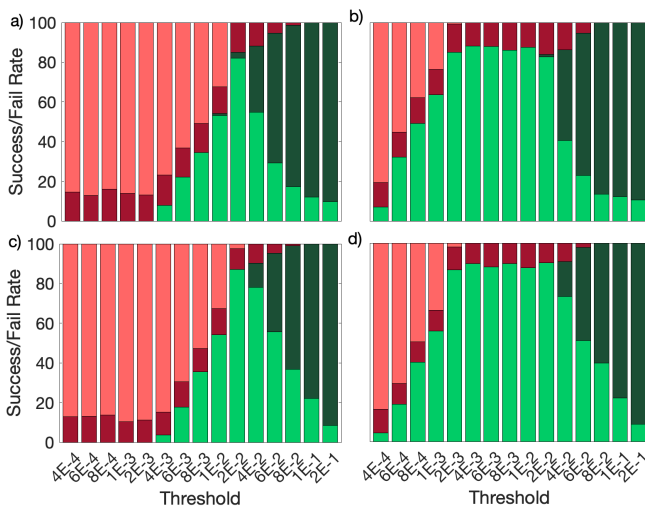
FIG. 4. **Results with noise.** Success/fail rates over the MC runs for the algorithm performed on a star network (a-b) and a complete network (c-d) for $N_r = 500$ resources (a-c) and $N_r = 5000$ resources (b-d). Light green: true positives; Dark green: false positives; Light red: false negatives; Dark red: true negatives.

sites, as does the search space. Our results show a remarkable efficiency of the search algorithm employed: in fact, the number of possible combinations Λ_i scales with 2^{n_c}, while we are inspecting, at most, $2 \cdot n_c^2 \cdot n_g$ combinations, assuming the worst case scenario in which we run the algorithm for the maximum number of generations and completely replace the population each time. For n=10, when $n_c = 45$ the combinations hence amount to $\sim 3.5 \cdot 10^{13}$, and we are exploring less than $\sim 4 \cdot 10^5$ configurations.

In a real-life scenario, the probabilities $p(t_k, \mathbf{\Lambda^{QW}})$ used to evaluate the fitness score would be affected by noise. This needs to be accounted for when evaluating the distance by setting a threshold value T below which two probabilities are considered equal. The algorithm thus needs to be modified to halt whether the distance between the measured and evaluated probabilities is smaller than T , which counts as a success, or when it reaches the maximum number of generations, in which case the algorithm has failed. Depending on the value of T, there can be four outcomes: 1) *True negative*: The algorithm fails to reach T and the couplings are not found. 2) *False negative*: The algorithm fails to reach T, but the exact string of couplings has been found. This happens if T is set too low compared to the noise affecting the probabilities. 3) *True positive*: the alogirhtm successfully finds a fitness below T, and that corresponds to the exact couplings. 4) *False positive*: the algorithm successfully finds a fitness below T, but the couplings are not correct. This happens when the threshold is set too high compared to the noise, and hence the algorithm stops be-

fore it can converge. In order to test this behaviour, we simulate the measured probabilities for a network with n=5 for a star topology and a fully connected topology, using the same hyperparameters as before aside from the max number of generations which we set to $n_g = 5$. We know from the ideal case (Fig. 3) that for these topologies the algorithm converges in more than 5 generations, so we do expect to have some true negative outcomes. We simulate the probability measurements with a total of N_r resources ranging from $N_r = 500$ to $N_r = 5000$, and through a Montecarlo (MC) routine we add Possionian noise to the simulated probabilities. For each MC run, we average the successes/fails over N=10 runs of the algorithm. We record the results for threshold values ranging from T= $4 \cdot 10^{-4}$ to T= 0.2. In Fig. 4 we report the results of the success/fail rates over 100 MC runs for the star network (a-b) and complete network (c-d) with $N_r = 500$ (a-c) and $N_r = 5000$ (b-d) as a function of the threshold value (for other N_r see Supplementary Information). As expected, we can observe the four behaviours described early: when T is too low, the outcomes are dominated by false negatives (light red), with a small percentage of true negatives, due to the fact that the algorithm would take more than 5 generations to converge. As the threshold increases so do the number of true positives, while the true negatives remain constant. For larger thresholds both the true positive and true negative drop. The algorithm always satisfies the threshold condition before it can converge to the actual solution.

In conclusion, we have employed a genetic algorithm to retrieve the topology of a network having access solely to the initial state of a probe undergoing a CTWQ and to the measured probability distributions at given times. We have explored the performance of the algorithm for different network sizes and topologies, as well as when the measured probabilities are affected by Poissonian noise. The algorithm maintains high performance levels for all the configurations explored, which could be further optimized by fine-tuning the hyperparameters for a specific network size. The genetic algorithm is particularly suited to address large parameter spaces, however increasing the network size by order of magnitudes or remove the constraint on the coupling strength would make it challenging in terms of computational times and resources. In order to achieve such scalability, a perspective is that of extending these results to incorporated machine learning techniques. By relying solely on measured probabilities, our technique provides a simple but yet effective strategy for the routine characterization of networks, and as such constitutes an enabling step towards most developing quantum technologies based on complex networks [46–51], as well as a tool for exploring new involved simulation regimes which have non-trivial mapping between the experimental control and the CTQW parameters [52–54].

[1] S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications* (Cambridge University Press, 1994).

[2] J.-P. Onnela, J. Saramäki, J. Hyvönen, G. Szabó, M. A. de Menezes, K. Kaski, A.-L. Barabási, and J. Kertész, Analysis of a large-scale weighted network of one-to-one human communication, New J. Phys. **9**, 179 (2007).

[3] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A. L. Barabási, The large-scale organization of metabolic networks, Nature **407**, 651 (2000).

[4] R. Pastor-Satorras and A. Vespignani, Epidemic spreading in scale-free networks, Phys. Rev. Lett. **86**, 3200 (2001).

[5] S. Maslov and K. Sneppen, Specificity and stability in topology of protein networks, Science **296**, 910 (2002).

[6] E. de Silva and M. Stumpf, Complex networks and simple models in biology, J. R. Soc. Interface **2**, 419–430.

[7] M. B. Plenio and S. F. Huelga, Dephasing-assisted transport: quantum networks and biomolecules, New J. Phys. **10**, 113019 (2008).

[8] W. Winterbach, P. V. Mieghem, M. Reinders, H. Wang, and D. d. Ridder, Topology of molecular interaction networks, BMC Systems Biology **7**, 90 (2013).

[9] L. De Keer, K. I. Kilic, P. H. M. Van Steenberge, L. Daelemans, D. Kodura, H. Frisch, K. De Clerck, M.-F. Reyniers, C. Barner-Kowollik, R. H. Dauskardt, and D. R. D'hooge, Computational prediction of the molecular configuration of three-dimensional network polymers, Nat. Mater. **20**, 1422 (2021).

[10] M. Faloutsos, P. Faloutsos, and C. Faloutsos, On power-law relationships of the internet topology, in *ACM SIGCOMM 99', Comput. Commun. Rev.* (Association for Computing Machinery, New York, NY, USA, 1999) p. 251–262.

[11] G. Caldarelli, R. Marchetti, and L. Pietronero, The fractal properties of internet, Europhys. Lett. **52**, 386 (2000).

[12] R. Pastor-Satorras, A. Vázquez, and A. Vespignani, Topology, hierarchy, and correlations in internet graphs, in *Complex Networks* (Springer Berlin Heidelberg, Berlin, Heidelberg) pp. 425–440.

[13] Y. He, G. Siganos, and M. Faloutsos, Internet topology, in *Encyclopedia of Complexity and Systems Science* (Springer New York, New York, NY) pp. 4930–4947.

[14] J. I. Cirac, P. Zoller, H. J. Kimble, and H. Mabuchi, Quantum state transfer and entanglement distribution among distant nodes in a quantum network, Phys. Rev. Lett. **78**, 3221 (1997).

[15] T. Chanelière, D. N. Matsukevich, S. D. Jenkins, S. Y. Lan, T. A. B. Kennedy, and A. Kuzmich, Storage and retrieval of single photons transmitted between remote quantum memories, Nature **438**, 833 (2005).

[16] O. Mülken, M. Dolgushev, and M. Galiceanu, Complex quantum networks: From universal breakdown to optimal transport, Phys. Rev. E **93**, 022304 (2016).

[17] K. V. Krutitsky, Ultracold bosons with short-range interaction in regular optical lattices, Phys. Rep. **607**, 1 (2016).

[18] J. Nokkala, F. Galve, R. Zambrini, S. Maniscalco, and J. Piilo, Complex quantum networks as structured environments: engineering and probing, Sci. Rep. **6**, 26861 (2016).

[19] D. Deutsch, Quantum computational networks, Proc. R. Soc. Lond. A **425**, 73–90 (1989).

[20] M. Christandl, N. Datta, T. C. Dorlas, A. Ekert, A. Kay, and A. J. Landahl, Perfect transfer of arbitrary states in quantum spin networks, Phys. Rev. A **71**, 032312 (2005).

[21] B. S., Quantum communication through spin chain dynamics: an introductory overview, Contemp. Phys. **48**, 13 (2007).

[22] A. Politi, M. J. Cryan, J. G. Rarity, S. Yu, and J. L. O'Brien, Silica-on-silicon waveguide quantum circuits, Science **320**, 646 (2008).

[23] A. Aspuru-Guzik and P. Walther, Photonic quantum simulators, Nature Physics **8**, 285 (2012).

[24] E. Farhi and S. Gutmann, Quantum computation and decision trees, Phys. Rev. A **58**, 915 (1998).

[25] A. M. Childs, R. Cleve, E. Deotto, E. Farhi, S. Gutmann, and D. A. Spielman, Exponential algorithmic speedup by a quantum walk, in *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '03 (Association for Computing Machinery, New York, NY, USA, 2003) pp. 59–68.

[26] V. Kendon, Quantum walks on general graphs, Int. J. Quantum Inf. **4**, 791 (2006).

[27] K. A., Perfect, efficient, state transfer and its application as a constructive tool, Int. J. Quantum Inf. **08**, 641 (2010).

[28] O. Mülken and A. Blumen, Continuous-time quantum walks: Models for coherent transport on complex networks, Phys. Rep. **502**, 37 (2011).

[29] C. Benedetti, M. A. C. Rossi, and M. G. A. Paris, Continuous-time quantum walks on dynamical percolation graphs, Europhys. Lett. **124**, 60001 (2019).

[30] S. Chakraborty, L. Novo, and J. Roland, Optimality of spatial search via continuous-time quantum walks, Phys. Rev. A **102**, 032214 (2020).

[31] V. Gualtieri, C. Benedetti, and M. G. A. Paris, Quantum-classical dynamical distance and quantumness of quantum walks, Phys. Rev. A **102**, 012201 (2020).

[32] K. Kadian, S. Garhwal, and A. Kumar, Quantum walk and its application domains: A systematic review, Comput. Sci. Rev. **41**, 100419 (2021).

[33] G. Bressanini, C. Benedetti, and M. G. A. Paris, Decoherence and classicalization of continuous-time quantum walks on graphs, Quantum Inf. Proc. **21**, 317 (2022).

[34] I. Gianani and C. Benedetti, Multiparameter estimation of continuous-time quantum walk Hamiltonians through machine learning, arXiv:2211.05626 (2022).

[35] D. Tamascelli, C. Benedetti, S. Olivares, and M. G. A. Paris, Characterization of qubit chains by Feynman probes, Phys. Rev. A **94**, 042129 (2016).

[36] L. Seveso, C. Benedetti, and M. G. A. Paris, The walker speaks its graph: global and nearly-local probing of the tunnelling amplitude in continuous-time quantum walks, J. Phys. A: Math. Theo. **52**, 105304 (2019).

[37] M. Annabestani, M. Hassani, D. Tamascelli, and M. G. A. Paris, Multiparameter quantum metrology with discrete-time quantum walks, Phys. Rev. A **105**, 062411 (2022).

[38] K. Rambhatla, S. E. D'Aurelio, M. Valeri, E. Polino, N. Spagnolo, and F. Sciarrino, Adaptive phase estimation through a genetic algorithm, Phys. Rev. Res. **2**, 033078

(2020).

[39] P. A. Knott, A search algorithm for quantum state engineering and metrology, New J. Phys. **18**, 073033 (2016).

[40] R. Nichols, L. Mineh, J. Rubio, J. C. F. Matthews, and P. A. Knott, Designing quantum experiments with a genetic algorithm, Quantum Science and Technology **4**, 45012 (2019).

[41] Evolutionary Approach to Quantum and Reversible Circuits Synthesis, Artificial Intelligence Review **20**, 361 (2003).

[42] M. Mitchell, *An Introduction to Genetic Algorithms* (The MIT Press, 1998).

[43] S. Katoch, S. Chauhan, and V. Kumar, A review on genetic algorithm: past, present, and future, Multimed. Tools Appl. **80**, 8091–8126 (2021).

[44] D. E. Knuth, *The Stanford GraphBase: a platform for combinatorial computing* (New York: AcM Press, 993) pp. 74–87.

[45] V. Hugo, *Les misérables* (Lacroix, Verboeckhoven & Company, 1862).

[46] H.-J. Briegel, W. Dür, J. I. Cirac, and P. Zoller, Quantum repeaters: The role of imperfect local operations in quantum communication, Phys. Rev. Lett. **81**, 5932 (1998).

[47] H. J. Kimble, The quantum internet, Nature **453**, 1023 (2008).

[48] S. Perseguers, M. Lewenstein, A. Acín, and J. I. Cirac, Quantum random networks, Nature Phys. **6**, 539 (2010).

[49] C. Benedetti, D. Tamascelli, M. G. Paris, and A. Crespi, Quantum spatial search in two-dimensional waveguide arrays, Phys. Rev. Appl. **16**, 054036 (2021).

[50] B. C. Coutinho, W. J. Munro, K. Nemoto, and Y. Omar, Robustness of noisy quantum networks, Comm. Phys. **5**, 105 (2022).

[51] A. Candeloro, C. Benedetti, M. G. Genoni, and M. G. A. Paris, Feedback-assisted quantum search by continuous-time quantum walks, Adv. Quantum Technol. **6**, 2200093 (2023).

[52] M. Karski, L. Förster, J. M. Choi, A. Steffen, W. Alt, D. Meschede, and A. Widera, Quantum walk in position space with single optically trapped atoms, Science **325(5937)**, 174–177 (2009).

[53] P. Imany, N. B. Lingaraju, M. S. Alshaykh, D. E. Leaird, and A. M. Weiner, Probing quantum walks through coherent control of high-dimensionally entangled photons, Science Advances **6**, eaba8066 (2020).

[54] A. W. Young, W. J. Eckner, N. Schine, A. M. Childs, and A. M. Kaufman, Tweezer-programmable 2D quantum walks in a Hubbard-regime lattice, Science **377(6608)**, 885 (2022).

I. APPENDIX

A. genetic algorithm

The algorithm begins with an initial population initialized by generating n_p random binary arrays Λ_i of size n_c, containing the couplings J_{xy}, which in this representation correspond to the genes of each individual. These n_p chromosomes correspond to the zeroth generation. While the number of generations is lower than n_g, we proceed as follows: We evaluate the score S_i of each string Λ_i using the Fitness function described in details in the next section. The best fitness score, corresponding to the lowest value, and the relative couplings are stored. If the score is equal to zero, the optimal solution has been found, the algorithm stops and returns it. If the condition is not met, the algorithm continues by selecting the fittest $p_e \cdot n_p$ chromosomes Λ_i and places them in a hall of fame, to be cloned in the following generation. Since the population size has to stay constant, we need to create the remaining $n_p(1-p_e)$ individuals which will populate the next generation together with those cloned from the hall of fame. In order to do so we select the best parents from the whole population (including the hall of fame). This is achieved with the Tournament selection function, which randomly selects k individuals at a time and returns the best among them (lowest fitness score). The random selection of the k competitors ensures that the chosen individuals are not necessarily the best in the population. In this way, genetic diversity is ensured to mitigate the presence of local minima. Once the parents are selected, they are mixed through the Crossover function, which returns two children which, with probability p_c, are composed by a mixture of the parents chromosomes. To further ensure genetic diversity, the genes of the children can undergo mutations with mutation probability p_m. When a mutation happens, the gene is flipped. The generated children together with the hall of fame constitute the new generation. The algorithm repeats until either a chromosome with fitness score equal to zero is found, or the maximum number of generations is reached. The pseudocode of the algorithm reported in Algorithm 1.

Algorithm 1 Genetic Algorithm

1: $gen \leftarrow 0$
2: Randomly generate n_p binary arrays $\{\Lambda_i\}$
3: $P_{gen} \leftarrow \{\Lambda_i\}$ ▷ Initialize population
4: **while** $gen < n_g$ **do**
5:      **for** $i = 0 \rightarrow n_p - 1$ **do**
6:          $S_i =$ FITNESS$(\Lambda_i, \pi(\{t_k\}, \Lambda^{\text{QW}}))$ ▷ Evaluate scores
7:      **end for**
8:      $best \leftarrow (\mathbf{Min}(S), \Lambda_{\mathbf{Min(S)}})$
9:      **if** $best[0] = 0$ **then**
10:          **return** $best$
11:      **end if**
12:      **for** $i = 0 \rightarrow p_e n_p - 1$ **do**
13:          $\text{HOF}_i \leftarrow (\Lambda_i, S_i)$ sorted by scores ▷ Hall of fame
14:      **end for**
15:      Insert HOF into P_{gen+1}
16:      **for** $j = 0 \rightarrow n_p(1-p_e)/2 - 1$ **do**
17:          $\Lambda_1^j, \Lambda_2^j \leftarrow$ TOURNAMENT(P_{gen}, S) ▷ Select parents
18:          Add CROSSOVER$(\Lambda_1^j, \Lambda_2^j)$ to children ▷ Children
19:      **end for**
20:      **for** $i = 0 \rightarrow n_p(1-p_e) - 1$ **do**
21:          Apply MUTATION(children$_i$) ▷ Mutation
22:      **end for**
23:      Insert children in P_{gen+1}
24:      $gen \leftarrow gen + 1$
25: **end while**

The values of the hyperparameters are reported in Table I:

Parameter	Value
n_g Maximum number of generations	100
n_p Population size	$2 \cdot n_c^2$
p_e Elitist probability	0.02
k Tournament competitors	6
p_c Crossover probability	0.85
p_m Mutation probability	0.05

TABLE I. Genetic algorithm parameters

B. Genetic Operations

We define the genetic functions which are used in the algorithm:

Fitness evaluation. The algorithm evaluates the fitness of each individual in the population Λ_i by evolving the initial state using the couplings in Λ_i and measuring the distance between the generated and measured probabilities concatenated at different times t_k, i.e. $\pi_x(\{t_k\}, \Lambda_i)$ and $\pi_x(\{t_k\}, \mathbf{\Lambda}^{\mathrm{QW}})$ respectively. The distance is measured with the Kullback-Leibler divergence (KLD), defined as:

$$\mathrm{KLD}(\Lambda_i) = \sum_x \pi_x(\{t_k\}, \Lambda_i) \log\left(\frac{\pi_x(\{t_k\}, \Lambda_i)}{\pi_x(\{t_k\}, \mathbf{\Lambda}^{\mathrm{QW}})}\right). \quad (3)$$

We note we have also tried metrics such as the Kolmogorov distance, obtaining analogous results.

Torunament selection. We select the individuals for reproduction among the whole population running repeated tournaments between k individuals at a time. We need to select $n_p(1 - p_e)$ individuals so that, since every couple will produce two children with probability p_c, the size of the population remains unchanged at each iteration. During each tournament, k individuals at random are selected among the whole population. The fittest one among the k (i.e. that with the smallest KLD) is chosen as a parent.

Crossover. Children are created two at a time. Both are initialized with the chromosome of one of their parents each. With a probability p_c, their chromosomes are crossed over. If the crossover happens, a random integer number smaller than n_c is selected, and serves as the splitting point for the chromosome of the two parents: one child's chromosome will be comprised of the chromosome of the first parent up to the splitting point, and of the second parent thereafter - and viceversa for the other child.

Mutation For each child, each gene can undergo a mutation with a probability p_m. This is achieved by selecting a random number between 0 and 1. If the number is smaller than p_m, then the gene will be flipped.

The pseudocode for each function is reported in Algorithm 2.

Algorithm 2 Genetic functions
___
1: **function** FITNESS($\Lambda_i, \pi(t_k, \mathbf{\Lambda}^{\mathrm{QW}})$):
2:     Evaluate $\pi(t_k, \Lambda_i)$
3:     Evaluate KLD($\pi(t_k, \Lambda_i), \pi(t_k, \mathbf{\Lambda}^{\mathrm{QW}})$)
4:     **return** KLD
5: **end function**

6: **function** TOURNAMENT(P_{gen}, S):
7:     id$\leftarrow$ random integer in $[0, n_p]$
8:     **for** $j = 0 \rightarrow k - 2$ **do**
9:       aux$\leftarrow$ random integer in $[0, n_p]$
10:      **if** $S[\mathrm{aux}] < S[\mathrm{id}]$ **then**
11:        id$\leftarrow$ aux
12:      **end if**
13:     **end for**
14:     **return** $\Lambda[\mathrm{id}]$
15: **end function**

16: **function** CROSSOVER(Λ_1, Λ_2):
17:     Generate a random integer x in $[0, 1]$
18:     **if** $x < p_c$ **then**
19:       $y \leftarrow$ random integer in $[0, n_c)$
20:       child$_1 \leftarrow$ concatenate($\Lambda_1[0 : y], \Lambda_2[y + 1 : n_c - 1]$)
21:       child$_2 \leftarrow$ concatenate($\Lambda_2[0 : y], \Lambda_1[y + 1 : n_c - 1]$)
22:     **end if**
23:     **return** child$_1$,child$_2$
24: **end function**

25: **function** MUTATION(child$_i$):
26:     **for** $j = 0 \rightarrow n_c - 1$ **do**
27:       Generate random x in $[0,1]$
28:       **if** $x < p_m$ **then**
29:         child$_i[j] \leftarrow 1-$child$_i[j]$
30:       **end if**
31:     **end for**
32:     **return** child$_i$
33: **end function**

C. Les Misérables graph

In order to test the algorithm on a graph with a random topology we adopt a simplified version of the graph describing the connections between the characters in the novel Les Misérables by V. Hugo. We use only the main characters, and we fix all the coupling strengths to 1. We start from $n = 5$ characters, and then increase n by introducing new characters. The resulting graphs are reported in Fig. 5

D. Results for additional topologies

In Fig. 6 we report the results obtained without noise for the line and circle topologies. Panels (a-f) and (h-m) show the couplings for the N=100 runs of the algorithm, while panels (g,n) show the success rate. In Fig. 7 we report the generations needed for convergence.

E. Complete Network with n=10

As shown in the main text, the complete network for n=10 is the most affected by local minima, which pre-vent the algorithm to converge to the correct solution dropping the success rate to 31%. This is because at the chosen times, there are configurations leading to similar probabilities than the complete one. However, it is suffi-cient to repeat the algorithm adding a third probability measured at $t_3 = 1$, to drastically increase the success rate up to 73%. The retrieved couplings are reported in Fig. 8.

F. Results with noise for additional resources

We report additional results with noise for $N_r = 1000$, and $N_r = 2500$ for the star and the complete networks with n=5. The success/fail rates are shown in Fig. 9
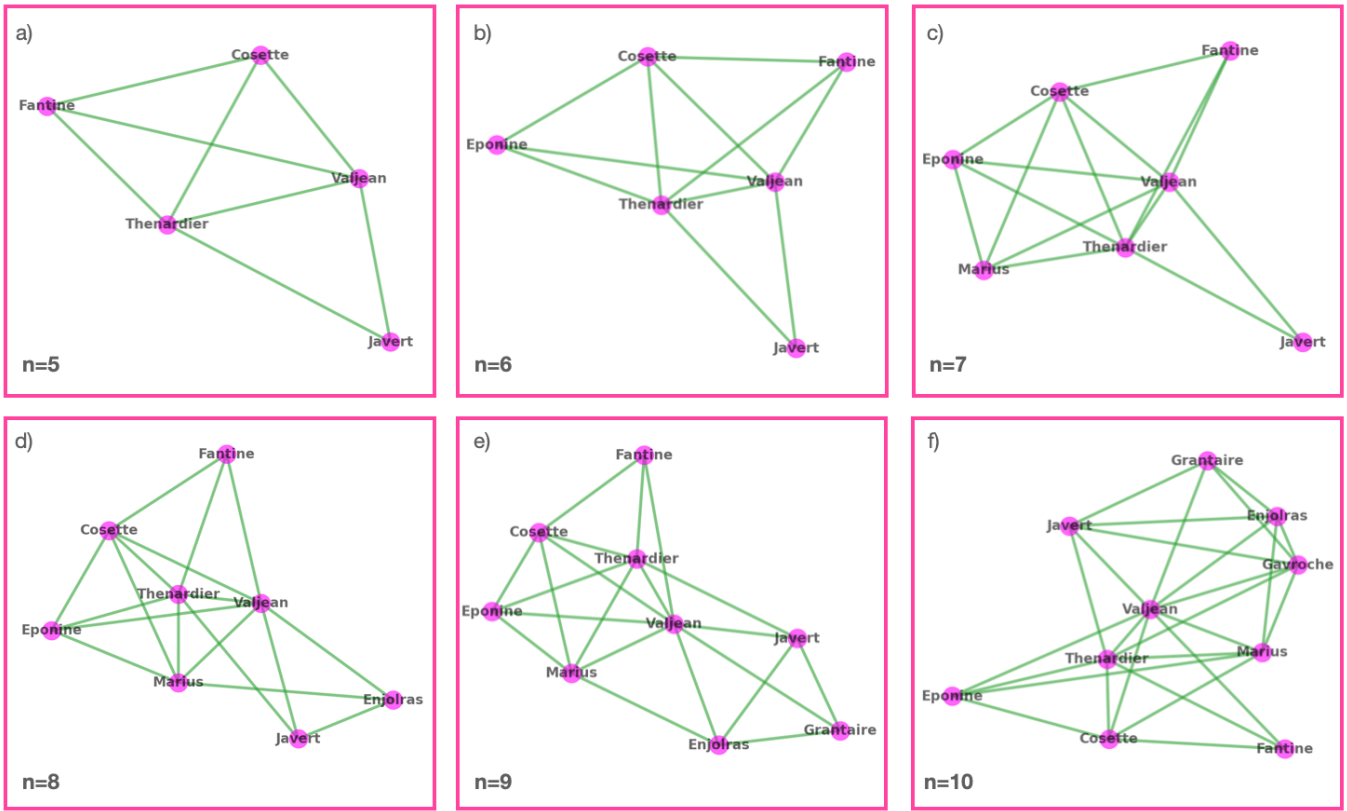
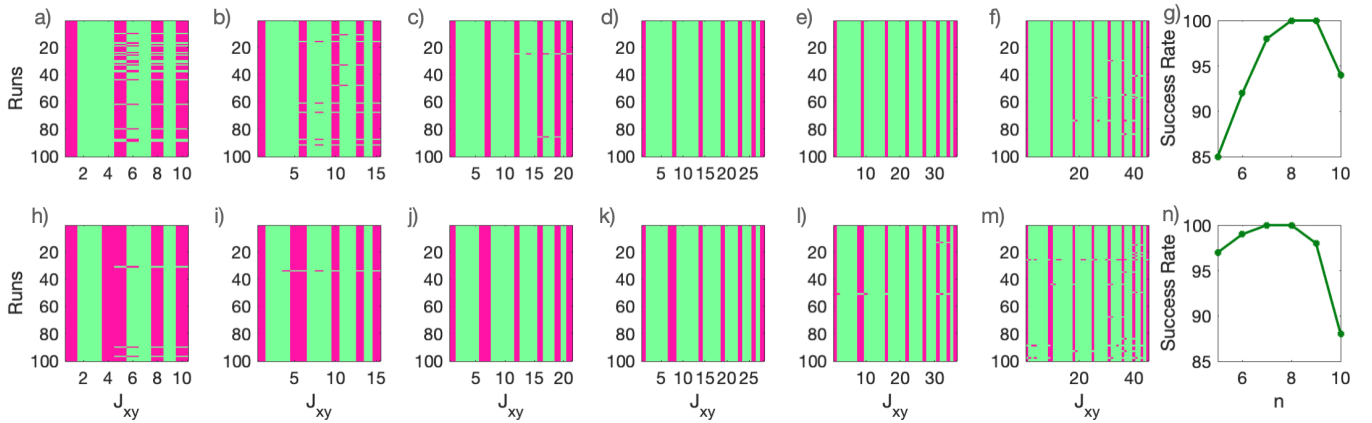FIG. 5. **Random Graph.** Composition of the random graph for n=5-10

FIG. 6. **Results for additional topologies - couplings.** Retrieved couplings over 100 runs for a line (a-f) and circle (h-m) network. g) Success rate for line (g) and circle (n).
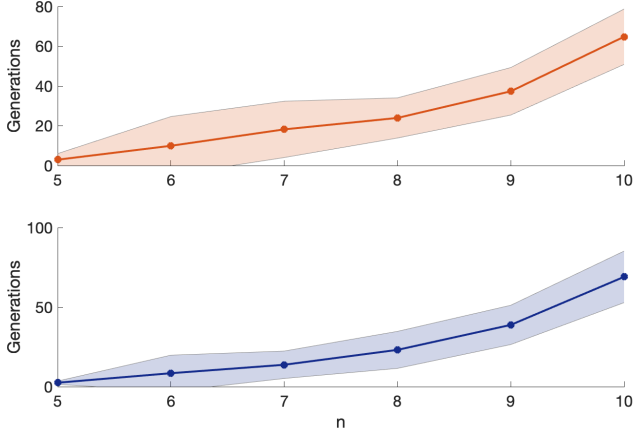
FIG. 7. Generations required for convergence for a line network (upper panel) and a circle network (lower panel)
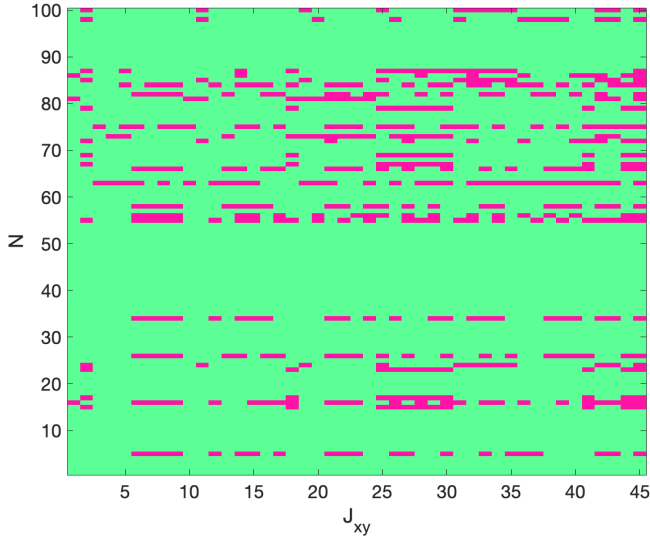

FIG. 8. Retrieved couplings for a n=10 complete network using probability distribution measured at three evolution times: $t_1 = 0.5$, $t_2 = 0.6$, and $t_3 = 1$
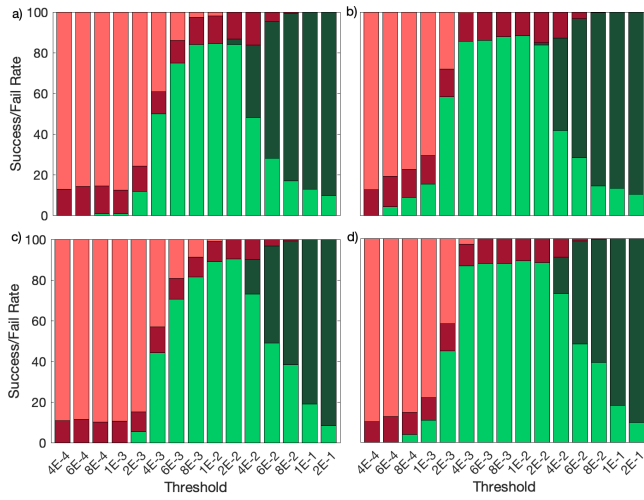
FIG. 9. (a-b) Results for star network, (c-d) results for complete graph with $N_r = 1000$ (a-c) and $N_r = 2500$ (b-d). Light red: false negatives, Dark red: true negatives, Light green: true positives, Dark green: false positives