Quantum approximate optimization with Gaussian boson sampling

Juan Miguel Arrazola,[*] Thomas R. Bromley,[†] and Patrick Rebentrost[‡]
Xanadu, 372 Richmond Street West, Toronto, Ontario, Canada M5V 1X6

(Received 5 April 2018; published 19 July 2018)

Hard optimization problems are often approached by finding approximate solutions. Here, we highlight the concept of proportional sampling and discuss how it can be used to improve the performance of stochastic algorithms for optimization. We introduce an NP-Hard problem called Max-Haf and show that Gaussian boson sampling (GBS) can be used to enhance any stochastic algorithm for this problem. These results are applied by enhancing the random search, simulated annealing, and greedy algorithms. With numerical simulations, we confirm that all algorithms are improved when employing GBS, and that a GBS-enhanced random search performs the best despite being the one with the simplest underlying classical routine.

DOI: 10.1103/PhysRevA.98.012322

In this unique time in history, we are witnessing the emergence of the first generation of quantum computers. The so-called noisy intermediate-scale quantum (NISQ) era [1] signals the first time that quantum devices capable of out-performing classical computers at specific tasks will become available for public use. With universal fault-tolerant quantum computing still a long road ahead, short-term focus has shifted to identifying the problems that current quantum devices can solve more efficiently than classical computers, without necessarily placing emphasis on their practical applications [2–11].

One such example is boson sampling, i.e., the task of generating random outcomes from the probability distribution induced by a collection of indistinguishable photons passing through a linear optics network [12]. Boson sampling was initially conceived as an experimentally appealing method of challenging the extended Church-Turing thesis, which states that all physically realizable processes can be efficiently simulated on a classical Turing machine. Since then, there has been significant effort to implement boson sampling [13–16] and to propose related models such as scattershot boson sampling [17–19] and Gaussian boson sampling [20,21] that are more amenable to experiments. In all these cases, the focus has been to outperform classical computers at the corresponding sampling task instead of using these devices for solving practical problems. Arguably the first clue that boson sampling could actually be linked to problems of interest came from Ref. [22], where it was shown that Gaussian boson sampling could be used to efficiently infer the vibronic spectra of complex molecules. Proof-of-principle experiments have also been recently reported [23,24].

In this work, which is a companion paper to [25], we show that Gaussian boson sampling (GBS) can be used to enhance stochastic algorithms for an NP-Hard optimization problem related to finding optimal submatrices. The main insight behind our results is that there exist problems where the GBS distribution naturally assigns a probability to each outcome that is proportional to their value with respect to the underlying figure of merit. This causes good outcomes to be sampled with high probability and bad results to almost never be observed. In cooperation with classical stochastic optimization algorithms, this intrinsic bias can be harnessed to give rise to hybrid algorithms that perform improved searches over the optimization landscape.

In the following, we discuss the concept of proportional sampling and outline its advantage with respect to uniform sampling in stochastic optimization. We introduce a generic optimization problem, which we call the Max-Haf problem, and we show that the output distribution from GBS is a proportional distribution for this task. Max-Haf is the canonical optimization problem for GBS and it serves as the template for extending GBS to other optimization problems, as we do for the densest k-subgraph problem in [25]. We give analytical results quantifying the expected improvement to a random search obtained by sampling from the GBS distribution, and we suggest how GBS can be used within more advanced heuristic algorithms. Our results are complemented with a numerical study of the performance of these enhanced algorithms on an example Max-Haf problem.

I. PROPORTIONAL SAMPLING

A combinatorial optimization problem can be cast as follows. Given a set $X = \{x_1, x_2, \ldots, x_N\}$ and an objective function $f : X \to \mathbb{R}$, the goal is to find an element $x^* \in X$ such that $f(x^*) \geqslant f(x)$ for all $x \in X$. Let $y_i := f(x_i)$ and define the set $Y = \{y_1, y_2, \ldots, y_N\}$ as well as the N-dimensional vector $y = (y_1, y_2, \ldots, y_N)$. We focus on the case in which $y_i \geqslant 0$ for all $i = 1, \ldots, N$. This can be guaranteed, for example, by finding a lower bound y_0 such that $f(x) \geqslant y_0$ for all $x \in X$ and introducing a new objective function $f'(x) = f(x) - y_0$ without altering the solution to the optimization problem.

Suppose that we can draw samples from X according to some probability distribution $P(x)$, so that we now think of Y as a random variable with corresponding probability distribution

[*]juanmiguel@xanadu.ai
[†]tom@xanadu.ai
[‡]pr@patrickre.com

$P(Y = y)$. The simplest stochastic algorithm for giving an approximate solution to the above combinatorial optimization problem, known as a random search, is to sample multiple values of X and choose the one with the largest value of the objective function. For a uniform distribution of samples, $P_U(x) = 1/N$, the expected value of Y is

$$\mathbb{E}_U(Y) = \frac{1}{N}\sum_{i=1}^{N} y_i = \frac{1}{N}\|y\|_1, \tag{1}$$

where $\|\cdot\|_p$ is the p-norm. Now suppose that we instead draw samples with respect to a distribution in which the probability of drawing a given element $x \in X$ is proportional to the objective function $f(x)$, i.e.,

$$P_\propto(x_i) = \frac{1}{\|y\|_1} y_i. \tag{2}$$

We refer to this type of sampling as *proportional sampling*. Here, the expected value of Y is

$$\mathbb{E}_\propto(Y) = \frac{1}{\|y\|_1}\sum_{i=1}^{N} y_i^2 = \frac{(\|y\|_2)^2}{\|y\|_1}. \tag{3}$$

For p-norms on finite-dimensional vector spaces, it holds that

$$\|y\|_p \leqslant N^{\frac{1}{p}-\frac{1}{q}}\|y\|_q \tag{4}$$

for any $q > p \geqslant 1$. The ratio between the expectation values of Y with respect to the proportional and uniform distributions hence satisfies

$$\frac{\mathbb{E}_\propto(Y)}{\mathbb{E}_U(Y)} = \left(\frac{\sqrt{N}\|y\|_2}{\|y\|_1}\right)^2 \geqslant 1, \tag{5}$$

where we have used Eq. (4) for the case $p = 1$, $q = 2$. This inequality means that sampling from the proportional distribution is never worse on average than sampling from the uniform distribution. The advantage obtained from proportional sampling increases when the elements of Y have very different values, in particular when there are a few elements that are much larger than the rest. Indeed, on one extreme, there is only a single nonzero element of Y, and the proportional distribution outputs the optimal element x^* with certainty, thus solving the combinatorial optimization problem with a single sample. On the other extreme, all of the elements of Y are equal and the proportional distribution reduces to the uniform distribution.

A. Enhancing random search algorithms

Let us build on the intuition that proportional sampling is beneficial by providing a framework for assessing the performance of a random search. Suppose that we sort X so that Y is in nondecreasing order, i.e., $y_1 \leqslant y_2 \leqslant \cdots \leqslant y_N$. The combinatorial optimization problem depends only on the relative values of Y, and we can hence change this random variable so that a given x_i has a corresponding y_i now given by $y_i = i/N$. The value y_i thus denotes the fraction of samples $x \in X$ whose objective function value $f(x)$ does not exceed $f(x_i)$. In typical optimization problems of interest, the size of the sample space N is exponentially large, so we can approximate Y as a continuous random variable with values in the interval (0,1] and probability density function $p(y)$. The goal of the combinatorial optimization is to find the $x^* \in X$ such that $Y = 1$, and we can find approximate solutions through a random search by taking κ samples of Y and finding the maximum. The result of the random search is another random variable,

$$Z = \max\{Y_1, Y_2, \ldots, Y_\kappa\}, \tag{6}$$

with $Z \in (0,1]$, and its corresponding probability density function is determined by $p(y)$.

When Y has a uniform distribution, i.e., $p(y) = 1$, the cumulative distribution of the random variable Z is

$$P_U(Z \leqslant z) = \prod_{j=1}^{\kappa} p(y \leqslant z) = z^\kappa, \tag{7}$$

and its probability distribution is

$$P_U(Z = z) = \kappa z^{\kappa-1}, \tag{8}$$

which is a beta distribution with mean $\kappa/(\kappa+1)$. We thus conclude that the expected output of the random search algorithm using uniform sampling is

$$\mathbb{E}_U(Z) = \frac{\kappa}{\kappa+1}. \tag{9}$$

This should be interpreted as stating that, after taking κ samples of X from the uniform distribution, on average we will find a value of $f(x)$ that is larger than a fraction $\kappa/(\kappa+1)$ of all possible values of $f(x)$ among $x \in X$.

Instead, if $p(y)$ corresponds to a proportional distribution on X, then

$$P_\propto(Z \leqslant z) = \left[\int_0^z p(y)dy\right]^\kappa, \tag{10}$$

and so

$$P_\propto(Z = z) = \frac{d}{dz}P_\propto(Z \leqslant z) = \kappa\left[\int_0^z p(y)dy\right]^{\kappa-1} p(z).$$

The expectation value of Y when sampled from the proportional distribution is thus

$$\mathbb{E}_\propto(Z) = \kappa\int_0^1 z\left[\int_0^z p(y)dy\right]^{\kappa-1} p(z)dz. \tag{11}$$

In general, the resultant expectation value will depend on the particular form of $p(y)$. To illustrate the effect of proportional sampling, we focus on the case in which $p(y)$ is exponentially increasing in y, given by the form

$$p(y) = \frac{\lambda e^{\lambda(y-1)}}{1-e^{-\lambda}}, \tag{12}$$

for some positive constant λ. For small values of λ, the distribution is close to uniform, whereas for larger values the probability is increasingly concentrated on the largest values, as illustrated in Fig. 1. With this distribution, the expectation value in Eq. (11) can be evaluated using standard integral tables as

$$\mathbb{E}_\propto(Z) = \frac{1}{\lambda}(1-e^\lambda)^{-\kappa}\{[(1-e^\lambda)^\kappa - 1]\lambda - H_\kappa \tag{13}$$

$$- \kappa e^\lambda\, {}_3F_2(1,1,1-\kappa;2,2;e^\lambda)\}, \tag{14}$$

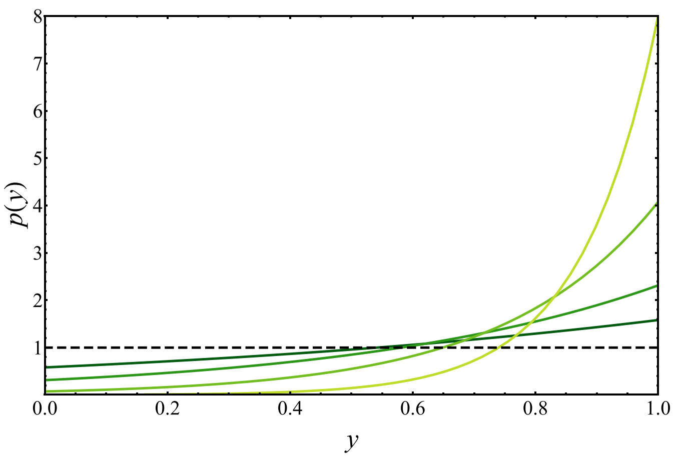

FIG. 1. Plot of a uniform distribution (dashed black line) and the probability distribution $p(y) = \lambda e^{\lambda(y-1)}/(1-e^{-\lambda})$ for $\lambda = 1, 2, 4$, and 8, where the distributions are further from uniform for increasing values of λ.

where H_κ is the κth harmonic number defined as

$$H_\kappa = \sum_{n=1}^{\kappa} \frac{1}{n} \tag{15}$$

and ${}_3F_2(a_1,a_2,a_3;b_1,b_2;z)$ is a generalized hypergeometric function. The improvement obtained by performing proportional sampling is illustrated in Fig. 2. Here we plot $\alpha := -\log_{10}[1-\mathbb{E}_\propto(Z)]$ as a function of κ for different values of λ. This function quantifies how close the expected result of a random search is to finding the optimal X with the value $Y = 1$, i.e., for a given value of α, then $\mathbb{E}_\propto(Z)$ is larger than a fraction $1-10^{-\alpha}$ of all possible values of Y. As expected, proportional sampling always leads to a significant improvement compared to uniform sampling, with the output of the optimization algorithm $\mathbb{E}_\propto(Z)$ being located in ranges that are orders of magnitude higher. The advantage increases with the parameter λ, showcasing the added benefits of proportional sampling when there is a small subset of elements with much larger values of the objective function.

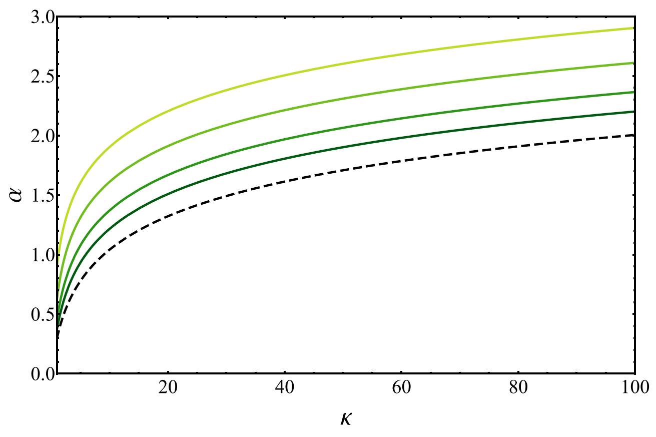

FIG. 2. Plot of the coefficient α as a function of the number of samples κ for proportional sampling compared to uniform sampling. The dashed black line corresponds to uniform sampling, and higher curves correspond to proportional sampling with distribution $p(y) = \lambda e^{\lambda(y-1)}/(1-e^{-\lambda})$ and values $\lambda = 1, 2, 4$, and 8. Proportional sampling always outperforms uniform sampling, and the advantage increases with λ.

The benefits of sampling from probability distributions that are tailored for specific problems have long been understood. Indeed, the goal of Markov chain Monte Carlo (MCMC) algorithms is to use a source of uniform randomness to generate samples from more general distributions. In this sense, proportional sampling can be interpreted as a type of importance sampling, a technique widely employed in optimization and Monte Carlo methods [26], where here the importance is achieved through proportionality. Nevertheless, the overhead that arises from MCMC algorithms can be extremely costly, and there exist distributions that cannot be sampled efficiently using classical algorithms. For this reason, it is highly desirable to have access to physical devices that can generate high-rate samples from useful probability distributions. As we show in the following section, Gaussian boson sampling systems can serve that role by sampling from proportional distributions that are useful for optimization problems related to finding optimal submatrices.

Before proceeding, it is important to remark on the methods used to solve combinatorial optimization problems and the resultant usefulness of stochastic algorithms. Approximate optimization becomes relevant for any problem that takes superpolynomial time to solve. In these cases, a desirable approach is to identify a polynomial time approximation scheme. However, such a scheme does not always exist for every problem, or it can be hard to identify. One alternative approach is to make use of deterministic- or stochastic-based heuristic algorithms to provide approximate solutions. The choice of deterministic or stochastic algorithm is problem-dependent. Proportional sampling is a tool to enhance stochastic heuristic algorithms.

II. GAUSSIAN BOSON SAMPLING

Gaussian boson sampling (GBS) [20,21] is a variant of boson sampling where the input to a linear optical network is a multimode Gaussian state instead of a collection of single photons. From an experimental perspective, this platform offers several advantages compared to traditional boson sampling, and it is therefore a leading candidate for the practical development of larger-scale boson sampling devices. In its simplest form, the input Gaussian state in GBS is a tensor product of squeezed vacuum states in each mode. We denote the possible outputs of GBS by vectors $S = (s_1, s_2, \dots, s_n)$, where there are a total of n input and output modes, and s_i is the number of photons detected in output mode i. It was shown in Ref. [20] that for a linear optics network characterized by the $n \times n$ unitary U and for input states with equal squeezing parameter $r > 0$ on all modes, the probability of observing an output pattern S is given by

$$\Pr(S) = \frac{1}{(\cosh r)^n} \frac{|\mathrm{Haf}(B_S)|^2}{s_1! s_2! \cdots s_n!}, \tag{16}$$

where $B = \tanh r \times UU^t$, and B_S is the matrix obtained by selecting the rows and columns corresponding to the nonzero entries of S. The function Haf$(\cdot)$ is the Hafnian of a matrix, defined as

$$\mathrm{Haf}(X) = \sum_{\sigma\in\mathrm{PMP}_{2m}} \prod_{j=1}^{m} X_{\sigma(2j-1),\sigma(2j)} \tag{17}$$

for even dimensional matrices X, where $\dim(X) = 2m$, and $\mathrm{Haf}(X) = 0$ for odd-dimensional X. Here, PMP_{2m} is the set of perfect-matching permutations on $2m$ elements. The Hafnian can be shown to be #P-Hard to approximate in a worst-case setting, a fact that has been leveraged to argue that even approximate simulation of GBS cannot be performed in polynomial time on classical computers unless the polynomial hierarchy collapses to the third level [20].

A. The Max-Haf problem

We now introduce the Max-Haf combinatorial optimization problem, defined on a complex-valued matrix B of any dimension. For Max-Haf, the problem is to find a submatrix B_S of B among those of a fixed even dimension $k = 2m$, with the largest Hafnian in absolute value, i.e.,

$$\begin{aligned}&\text{given } B,\text{ solve:}\\ &\underset{B_S}{\mathrm{argmax}}\ |\mathrm{Haf}(B_S)|^2\\ &\text{Subject to: } \dim(B_S) = k.\end{aligned} \tag{18}$$

This problem is NP-hard, as we now prove.

Theorem 1. The Max-Haf problem is NP-Hard to solve exactly in a worst-case setting.

Proof. The proof is by reduction to the maximum clique problem, which is known to be NP-Hard [27]. We consider the specific case in which B is the adjacency matrix of a graph G, so that the submatrices B_S are equivalent to subgraphs of G. The largest possible Hafnian of a graph is achieved by the complete graph, i.e., a clique, and a complete graph is the only graph that achieves this maximum value. Thus, given an algorithm to solve Max-Haf, we can iterate over all values of k and output the largest k such that the Hafnian of the optimal subgraph is equal to the Hafnian of a clique. This solves the maximum clique problem with $\sum_{k=1}^{n} k = O(n^2)$ calls, i.e., a polynomial number of calls, to the algorithm for Max-Haf. This implies that Max-Haf is also NP-Hard.

The NP-hardness of the Max-Haf problem means that in general we cannot expect to find exact solutions in polynomial time and must instead settle for approximate solutions. It is important to emphasize that the Max-Haf problem is introduced here due to its fundamental link to Gaussian boson sampling, as now discussed.

Suppose that B can be written as $\tanh r \times UU^t$. By construction, from Eq. (16) we know that the GBS distribution is a proportional distribution for Max-Haf, provided that the GBS output is postselected on samples corresponding to submatrices of dimension k; see Eq. (19) below. This enables all the advantages outlined in the previous section for stochastic algorithms aimed at providing approximate solutions. Moreover, approximate sampling from the GBS distribution is infeasible for classical devices, further motivating the importance of a physical GBS device. The NP-Hardness of Max-Haf also implies that we can in principle reduce any problem in NP to Max-Haf and then employ GBS to improve the performance of stochastic optimization algorithms for the resulting problem. Alternatively, we can identify quantitative connections between the Hafnian and relevant objective functions of other optimization problems, rendering samples from a GBS device directly useful for finding approximate solutions to these problems as well. In our companion paper [25], we show how GBS can be used for finding approximate solutions to the densest k-subgraph problem [28], which has several applications in data mining [29–32], bioinformatics [33,34], and finance [35].

B. Enhancement of approximate solutions through GBS

Here we focus on instances of GBS where the interferometer unitary U is drawn from the Haar measure. As shown in Ref. [20], in this case the $n \times n$ matrix $B = \tanh r \times UU^t$ is proportional to a unitary drawn from the circular orthogonal ensemble (COE). Moreover, for m sufficiently smaller than n, namely when $m = O(\sqrt{n})$, the matrix B_S is well approximated by a symmetric random Gaussian matrix with entries drawn from the complex normal distribution $\mathcal{N}(0, \frac{\tanh r}{\sqrt{n}})$. We focus on this setting in the following to evaluate the single-shot enhancement of using GBS proportional sampling, following methods introduced in [12].

As we have discussed in the previous section, a simple classical random search algorithm for solving the corresponding Max-Haf problem proceeds by sampling even k-dimensional submatrices B_S of B uniformly at random. The algorithm generates many such samples, calculates the Hafnians of the corresponding submatrices, and outputs the sample with the largest Hafnian. We represent the set of dimension k submatrices of B by the labels

$$\Gamma_{S,k} := \left\{ S : s_i \in \{0,1\}\,\forall\, i,\ \sum_i s_i = k \right\} \tag{19}$$

such that each element $S \in \Gamma_{S,k}$ describes the rows and columns of B selected to form the submatrix B_S. Since $|\Gamma_{S,k}| = \binom{n}{k}$, the expected value of $|\mathrm{Haf}(B_S)|^2$ under uniform sampling satisfies

$$\hat{\mu}_U := \binom{n}{k}^{-1} \sum_{S\in\Gamma_{S,k}} |\mathrm{Haf}(B_S)|^2. \tag{20}$$

The submatrices of B are approximated by symmetric Gaussian random matrices [20], so the above quantity is an approximation of the expectation value over the distribution $\mathcal{G}$ of these matrices, i.e., $\hat{\mu}_U \approx \mathbb{E}_{\mathcal{G}}[|\mathrm{Haf}(B_S)|^2]$.

We now compute the expectation value of $|\mathrm{Haf}(B_S)|^2$ when employing GBS, which we evaluate in terms of k, n, and r. Let p be the probability that a GBS sample satisfies $S \in \Gamma_{S,k}$. This probability approaches unity for $k \ll n$. Furthermore, let $q_{n,r}(k)$ be the user-controlled probability of sampling the required k photons when there are n input squeezed modes with squeezing parameter r, given by

$$q_{n,r}(k) = \binom{\frac{n+k}{2} - 1}{\frac{k}{2}} (sechr)^n (\tanh r)^k. \tag{21}$$

Using the fact that $s_1!s_2!\cdots s_n! = 1$ for all $S \in \Gamma_{S,k}$, we have from Eq. (16) that the expectation value of $|\mathrm{Haf}(B_S)|^2$ using

GBS satisfies

$$\begin{aligned}\hat{\mu}_{\mathrm{GBS}} :&= \frac{1}{pq_{n,r}(k)}\sum_{S\in\Gamma_{S,k}}|\mathrm{Haf}(B_S)|^2\frac{1}{(\cosh r)^n}|\mathrm{Haf}(B_S)|^2\\ &= \frac{1}{pq_{n,r}(k)(\cosh r)^n}\binom{n}{k}\binom{n}{k}^{-1}\sum_{S\in\Gamma_{S,k}}|\mathrm{Haf}(B_S)|^4\\ &\approx \frac{\binom{n}{k}}{q_{n,r}(k)(\cosh r)^n}\mathbb{E}_{\mathcal{G}}[|\mathrm{Haf}(B_S)|^4]. \end{aligned}\tag{22}$$

Thus, to compute the expected value of $|\mathrm{Haf}(B_S)|^2$ when sampling from the uniform and GBS distributions, we need to compute the first two moments $\mathbb{E}_{\mathcal{G}}[|\mathrm{Haf}(B_S)|^2]$ and $\mathbb{E}_{\mathcal{G}}[|\mathrm{Haf}(B_S)|^4]$ of the random variable $|\mathrm{Haf}(B_S)|^2$ with B_S sampled from k-dimensional symmetric Gaussian random matrices with entries drawn from $\mathcal{N}(0,\frac{\tanh r}{\sqrt{n}})$. For simplicity, we introduce the random variable

$$X = \frac{\sqrt{n}}{\tanh r}B_S, \tag{23}$$

so that X is now a random Gaussian matrix with entries drawn from the standard complex normal distribution $\mathcal{N}(0,1)$. It is straightforward to compute the first moment $\mathbb{E}_{\mathcal{G}}[|\mathrm{Haf}(X)|^2]$ using the definition in Eq. (17):

$$\begin{aligned}&\mathbb{E}_{\mathcal{G}}\left(\sum_{\sigma,\tau\in\mathrm{PMP}_k}\prod_{j=1}^{n}X_{\sigma(2j-1),\sigma(2j)}X^*_{\tau(2j-1),\tau(2j)}\right)\\ &= \mathbb{E}_{\mathcal{G}}\left(\sum_{\sigma\in\mathrm{PMP}_k}\prod_{j=1}^{n}|X_{\sigma(2j-1),\sigma(2j)}|^2\right)\\ &= \sum_{\sigma\in\mathrm{PMP}_k}\prod_{j=1}^{n}\mathbb{E}_{\mathcal{G}}(|X_{\sigma(2j-1),\sigma(2j)}|^2) = \sum_{\sigma\in\mathrm{PMP}_k}1 = (k-1)!!.\end{aligned}\tag{24}$$

The calculation for the second moment $\mathbb{E}_{\mathcal{G}}[|\mathrm{Haf}(X)|^4]$ is significantly more involved, the details of which we defer to the Appendix. The result is

$$\mathbb{E}_{\mathcal{G}}[|\mathrm{Haf}(X)|^4] = k! \tag{25}$$

From Eqs. (23), (24), and (25), we get

$$\mathbb{E}_{\mathcal{G}}[|\mathrm{Haf}(B_S)|^2] = \frac{(\tanh r)^k}{n^{k/2}}(k-1)!!, \tag{26}$$

$$\mathbb{E}_{\mathcal{G}}[|\mathrm{Haf}(B_S)|^4] = \frac{(\tanh r)^{2k}}{n^k}k!. \tag{27}$$

Finally, using these values we obtain the ratio between the expectation values μ_{GBS} and μ_U, given by

$$R := \frac{\hat{\mu}_{\mathrm{GBS}}}{\hat{\mu}_U} = \frac{\binom{n}{k}k!!}{\binom{\frac{n+k}{2}-1}{\frac{k}{2}}n^{k/2}}. \tag{28}$$

This ratio characterizes the advantage obtained by using GBS, as illustrated in Fig. 3. Note that this ratio does not depend on the squeezing parameter r, whose only role is to determine the probability $q_{n,r}(k)$ of observing the desired number of photons.

This result quantifies the fundamental feature that makes GBS useful: whenever we sample a submatrix, the GBS distribution has a preference for selecting submatrices that have a large absolute Hafnian while neglecting those with negligible Hafnians. The net result of this effect is that submatrices with larger Hafnians are on average sampled, leading to better approximations to the Max-Haf problem in a random search algorithm. However, random search algorithms, although powerful, do not exploit the local structure of the optimization landscape. Alternative stochastic algorithms employ a combination of exploration of the landscape—which is done by random sampling—and exploitation of local structure. We now outline how GBS can be used to enhance both of these elements within stochastic algorithms.

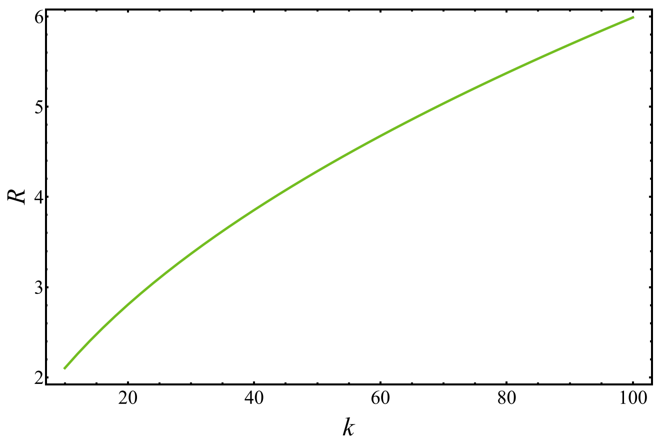

FIG. 3. Plot of the ratio R of Eq. (28) quantifying the improvement of using GBS sampling as opposed to uniform sampling for finding approximate solutions to the Max-Haf problem with submatrix size k. We plot R for different values of even k when $n = k^2$. The ratio is larger than 1 for all $k \geqslant 2$, showcasing the advantage obtained by using GBS.

III. USING GBS TO ENHANCE STOCHASTIC OPTIMIZATION ALGORITHMS

Stochastic algorithms usually employ randomness in two ways: exploration and tweaking, which is used in the exploitation phase. Let us first show how GBS can be used to enhance exploration, which in the case of Max-Haf corresponds to a random selection of a set of submatrices B_S. Optimization algorithms usually perform this step either by selecting submatrices uniformly at random, or by seeding them, i.e., picking a few that are believed to be good submatrices. This latter strategy is often infeasible when the number of possibilities is exponentially large and may also restrict the resultant approximate solutions to suboptimal local maxima since there are no guarantees that the initial picks are indeed good candidates. For this reason, uniform initialization is used as a reliable approach. However, as we have shown, it is always preferable to replace uniform sampling with proportional sampling. In the case of Max-Haf, this corresponds to sampling from the suitably postselected GBS distribution. We thus define a function, GBS-Explore, to be employed in heuristic algorithms:

Definition 1. GBS-Explore(k,B): Given the matrix B, generate a sample S from the conditional GBS distribution $\Pr(S|S\in\Gamma_{S,k}) = \frac{1}{pq_{n,r}(k)(\cosh r)^n}|\mathrm{Haf}(B_S)|^2$. Output the corresponding matrix B_S.

In a physical implementation, sampling from $\Pr(S|S \in \Gamma_{S,k})$ can be performed by generating outcomes from the complete distribution $\Pr(S)$ and keeping only outcomes with k photons. This incurs a penalty of $pq_{n,r}(k)$ in terms of the sampling rate $q_{n,r}(k)$, which can be optimized by selecting an appropriate value of the squeezing parameter r, and the valid sample probability p, which approaches 1 for $k \ll n$.

Tweaking is an operation used within exploitation where samples are randomly modified to produce new outputs in the vicinity of the original sample. In our case, since each possible submatrix is specified by a binary vector $S = (s_1, s_2, \dots, s_n)$ with $\sum_{i=1}^{n} = k$, the only freedom for modification is to change the location of some of the nonzero entries of S. This is achieved by randomly selecting a subset of nonzero entries and setting them to zero, and then randomly selecting a subset of zero entries and setting them to 1. To make use of GBS in this tweaking stage, we sample from the GBS distribution to identify which entries to change from 0 to 1. Here, the idea is that GBS will prefer to highlight new submatrices with a large absolute Hafnian to substitute into the candidate submatrix B_S. We formalize this into the subroutine GBS-Tweak, where $\ell \in \{0,2,4,\dots,k-2\}$ fixes the minimum number of nonzero entries of S to be left unchanged:

Definition 2. GBS-Tweak(S,ℓ,B):

(i) Generate a binary vector S' with $\ell + L$ entries randomly selected from the nonzero entries of S and with the remaining entries set to zero, where $L \in \{0,1,\dots,k-\ell-1\}$ is selected uniformly at random.

(ii) Generate a sample T from the conditional GBS distribution $\Pr(T|\Gamma_{T,\ell}) = \frac{1}{pq_{n,r}(\ell)(\cosh r)^{2n}}|\mathrm{Haf}(B_T)|^2$, producing samples T with ℓ nonzero entries.

(iii) Fix a binary vector T' by selecting L of the nonzero entries of T at random and setting the remaining entries to zero. If there is any overlap between the nonzero entries of T' and S', repeat this step.

(iv) Calculate the vector R given by combining the nonzero entries of S' with T' and return the submatrix B_R.

The result of GBS-Tweak is a submatrix B_R that has been tweaked from B_S by using GBS to preferentially select submatrices of size $\ell < k$ with large absolute Hafnians.

GBS-Explore and GBS-Tweak can be used to construct quantum-enhanced versions of any stochastic algorithm employing randomness for exploration and tweaking. This is the essential merit of GBS as a tool for approximate optimization: anything that a classical stochastic optimization algorithm can do, we can do better by enhancing it using GBS randomness. More generally, since classical algorithms must always operate on an initial source of uniform randomness, even algorithms whose goal is to sample from nonuniform distributions can be expected to improve by replacing their initial uniform sampling with GBS.

A. Applying GBS to example algorithms

To exhibit the enhancement of stochastic algorithms through GBS for approximate optimization, we focus on two well-known algorithms: simulated annealing [36] and a greedy algorithm. We describe each algorithm and propose the enhanced versions using the GBS-Explore and GBS-Tweak subroutines. A comparison of performance between the two versions of each algorithm, i.e., enhanced with GBS sampling and the traditional uniform sampling, is then carried out in the following section.

Simulated annealing is a heuristic optimization algorithm that combines elements of random search and hill climbing. It begins with an exploration phase, where an initial submatrix is generated, and then proceeds to exploit the local landscape by repetitively tweaking the submatrix. For each tweak, if the absolute Hafnian of the proposal submatrix is larger than the current one, it is kept. If its absolute Hafnian is smaller, the proposal submatrix can still be retained with a probability that depends on the difference between the absolute Hafnians and a time-evolving temperature parameter. The temperature is initially set high, so new submatrices with lower absolute Hafnians are often accepted, leading effectively to a random search that avoids getting stuck in local minima. As the algorithm progresses, the temperature is lowered and only better submatrices are kept, leading to an effective hill-climbing behavior. The GBS version of simulated annealing is shown as Algorithm 1.

Algorithm 1 GBS Simulated Annealing for Max-Haf

```
$B$ : Input matrix
$k$ : Dimension of submatrix
$\ell$ : Minimum number of entries to change when tweaking
$t$ : Initial temperature
$a_{\max}$ : Number of steps
$B_S$ = GBS-Explore$(k,B)$
Best = $B_S$
for a from 1 to $a_{\max}$:
    $B_R$ = GBS-Tweak$(S,\ell,B)$
    if $|\mathrm{Haf}(B_R)| > |\mathrm{Haf}(B_S)|$:
        $B_S = B_R$
    else:
        Set $B_S = B_R$ with prob. $\exp\left[\frac{|\mathrm{Haf}(B_R)|-|\mathrm{Haf}(B_S)|}{t}\right]$
    if $|\mathrm{Haf}(B_S)| > |\mathrm{Haf}(\mathrm{Best})|$:
        Best = $B_S$
    Decrease $t$
end for
Output Best, Haf(Best)
```

Algorithm 2 GBS Greedy Algorithm for Max-Haf

```
$B$ : Input matrix
$k$ : Dimension of submatrix
$B_S$ = GBS-Explore$(k,B)$
for $i$ from 1 to $k$:
    BestHaf = 0
    Bestj = 0
    for $j$ from 1 to $n$:
        $C$ = Matrix obtained by replacing $i$th row/column
        of $B_S$ with $j$th row/column of $B$.
        if $|\mathrm{Haf}(C)| >$ BestHaf:
            BestHaf = $|\mathrm{Haf}(C)|$
            Bestj = j
        end if
    end for
    Update $B_S$ by replacing its $i$th row/column with the
    row/column corresponding to Bestj.
end for
Output $B_S$, Haf$(B_S)$
```

The greedy algorithm begins with exploration by randomly generating a submatrix B_S from GBS. It then proceeds to generate candidate replacement submatrices by exhaustively substituting the first row and column of B_S with all possible remaining rows and columns of B, while keeping all other rows and columns of B_S fixed. The best of these candidates is kept and set as B_S. The process is repeated for all rows and columns of B_S. Note that tweaking in the greedy algorithm is deterministic, and hence cannot be enhanced through GBS randomness. However, the initial exploration step can still be improved, leading us to the GBS Greedy Algorithm outlined in Algorithm 2. This algorithm can be repeated many times with a new random initial submatrix in each run. The resulting output is the best submatrix among all repetitions. The quantum enhancement of this hybrid algorithm arises because the starting submatrix will on average have a larger absolute Hafnian than one obtained from uniform sampling.

B. Numerical study of enhancement through GBS

Here we quantitatively analyze the performance of a random search, simulated annealing, and greedy algorithms in both their GBS and uniform versions. To do this, we set up an example Max-Haf problem by generating a 30×30 random matrix $B = \tanh r \times UU^t$, where U is drawn from the Haar measure. B is given explicitly in the supplemental material [37]. We fix the instance of the problem to be finding the submatrix of dimension 10 with the largest absolute Hafnian. To exactly perform GBS sampling, we carry out a brute force calculation of the GBS probability distribution conditioned on 10 output photons with no more than one photon in each mode [38], and then we sample from the resulting distribution. The brute force calculation also allows us to find the exact solution of Max-Haf [37]. Note that the dimensions of the matrix and submatrix in this instance of Max-Haf are chosen to allow us to perform the numerical simulations in reasonable time.

We report results for algorithms making 1000 evaluations of the objective function, which is the largest overhead in each algorithm. For a random search and simulated annealing, the number of evaluations is equivalent to the number of samples. However, for the greedy algorithm, given an initial submatrix of dimension 10, the algorithm iterates over at least $21 \times 10 = 210$ other submatrices, evaluating the Hafnian for each. Thus, to compare fairly to the results of a random search and simulated annealing, we restrict ourselves to five repetitions of the greedy algorithm. Our results complement the study in [25] testing the enhancement through GBS of a random search and simulated annealing in finding dense subgraphs.

The performances of a random search, simulated annealing, and the greedy algorithm are shown in Fig. 4. The simulated annealing parameters were set to $T = 3 \times 10^{-5}$ and $\ell = 6$, with a linear cooling schedule used. As expected, we observe an improvement when using GBS over uniform sampling in all cases. The advantage is most significant for a random search, which is the algorithm harnessing the most randomness. The improvement is more modest for simulated annealing and the greedy algorithm. This is to be expected for the greedy algorithm since it is mostly deterministic: the only random component is the initial exploration. On the other hand,

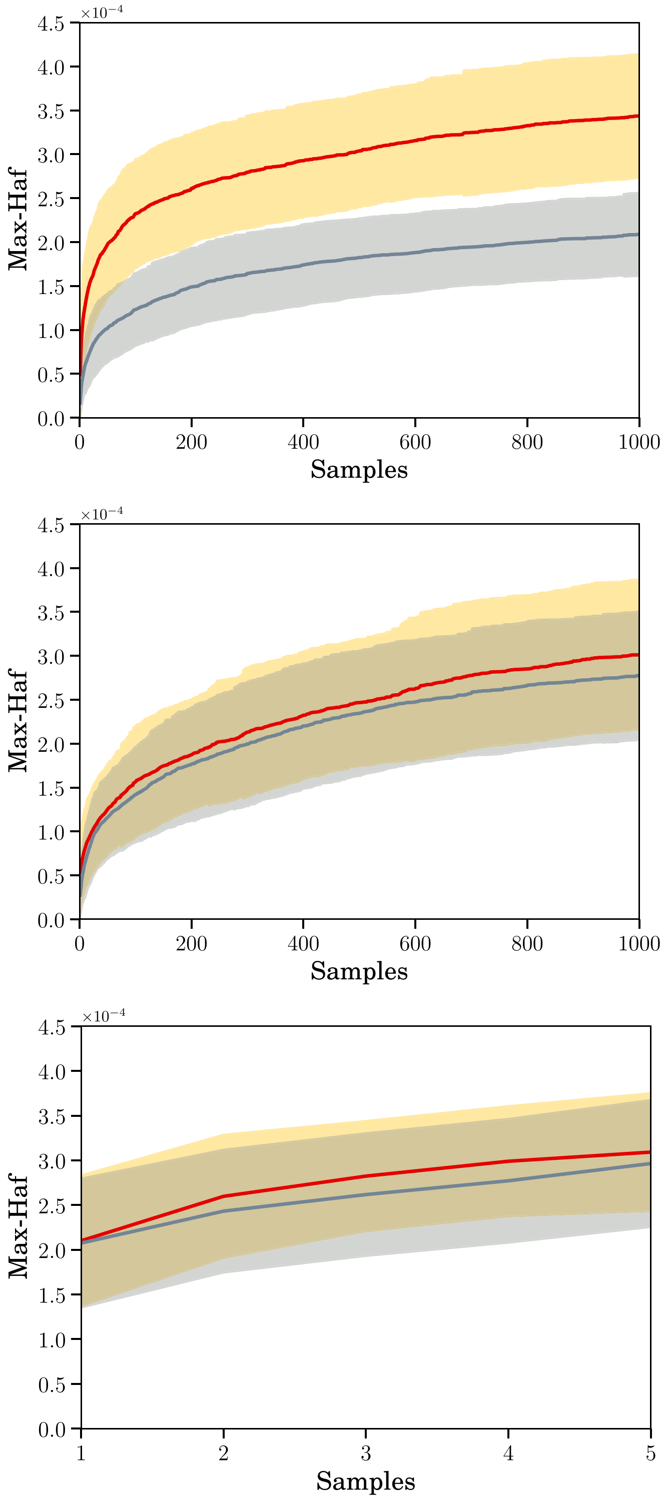

FIG. 4. Performance of random search (top), simulated annealing (middle), and greedy (bottom) algorithms for Max-Haf. The goal is to find an optimal submatrix of dimension 10 from a 30×30 starting matrix B drawn from the circular orthogonal ensemble. The top red curves correspond to GBS random search and the bottom gray curves to uniform random search. The solid curve is the average observed over 400 repetitions, and the continuous error bars represent one standard deviation. Max-Haf can be solved by brute force with the result 7.17×10^{-4} [37].

the small advantage for simulated annealing indicates that GBS-tweak is only slightly more useful than a uniform tweak. For certain situations, as in Ref. [25], it is possible to further optimize GBS-tweak to increase the overall advantage. A striking feature of these results is that the GBS random search is the best performing algorithm among all variations, whereas a uniform random search is the worst. This highlights the power of proportional sampling for optimization: even the simplest algorithm we have studied, when enhanced with GBS, surpasses the capabilities of more sophisticated classical algorithms.

The algorithms we have discussed so far require several evaluations of a Hafnian, which is generally a #P-Hard quantity to compute. For large problem sizes, when this evaluation becomes infeasible, it may be desirable to employ alternative algorithms such as Bayesian optimization [39] that are more suitable for handling objective functions that are costly to evaluate. We reiterate that a random search, greedy algorithms, and simulated annealing are only examples, but the improvement from GBS can occur for any stochastic algorithm relying on exploration and tweaking. It is important to further investigate the extent to which GBS-enhanced stochastic algorithms for Max-Haf may outperform purely classical methods, in particular deterministic algorithms that are not straightforwardly improved by using GBS.

Finally, the Max-Haf problem is the canonical optimization task for GBS, so studying it provides the best insights for understanding how GBS can be applied to approximate optimization. However, the true potential of GBS is unlocked when we carry these insights over to a wider class of optimization problems, as shown in detail in Ref. [25] for the densest k-subgraph problem.

IV. DISCUSSION

We have argued that Gaussian boson sampling, besides providing an avenue for challenging the extended Church-Turing thesis, can be a valuable tool in enhancing stochastic algorithms for NP-Hard optimization problems. Our results provide additional evidence to challenge the common perception that boson sampling devices, while performing a task that is hard to replicate with classical computers, are not immediately relevant to problems of practical interest. Furthermore, classical sampling algorithms employ deterministic rules and sources of uniform randomness to generate samples from target distributions. This approach can often introduce significant overhead, and it is incapable of efficiently sampling from arbitrary distributions. Boson sampling devices, and quantum computers more generally, when programed in a deliberate and engineered fashion, have the potential to efficiently sample from a larger class of probability distributions, leading to improved algorithms for optimization and simulation.

Future work should focus on expanding the range of applications of Gaussian boson sampling for approximate optimization. One particularly appealing direction is to understand how we can generate proportional distributions for problems beyond submatrix optimization, for instance by employing techniques from machine learning. It will also be important for a given problem to benchmark the enhancement through GBS against optimized classical algorithms.

ACKNOWLEDGMENTS

The authors thank Alex Arkhipov, Kamil Brádler, Pierre-Luc Dallaire-Demers, Nathan Killoran, Seth Lloyd, and Christian Weedbrook for valuable discussions.

APPENDIX

We perform a detailed calculation of the quantity $\mathbb{E}_{\mathcal{G}}[|\mathrm{Haf}(X)|^4]$. Our calculation follows closely a strategy from Ref. [12] to compute analogous moments of permanents of Gaussian random matrices. We have

$$\begin{aligned}\mathbb{E}_{\mathcal{G}}[|\mathrm{Haf}(X)|^4] &= \mathbb{E}_{\mathcal{G}}\left(\sum_{\sigma,\tau,\alpha,\beta\in\mathrm{PMP}_{2m}}\prod_{j=1}^{n} X_{\sigma(2j-1),\sigma(2j)}X_{\tau(2j-1),\tau(2j)}X^*_{\alpha(2j-1),\alpha(2j)}X^*_{\beta(2j-1),\beta(2j)}\right)\\ &= \sum_{\sigma,\tau,\alpha,\beta\in\mathrm{PMP}_{2m}}\mathbb{E}_{\mathcal{G}}\left(\prod_{j=1}^{n} X_{\sigma(2j-1),\sigma(2j)}X_{\tau(2j-1),\tau(2j)}X^*_{\alpha(2j-1),\alpha(2j)}X^*_{\beta(2j-1),\beta(2j)}\right) = \sum_{\sigma,\tau,\alpha,\beta\in\mathrm{PMP}_{2m}} M(\sigma,\tau,\alpha,\beta),\end{aligned}$$

where we have implicitly defined the function $M(\sigma,\tau,\alpha,\beta)$. We write $\sigma\cup\tau=\alpha\cup\beta$ if

$$\begin{aligned}&\{(\sigma(1),\sigma(2)),(\tau(1),\tau(2)),\ldots,(\sigma(2m-1),\sigma(2m)),(\tau(2m-1),\tau(2m-1))\}\\ &\quad=\{(\alpha(1),\alpha(2)),(\beta(1),\beta(2)),\ldots,(\alpha(2m-1),\alpha(2m)),(\beta(2m-1),\beta(2m-1))\}.\end{aligned}$$

Note that $M(\sigma,\tau,\alpha,\beta)\neq 0$ if and only if $\sigma\cup\tau=\alpha\cup\beta$, so we can restrict our attention to the case $\sigma\cup\tau=\alpha\cup\beta$. For each $j=1,2,\ldots,n$ there are two possibilities. If $\sigma(2j-1),\ \sigma(2j)\neq\tau(2j-1),\tau(2j)$, then

$$\begin{aligned}&\mathbb{E}_{\mathcal{G}}(X_{\sigma(2j-1),\sigma(2j)}X_{\tau(2j-1),\tau(2j)}X^*_{\alpha(2j-1),\alpha(2j)}X^*_{\beta(2j-1),\beta(2j)})\\ &= \mathbb{E}_{\mathcal{G}}(|X_{\sigma(2j-1),\sigma(2j)}|^2|X_{\tau(2j-1),\tau(2j)}|^2)\\ &= \mathbb{E}_{\mathcal{G}}(|X_{\sigma(2j-1),\sigma(2j)}|^2)\mathbb{E}_{\mathcal{G}}(|X_{\tau(2j-1),\tau(2j)}|^2) = 1.\end{aligned}$$

Similarly, if $\sigma(2j-1),\sigma(2j)=\tau(2j-1),\tau(2j)$, we have

$$\begin{aligned}&\mathbb{E}_{\mathcal{G}}(X_{\sigma(2j-1),\sigma(2j)}X_{\tau(2j-1),\tau(2j)}X^*_{\alpha(2j-1),\alpha(2j)}X^*_{\beta(2j-1),\beta(2j)})\\ &= \mathbb{E}_{\mathcal{G}}(|X_{\sigma(2j-1),\sigma(2j)}|^4) = 2.\end{aligned}$$

We conclude that, whenever $\sigma\cup\tau=\alpha\cup\beta$, it holds that

$$M(\sigma,\tau,\alpha,\beta)=2^{K(\sigma,\tau)}, \tag{A1}$$

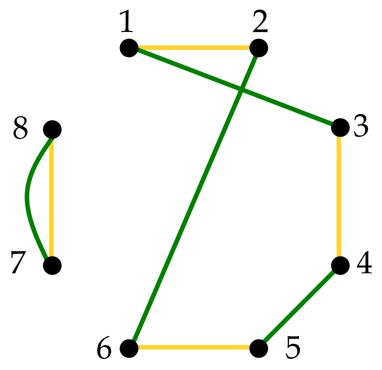

FIG. 5. Graph $G_\xi(4)$ for the perfect matching permutation $\xi = (13)(26)(45)(78)$. There is a 1-cycle corresponding to the equivalent action of the two permutations on the nodes 7 and 8. The remaining edges form a 3-cycle for which there are two inequivalent choices of permutations α,β such that $\sigma_0 \cup \xi = \alpha \cup \beta$, namely (i) $\alpha = (12)(34)(56)$, $\beta = (13)(26)(45)$ and (ii) $\alpha = (13)(26)(45)$, $\beta = (12)(34)(56)$.

where $K(\sigma,\tau)$ is the number of j such that $\sigma(2j-1), \sigma(2j) = \tau(2j-1), \tau(2j)$. Finally, we obtain

$$\mathbb{E}_\mathcal{G}[|\mathrm{Haf}(X)|^4] = \sum_{\sigma,\tau\in \mathrm{PMP}_{2m}} 2^{K(\sigma,\tau)} N(\sigma,\tau),$$

where $N(\sigma,\tau)$ is the number of permutations α,β such that $\sigma\cup\tau = \alpha\cup\beta$. This expression can be simplified further as follows. Let σ_0 be the identity permutation, which is a perfect matching permutation. Furthermore, for a given perfect matching permutation σ, let σ^{-1} be the unique perfect matching permutation such that $\sigma^{-1}\sigma = \sigma_0$. We then have

$$\begin{aligned}\mathbb{E}_\mathcal{G}[|\mathrm{Haf}(X)|^4] &= \sum_{\sigma,\tau\in \mathrm{PMP}_{2m}} 2^{K(\sigma,\tau)} N(\sigma,\tau)\\ &= [(2m-1)!!]^2 \mathop{\mathbb{E}_\mathcal{G}}_{\sigma,\tau}[2^{K(\sigma,\tau)}N(\sigma,\tau)]\\ &= \mathop{\mathbb{E}_\mathcal{G}}_{\sigma,\tau}[2^{K(\sigma^{-1}\sigma,\sigma^{-1}\tau)}N(\sigma^{-1}\sigma,\sigma^{-1}\tau)]\\ &= \mathop{\mathbb{E}_\mathcal{G}}_{\xi\in\mathrm{PMP}_{2m}}[2^{K(\sigma_0,\xi)}N(\sigma_0,\xi)]\\ &= (2m-1)!! \sum_{\xi\in\mathrm{PMP}_{2m}} 2^{K(\sigma_0,\xi)}N(\sigma_0,\xi).\end{aligned}$$

We now need to calculate $\sum_{\xi\in\mathrm{PMP}_{2m}} 2^{K(\sigma_0,\xi)}N(\sigma_0,\xi)$.

Define a graph $G_\xi(m)$ of $2m$ nodes as follows. For all $j = 1,2,\dots,m$, draw an edge between nodes $(2j-1,2j)$. This is the graph representation of the identity permutation σ_0. Similarly, for $\xi = (i_1 j_1)(i_2 j_2)\cdots(i_m j_m)$, draw an edge between nodes (i_k, j_k) for all $k = 1,2,\dots,m$. This is the graph representation of ξ. An example of a graph $G_\xi(m)$ is shown in Fig. 5. Note that, by construction, $G_\xi(m)$ is a union of cycles.

Whenever $(\sigma_0(2j-1),\sigma_0(2j)) = (\xi(2j-1),\xi(2j))$, there will be a 1-cycle in the graph $G_\xi(m)$ consisting of the two edges joining nodes $2j-1$ and $2j$. Thus, $K(\sigma_0,\xi)$ is equal to the number of 1-cycles in $G_\xi(m)$. Every larger cycle in $G_\xi(m)$ gives rise to two inequivalent choices of α,β such that $\sigma_0\cup\xi = \alpha\cup\beta$ because we can assign the action of α to coincide with either σ_0 or ξ when restricted to the nodes in the cycles, and

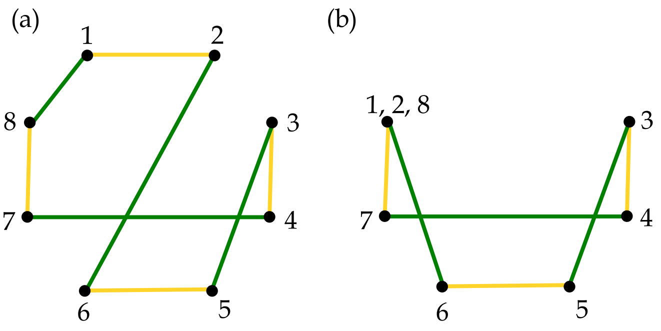

FIG. 6. When the permutation ξ has a swap of the form $(1j)$ with $j \neq 2$, we can combine the nodes 1,2 and j into a single node without changing the number of cycles in the graph $G_\xi(m)$. Part (a) shows the graph $G_\xi(4)$ for the perfect matching permutation $\xi = (18)(26)(35)(26)$ and (b) shows a graph with an equal number of cycles obtained by combining nodes 1, 2, and 8 into a single node.

similarly for β. We therefore have that

$$N(\sigma_0,\xi) = 2^{\Gamma_\xi}, \tag{A2}$$

where 2^{Γ_ξ} is the number of cycles in $G_\xi(m)$ of length equal to or larger than 2. Combining these results, we conclude that

$$\mathbb{E}_\mathcal{G}[|\mathrm{Haf}(X)|^4] = (2m-1)!! \sum_{\xi\in\mathrm{PMP}_{2m}} 2^{\mathrm{cyc}(\xi)}, \tag{A3}$$

where $\mathrm{cyc}(\xi)$ is the number of cycles in $G_\xi(m)$. Now define the function

$$f(m) := \sum_{\sigma\in\mathrm{PMP}_{2m}} 2^{\mathrm{cyc}(\xi)} \tag{A4}$$

so that

$$\mathbb{E}_\mathcal{G}[|\mathrm{Haf}(X)|^4] = (2m-1)!!\,f(m). \tag{A5}$$

Our goal is to derive a recursion relation for $f(m)$. Focus on the nodes (1,2). There are two possibilities for the action of ξ. One case is when (12) appears in ξ. Call all such permutations ξ_{12}. For these permutations, the nodes (1,2) already form a cycle in $G_\xi(m)$ and so it holds that

$$\begin{aligned}\sum_{\xi_{12}\in\mathrm{PMP}_{2m}} 2^{\mathrm{cyc}(\xi)} &= 2\sum_{\xi\in\mathrm{PMP}_{2(m-1)}} 2^{\mathrm{cyc}(\xi)}\\ &= 2f(m-1).\end{aligned}$$

Similarly, for all other permutations ξ with an element of the form $(1j)$, for $j\neq 2$, we can combine the nodes 1, 2, and j into a single node without changing the number of cycles in the graph. This is shown in Fig. 6. Call these permutations ξ_j. Since there are $2m-2$ possible values of j, we have

$$\begin{aligned}\sum_{\xi_j\in\mathrm{PMP}_{2m}} 2^{\mathrm{cyc}(\xi)} &= (2m-2)\sum_{\xi\in\mathrm{PMP}_{2(m-1)}} 2^{\mathrm{cyc}(\xi)}\\ &= (2m-2)f(m-1).\end{aligned}$$

Combining these results, we obtain the desired recursion relation $f(m) = 2mf(m-1)$, which implies $f(m) = (2m)!!$. Setting $k = 2m$, we finally obtain the desired expression for the second moment,

$$\mathbb{E}_\mathcal{G}[|\mathrm{Haf}(X)|^4] = (k-1)!!k!! = k!. \tag{A6}$$

[1] J. Preskill, arXiv:1801.00862.
[2] S. Boixo, S. V. Isakov, V. N. Smelyanskiy, R. Babbush, N. Ding, Z. Jiang, M. J. Bremner J. M. Martinis, and H. Neven, Nat. Phys. **14**, 595 (2018).
[3] S. Aaronson and L. Chen, arXiv:1612.05903.
[4] E. Farhi and A. W. Harrow, arXiv:1602.07674.
[5] M. J. Bremner, A. Montanaro, and D. J. Shepherd, Phys. Rev. Lett. **117**, 080501 (2016).
[6] M. J. Bremner, A. Montanaro, and D. J. Shepherd, Quantum **1**, 8 (2017).
[7] T. Douce, D. Markham, E. Kashefi, E. Diamanti, T. Coudreau, P. Milman, P. van Loock, and G. Ferrini, Phys. Rev. Lett. **118**, 070503 (2017).
[8] J. M. Arrazola, P. Rebentrost, and C. Weedbrook, arXiv:1712.07288.
[9] X. Gao, S.-T. Wang, and L.-M. Duan, Phys. Rev. Lett. **118**, 040502 (2017).
[10] J. Bermejo-Vega, D. Hangleiter, M. Schwarz, R. Raussendorf, and J. Eisert, Phys. Rev. X **8**, 021010 (2018).
[11] A. W. Harrow and A. Montanaro, Nature (London) **549**, 203 (2017).
[12] S. Aaronson and A. Arkhipov, in *Proceedings of the Forty-third Annual ACM Symposium on the Theory of Computing* (ACM, New York, 2011), pp. 333–342.
[13] J. B. Spring, B. J. Metcalf, P. C. Humphreys, W. S. Kolthammer, X.-M. Jin, M. Barbieri, A. Datta, N. Thomas-Peter, N. K. Langford, D. Kundys, J. C. Gates, B. J. Smith, P. G. R. Smith, and I. A. Walmsley, Science **339**, 798 (2012).
[14] M. A. Broome, A. Fedrizzi, S. Rahimi-Keshari, J. Dove, S. Aaronson, T. C. Ralph, and A. G. White, Science **339**, 794 (2013).
[15] M. Tillmann, B. Dakić, R. Heilmann, S. Nolte, A. Szameit, and P. Walther, Nat. Photon. **7**, 540 (2013).
[16] A. Crespi, R. Osellame, R. Ramponi, D. J. Brod, E. F. Galvao, N. Spagnolo, C. Vitelli, E. Maiorino, P. Mataloni, and F. Sciarrino, Nat. Photon. **7**, 545 (2013).
[17] A. P. Lund, A. Laing, S. Rahimi-Keshari, T. Rudolph, J. L. O'Brien, and T. C. Ralph, Phys. Rev. Lett. **113**, 100502 (2014).
[18] M. Bentivegna, N. Spagnolo, C. Vitelli, F. Flamini, N. Viggianiello, L. Latmiral, P. Mataloni, D. J. Brod, E. F. Galvão, A. Crespi *et al.*, Sci. Adv. **1**, e1400255 (2015).
[19] L. Latmiral, N. Spagnolo, and F. Sciarrino, New J. Phys. **18**, 113008 (2016).
[20] C. S. Hamilton, R. Kruse, L. Sansoni, S. Barkhofen, C. Silberhorn, and I. Jex, Phys. Rev. Lett. **119**, 170501 (2017).
[21] R. Kruse, C. S. Hamilton, L. Sansoni, S. Barkhofen, C. Silberhorn, and I. Jex, arXiv:1801.07488.
[22] J. Huh, G. G. Guerreschi, B. Peropadre, J. R. McClean, and A. Aspuru-Guzik, Nat. Photon. **9**, 615 (2015).
[23] W. R. Clements, J. J. Renema, A. Eckstein, A. A. Valido, A. Lita, T. Gerrits, S. W. Nam, W. S. Kolthammer, J. Huh, and I. A. Walmsley, arXiv:1710.08655.
[24] C. Sparrow, E. Martín-López, N. Maraviglia, A. Neville, C. Harrold, J. Carolan, Y. N. Joglekar, T. Hashimoto, N. Matsuda, J. L. O'Brien *et al.*, Nature (London) **557**, 660 (2018).
[25] J. M. Arrazola and T. R. Bromley, Phys. Rev. Lett. **121**, 030503 (2018).
[26] R. Y. Rubinstein and D. P. Kroese, *Simulation and the Monte Carlo Method* (Wiley, Hoboken, NJ, 2016), Vol. 10.
[27] I. M. Bomze, M. Budinich, P. M. Pardalos, and M. Pelillo, in *Handbook of Combinatorial Optimization* (Springer, Boston, 1999), pp. 1–74.
[28] U. Feige, D. Peleg, and G. Kortsarz, Algorithmica **29**, 410 (2001).
[29] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins, Comput. Netw. **31**, 1481 (1999).
[30] A. Angel, N. Sarkas, N. Koudas, and D. Srivastava, Proc. VLDB Endow. **5**, 574 (2012).
[31] A. Beutel, W. Xu, V. Guruswami, C. Palow, and C. Faloutsos, in *Proceedings of the 22nd International Conference on World Wide Web* (ACM, New York, 2013), pp. 119–130.
[32] J. Chen and Y. Saad, IEEE Trans. Knowl. Data Eng. **24**, 1216 (2012).
[33] E. Fratkin, B. T. Naughton, D. L. Brutlag, and S. Batzoglou, Bioinformatics **22**, e150 (2006).
[34] B. Saha, A. Hoch, S. Khuller, L. Raschid, and X.-N. Zhang, in *Annual International Conference on Research in Computational Molecular Biology* (Springer, Berlin, 2010), pp. 456–472.
[35] S. Arora, B. Barak, M. Brunnermeier, and R. Ge, Commun. ACM **54**, 101 (2011).
[36] P. van Laarhoven and E. Aarts, *Simulated Annealing: Theory and Applications* (Springer, Berlin, 1987), Vol. 37.
[37] See supplemental material at http://link.aps.org/supplemental/10.1103/PhysRevA.98.012322, where B is supplied as a CSV file. The submatrix solving Max-Haf is given by the entries (4,6,7,8,13,17,20,24,25,26) of B.
[38] A. Björklund, B. Gupt, and N. Quesada, arXiv:1805.12498.
[39] J. Mockus, *Bayesian Approach to Global Optimization: Theory and Applications* (Springer Science & Business Media, Berlin, 2012), Vol. 37.