Enhancing Combinatorial Optimization with Quantum Generative Models

Javier Alcazar\textsuperscript{1} and Alejandro Perdomo-Ortiz\textsuperscript{1,\#}

\textsuperscript{1}Zapata Computing Canada Inc., 325 Front St W, Toronto, ON, M5V 2Y1

(Dated: January 18, 2021)

Combinatorial optimization is one of the key candidates in the race for practical quantum advantage. In this work, we introduce a new family of quantum-enhanced optimizers and demonstrate how quantum machine learning models known as quantum generative models can find lower minima than those found by means of stand-alone state-of-the-art classical solvers. We present two new quantum-enhanced optimization strategies. The first scheme leverages data points evaluated during the optimization search from any quantum or classical optimizer. In this scheme, we show how our quantum generative model boosts the performance of classical solvers in hard-to-solve instances where the classical solver is not capable of making progress as a stand-alone strategy. The second quantum optimization strategy works as a stand-alone solver. Here we show its superior performance when the goal is to find the best minimum within the least number of cost function evaluations. Under this setting, we benchmark our quantum-enhanced optimization strategies against several solvers, including Bayesian optimizers which are known to be one of the best competing solvers in such tasks. To illustrate our findings, these benchmarks are performed in the context of the portfolio optimization problem by constructing instances from the S&P 500 stock market index. We show that our quantum-inspired generative models based on tensor networks generalize to unseen candidates with lower cost function values than any of the candidates seen by the classical solvers. This is the first demonstration of the generalization capabilities of quantum generative models that brings real value in the context of an industrial-scale application.

I. INTRODUCTION

Along with machine learning and the simulation of materials, combinatorial optimization is one of top candidates for practical quantum advantage. That is, the moment where a quantum-assisted algorithm outperforms the best classical algorithms in the context of a real-world application with a commercial or scientific value. There is an ongoing portfolio of techniques to tackle optimization problems with quantum subroutines, ranging from algorithms tailored for quantum annealers (e.g., Refs. \cite{1,2}), gate-based quantum computers (e.g., Refs. \cite{3,4}) and quantum-inspired (QI) models based on tensor networks (e.g., Ref. \cite{5}).

Regardless of the quantum optimization approach proposed to date, there is a need to translate the real-world problem into a polynomial unconstrained binary optimization (PUBO) expression – a task which is not necessarily straightforward and that usually results in an overhead in terms of the number of variables. Specific real-world use cases illustrating these PUBO mappings are depicted in Refs. \cite{6} and \cite{7}. Therefore, to achieve practical quantum advantage in the near-term, it would be ideal to find a quantum optimization strategy that can work on arbitrary objective functions, bypassing the translation and overhead limitations raised here.

In our work, we offer a solution to these challenges by proposing a family of quantum enhanced optimizers (QEOs) which can scale to large problems where combinatorial problems become intractable in real-world settings. Since our solver does not rely on the details of the objective function to be minimized it is categorized in the family of the so-called black-box solvers. Another highlight of our approach is that it can utilize available observations obtained from attempts to solve the optimization problem. These initial evaluations can come from any source, from random search trials to tailored state-of-the-art classical (or quantum) optimizers for the specific problem at hand.

Our QEO strategy is based on two key ideas. First, our model relies on a probabilistic component which aims to capture the correlations in previously observed data (step 0-3 in Fig. \ref{fig:proposal}). In the proposal presented here, our QEOs leverage the probabilistic modeling framework of generative models. Second, the (quantum) generative models need to be capable of generating new “unseen” solution candidates which have the potential to have a lower value for the objective function than those already “seen” and used as the training set (step 4-6 in Fig. \ref{fig:proposal}). This is the fundamental concept of generalization: the most desirable and important feature of any practical ML model. We will elaborate next on each of these components and demonstrate these two properties in the context of the tensor-network-based generative models and its application to a non-deterministic polynomial-time hard (NP-hard) version of the portfolio optimization in finance.

II. QUANTUM-ENHANCED OPTIMIZATION WITH GENERATIVE MODELS

As shown in Fig. \ref{fig:proposal} depending on the specifics we can construct an entire family of solvers whose generative modeling core range from classical, QI or quantum circuit (QC) enhanced, or hybrid quantum-classical model. These options can be realized by utilizing, for example, Boltzmann machines \cite{8} or Generative Adversarial Networks (GAN) \cite{9}. Tensor-Network Born Machines (TNBM) \cite{10}, Quantum Circuit Born Machines (QCBM)\cite{11} or Quantum-Circuit Associative Adversarial Networks (QC-AAN)\cite{12} respectively, to name just a few of the many options for this probabilistic component.

\#alejandro@zapatacomputing.com
QI algorithms come as an interesting alternative since these allow one to simulate larger scale quantum systems with the help of efficient tensor-network (TN) representations. Depending on the complexity of the TN used to build the quantum generative model, one can simulate from thousands of problem variables to a few tens, the latter being the limit of simulating an universal gate-based quantum computing model. This is, one can control the amount of quantum resources available in the quantum generative model by choosing the QI model.

Therefore, from all quantum generative model options, we chose to use a QI generative model based on TNs to test and scale our QEO strategy to instances with a number of variables commensurate with those found in industrial-scale scenarios. We refer to our solver hereafter as TN-QEO. For the training of our TN-QEO models we followed the work of Han et al. [13] where they proposed to use Matrix Product States (MPS) to build the unsupervised generative model. This work extends the scope from early successes of these quantum-inspired models in the context of supervised ML [14–17].

In this paper we will discuss two modes of operation for our family of quantum-enhanced solvers:

- In **TN-QEO as a “booster”** we leverage past observations from classical (or quantum) solvers. To illustrate this mode we use observations from simulated annealing (SA) runs. Simulation details are provided in Appendix A 4.

- In **TN-QEO as a stand-alone solver** all initial cost function evaluations are decided entirely by the quantum-inspired generative model, and a random prior is constructed just to give support to the target probability distribution the MPS model is aiming to capture. Simulation details are provided in Appendix A 5.

Both of these strategies are captured in the algorithm workflow diagram in Fig. 1 and described in more detail in Appendix A .

## III. RESULTS AND DISCUSSION

To illustrate the implementation for both of these settings we tested their performance on an NP-hard version of the portfolio optimization problem with cardinality constraints. The
selection of optimal investment on a specific set of assets, or portfolios, is a problem of great interest in the area of quantitative finance. This problem is of practical importance for investors, whose objective is to allocate capital optimally among assets while respecting some investment restrictions. The goal of this optimization task, introduced by Markowitz [18], is to generate a set of portfolios that offers either the highest expected return (profit) for a defined level of risk or the lowest risk for a given level of expected return. In this work, we focus in the combinatorial optimization problem of choosing portfolios which minimizes its volatility or risk given a specific target return. More details are provided in Appendix A.1

A. TN-QEO as a booster for any other combinatorial optimization solver

In Fig. 2 we present the experimental design and the results obtained from using TN-QEO as a booster. In these experiments we illustrate how using intermediate results from simulated annealing (SA) can be used as seed data for our TN-QEO algorithm. As described in Fig. 2 there are two strategies we explored (strategies 1 and 2) to compare with our TN-QEO strategy (strategy 4). To fairly compare each strategy, we provide each with approximately the same computational wall-clock time. For strategy 2, this translates into performing additional restarts of SA with the time allotted for TN-QEO. In the case of strategy 1, where we explored different settings for SA from the start compared to those used in strategy 2, this amounts to using the same total number of number of cost functions evaluations as those allocated to SA in strategy 2. For our experiments this number was set to 20,000 cost function evaluations for strategies 1 and 2. In strategy 4, the TN-QEO was initialized with a prior consisting of the best 1,000 observations out of the first 10,000 coming from strategy 2 (see Appendix A.4 for details). To evaluate the performance enhancement obtained from the TN-QEO strategy we compute the relative quantum enhancement \( \eta \), which we define as

\[
\eta = \frac{C_{\text{cl} \text{ min}} - C_{\text{QEO} \text{ min}}}{C_{\text{cl} \text{ min}}} \times 100\%. \tag{1}
\]

Here, \( C_{\text{cl} \text{ min}} \) is the lowest minimum value found by the classical strategy (e.g., strategies 1-3) while \( C_{\text{QEO} \text{ min}} \) corresponds to the lowest value found with the quantum-enhanced approach (e.g., with TN-QEO). Therefore, positive values reflect an improvement over the only-classical approaches, while negative values indicate cases where the classical solvers outperform

the quantum-enhanced proposal.

As shown in the Fig. 2, we observe that TN-QEO outperforms on average both of the only-classical strategies implemented. The quantum enhancement observed here, as well as the trend for a larger enhancement as the number of variables (assets) becomes larger, is confirmed in many other investment universes with a number of variables ranging from \( N = 30 \) to \( N = 100 \) (see Appendix [B] for more details). Although we show an enhancement compared to SA, similar results could be expected when other solvers are used, since our approach builds on solutions found by the solver and does not compete with it from the start of the search. Furthermore, the more data available, the better the expected performance of TN-QEO is. An important highlight of TN-QEO as a booster is that these previous observations can come from a combination of solvers, as different as purely quantum or classical, or...
FIG. 3. Generalization capabilities of our quantum-inspired generative model. Left panel corresponds to an investment universe with \( N = 50 \) assets while the right panel corresponds to one with \( N = 100 \) assets. The blue histogram represents the number of observations or portfolios obtained from the classical solver (seed data set). In orange we represent samples coming from our quantum generative model at the core of TN-QEO. The green dash line is positioned at the best risk value found in the seed data. This mark emphasizes all the new outstanding samples obtained with the quantum generative model and which correspond to lower portfolio risk value (better minima) than those available from the classical solver by itself. The number of outstanding samples in the case of \( N = 50 \) is equal to 31, while 349 outstanding samples were obtained from the MPS generative model in the case of \( N = 100 \).

The observed performance enhancement compared with the classical-only strategy must be coming from a better exploration of the relevant search space, i.e., the space of those bitstring configurations \( x \) representing portfolios which could yield a low risk value for a specified expected investment return. That is the intuition behind the construction of TN-QEO. The goal of the generative model is to capture the important correlations in the previously observed data, and to use its generative capabilities to propose similar new candidates.

Generating new candidates is by no means a trivial task in ML and it determines the usefulness and power of the model since it measures its generalization capabilities. In this setting of QI generative models, one expects that the MPS-based generative model at the core of TN-QEO is not simply memorizing the observations given as part of the training set, but that it will provide new unseen candidates. This is an idea which has been recently tested and demonstrated to some extent on synthetic data sets (see e.g., Refs. [19] and [20]). In Fig. 3 we demonstrate that our quantum-inspired generative model is generalizing to new samples and that these add real value to the optimization search. To the best of our knowledge this is the first demonstration of the generalization capabilities of quantum generative models in the context of a real-world application in an industrial scale setting, and one of our main findings in our paper. We provide more examples of this observation in Fig. 5.

Note that our TN-based generative model not only produces better minima than the classical seed data, but it also generates a rich amount of samples in the low cost spectrum. This bias is imprinted in the design of our TN-QEO and it is the purpose of the softmax surrogate prior distribution shown in Fig. 1.

This richness of new samples could be useful not only for the next iteration of the algorithm, but they may also be readily of value to the user solving the application. In some applications there is value as well in having information about the runners-up. Ultimately, the cost function is just a model of the system guiding the search, and the lowest cost does not translate to the best performance in the real-life investment strategy.

B. Quantum-Enhanced Optimization as a Stand-Alone Solver

Next, we explore the performance of our TN-QEO framework as a stand-alone solver. The focus is in combinatorial problems whose cost functions are expensive to evaluate and where finding the best minimum within the least number of calls to this function is desired. In Fig. 4 we present the comparison against three different classical optimization strategies. As the first solver, we use the conditioned random solver which is a more sophisticated random strategy compared to the fully random search over the \( 2^N \) bitstrings of all possible portfolios, where \( N \) is the number of assets in our investment universe. The conditioned random strategy uses \( a \) priori information that the search is restricted to bitstrings containing a fixed number of \( \kappa \) assets. Therefore the number of combinatorial possibilities is \( M = \binom{N}{\kappa} \), which is significantly less than \( 2^N \). As expected, when this information is not used the performance of the random solver over the entire \( 2^N \) search space is significantly worse (data not shown). The other two competing strategies considered here are SA and the Bayesian optimization library GPyOpt [21]. In both...
FIG. 4. TN-QEO as a stand-alone solver: In this comparison of TN-QEO against three classical competing strategies, investment universes are constructed from subsets of the S&P 500 with a diversity in the number of assets (problem variables) ranging from \( N = 30 \) to \( N = 100 \).

The goal is to minimize the risk given an expected return which is one of the specifications in the combinatorial problem addressed here. Error bars and their 95% confidence intervals are calculated from bootstrapping over 100 independent random initializations for each solver on each problem. The main line for each solver corresponds to the bootstrapped median over these 100 repetitions, demonstrating the superior performance of TN-QEO over the classical solvers considered here. As specified in the text, with the exception of TN-QEO, the classical solvers use to their advantage the \textit{a priori} information coming from the cardinality constraint imposed in the selection of valid portfolios.

Of these classical solvers, we adapted their search strategy to impose this cardinality constraint with fixed \( \kappa \) as well (details in Appendix A.3). This raises the bar even higher for TN-QEO which is not using that \textit{a priori} information to boost its performance [22]. As explained in Appendix A.5 we only use this information indirectly during the construction of the artificial seed data set which initializes the algorithm (step 0, Fig. 1), but it is not a strong constraint during the construction of the QI generative model (step 3, Fig. 1) or imposed to generate the new candidate samples coming from it (step 4, Fig. 1). Post selection can be applied \textit{a posteriori} such that only samples with the right cardinality are considered as valid candidates towards the selected set (step 5, Fig. 1).

In Fig. 4 we demonstrate the advantage of our TN-QEO stand-alone strategy compared to any of these widely-used solvers. In particular, it is interesting to note that the gap between TN-QEO and the other solvers seems to be larger for larger number of variables.

IV. OUTLOOK

Although we limited the scope of this work to tensor-network based quantum generative models, one natural extension would be to consider other quantum generative models. For example, hybrid quantum-classical models such as the quantum circuit Associative Adversarial Networks (QC-AAN) [12] can be readily explored to leverage the power of quantum generative models with so called noisy intermediate-scale quantum (NISQ) devices [23]. Both quantum-inspired and hybrid quantum-classical algorithms can be tested within this QEO framework in even larger problem sizes of this NP-hard version of the portfolio optimization problem, or any other combinatorial optimization problem. As NISQ devices increase their qubit count, it would be interesting to explore generative models which are capable of leveraging more quantum resources, such as the Quantum Circuit Born Machines (QCBM) [11]: a general framework to model arbitrary probability distributions and to perform generative modeling tasks with gate-based quantum computers.

As presented in Fig. 1 benchmarking any of the QEO strategies against classical generative models is another promising possibility to explore. Increasing the expressibility power of the quantum-inspired core from MPS to other more complex but still efficient QI approaches, such as tree tensor networks [24], is another interesting research direction. Although we proved the relevancy and scalability of our algorithm towards full-scale industrial applications by boosting
the performance of classical solvers in full industrial-scale instances (all 500 assets in the S&P 500 market index), it is to be explored the performance enhancement that could be obtained from more complex TN representations or in other combinatorial problems.

Compared to other quantum optimization strategies, one important feature of TN-QEO is its algorithmic flexibility. As demonstrated here and in contrast to other proposals, our QEO framework can be applied to arbitrary cost functions, opening the possibility to new applications which are not readily amenable to an explicit mapping to a polynomial unconstrained binary optimization (PUBO) problem. Our approach is also flexible in terms of the source of the seed samples since these can come from any solver. Therefore, although we showed an enhancement compared to SA, it is expected that TN-QEO can learn from available cost function evaluations from any other potentially more efficient, or even application-tailored optimizers. The demonstrated generalization capabilities of the generative model at its core helps TN-QEO to build on any progress from previous attempts with other state-of-the-art solvers, and it returns new candidates which might not be reachable by the classical optimizer on its own. We are optimistic that this flexible approach will open up the wide applicability of quantum and quantum-inspired generative models in industrial-scale real-world combinatorial optimization problems.

ACKNOWLEDGMENTS

The authors would like to acknowledge Manuel S. Rudolph, Marta Mauri, Matthew J.S. Beach, Yudong Cao, Luis Serrano, Jhonathan Romero-Fontalvo, and Brian Dellabetta for their feedback on an early version of this manuscript.

[1] Tadashi Kadowaki and Hidetoshi Nishimori, “Quantum annealing in the transverse ising model,” Phys. Rev. E. 58, 5355 (1998).
[2] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, Joshua Lapan, Andrew Lundgren, and Daniel Preda, “A quantum adiabatic evolution algorithm applied to random instances of an NP-Complete problem,” Science 292, 472–475 (2001).
[3] Sam Gutmann Edward Farhi, Jeffrey Goldstone, “A quantum approximate optimization algorithm,” arXiv:1411.4028 (2014).
[4] Stuart Hadfield, Zhihui Wang, Bryan O’Gorman, Eleanor G Rieffel, Davide Venturelli, and Rupak Biswas, “From the quantum approximate optimization algorithm to a quantum alternating operator ansatz,” Algorithms 12, 34 (2019).
[5] Samuel Mugel, Carlos Kuchkovsky, Escolastico Sanchez, Samuel Fernandez-Lorenzo, Jorge Luis-Hita, Enrique Lizaso, and Roman Orus, “Dynamic portfolio optimization with real datasets using quantum processors and quantum-inspired tensor networks,” (2020), arXiv:2007.00017 [quant-ph].
[6] A. Perdomo-Ortiz, N. Dickson, M. Drew-Brook, G. Rose, and A. Aspuru-Guzik, “Finding low-energy conformations of lattice protein models by quantum annealing,” Sci. Rep. 2, 571 (2012).
[7] Alejandro Perdomo-Ortiz, Alexander Feldman, Asier Ozaeta, Sergej V. Isakov, Zheng Zhu, Bryan O’Gorman, Helmut G. Katzgraber, Alexander Diedrich, Hartmut Neven, Johan de Kleer, Brad Lackey, and Rupak Biswas, “Readiness of quantum optimization machines for industrial applications,” Phys. Rev. Applied 12, 014004 (2019).
[8] Song Cheng, Jing Chen, and Lei Wang, “Information perspective to probabilistic modeling: Boltzmann machines versus born machines,” Entropy 20, 583 (2018).
[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, “Generative adversarial nets,” in Advances in Neural Information Processing Systems 27 edited by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (Curran Associates, Inc., 2014) pp. 2672–2680.
[10] Song Cheng, Jing Chen, and Lei Wang, “Information perspective to probabilistic modeling: Boltzmann machines versus Born machines,” Entropy 20 (2017).
[11] Marcello Benedetti, Delfina Garcia-Pintos, Oscar Perdomo, Vincente Leyton-Ortega, Yuneong Nam, and Alejandro Perdomo-Ortiz, “A generative modeling approach for benchmarking and training shallow quantum circuits,” npj Quantum Information 5, 45 (2019).
[12] Manuel S. Rudolph, Ntwali Toussaint Bashige, Amara Katabarwa, Sonika Johr, Borja Peropadre, and Alejandro Perdomo-Ortiz, “Generation of high resolution handwritten digits with an ion-trap quantum computer,” (2020), arXiv:arXiv:2012.03924 [quant-ph].
[13] Zhao-Yu Han, Jun Wang, Heng Fan, Lei Wang, and Pan Zhang, “Unsupervised generative modeling using matrix product states,” Physical Review X 8 (2018), 10.1103/physrevx.8.031012.
[14] Edwin Stoudenmire and David J Schwab, “Supervised learning with tensor networks,” in Advances in Neural Information Processing Systems 29, edited by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Curran Associates, Inc., 2016) pp. 4799–4807.
[15] Stavros Elthymiou, Jack Hidary, and Stefan Leichenauer, “TensorNetwork for machine learning,” (2019), arXiv:1906.06329 [cs.LG].
[16] Chase Roberts, Ashley Milsted, Martin Ganahl, Adam Zalcman, Bruce Fontaine, Yijian Zou, Jack Hidary, Guifre Vidal, and Stefan Leichenauer, “TensorNetwork: A library for physics and machine learning,” (2019), arXiv:1905.01330 [physics.comp-ph].
[17] Matthew Fishman, Steven R. White, and E. Miles Stoudenmire, “The ITensor software library for tensor network calculations,” (2020), arXiv:2007.14824 [cs.MS].
[18] Harry Markowitz, “Portfolio selection,” The Journal of Finance 7, 77–91 (1952).
[19] Tai-Danae Bradley, E M Stoudenmire, and John Terilla, “Modeling sequences with quantum states: a look under the hood,” Machine Learning: Science and Technology 1, 035008 (2020).
[20] James Stokes and John Terilla, “Probabilistic modeling with matrix product states,” Entropy 21 (2019).
[21] The GPyOpt authors, “Gpyopt: A bayesian optimization framework in python,” http://github.com/SheffieldML/GPyOpt (2016).
[22] Specific adaptations of the MPS generative model could be implemented such that it conserves the number of assets by construc-
tion, borrowing ideas from condensed matter physics where one can impose MPS a conservation in the number of particles in the quantum state.

[23] John Preskill, “Quantum computing in the NISQ era and beyond,” Quantum 2, 79 (2018)

[24] Song Cheng, Lei Wang, Tao Xiang, and Pan Zhang, “Tree tensor networks for generative modeling,” Phys. Rev. B 99, 155131 (2019)

[25] Joachim Dahl Martin Andersen and Lieven Vandenberghe, “Python software for convex optimization,” http://cvxopt.org (2020).

[26] Ignacio Cirac, David Perez-Garcia, Norbert Schuch, and Frank Verstraete, “Matrix product states and projected entangled pair states: Concepts, symmetries, and theorems,” (2020), arXiv:2011.12127 [quant-ph].

[27] “Code for unsupervised generative modeling using matrix product states,” https://github.com/congzlwag/UnsupGenModbyMPS (2018).

[28] Matthew T. Perry and Richard J. Wagner, “Python module for simulated annealing,” https://github.com/perrygeo/simanneal (2019).

[29] Javier Alcazar, Vicente Leyton-Ortega, and Alejandro Perdomo-Ortiz, “Classical versus quantum models in machine learning: insights from a finance application,” Machine Learning: Science and Technology 1, 035003 (2020).

Appendix A: Methods

1. Generation of portfolio optimization instances

The portfolio optimization problem aims at determining the fractions \( w_i \) of a given capital to be invested in each asset \( i \) of a universe of \( N \) assets, such that the risk \( \sigma(w) \) is minimized, constrained to the fact sum \( \sum_i w_i = 1 \). The problem can be formulated as:

\[
\min_{w} \{ \sigma^2(w) : w^T \Sigma \cdot w : \langle r(w) \rangle = w \cdot \rho = \rho \} \quad (A1)
\]

where the vectors \( w \) and \( r \) have dimensionality \( N \), \( \Sigma \) is the sample covariance matrix obtained from the return time series of pair of asset \( i \) and \( j \), and \( \rho \) is the vector of average return of the time series for each asset, with each daily return, \( r^i \), calculated as the relative increment in asset price from its previous day (i.e., \( r^i = (p^i - p^{(t-1)})/p^{(t-1)} \), with \( p^i \) as the price for a particular asset at time \( t \)). The solution to Eq. (A1) for a given return level \( \rho \) corresponds to the optimal portfolio strategy \( w^\ast \) and the minimal value of this objective function \( \sigma(w) \) correspond to the portfolio risk and will be denoted by \( \sigma^\ast_\rho \).

Note that the optimization task in Eq. (A1) has the potential outcome of investing small amounts in a large number of assets as an attempt to reduce the overall risk by “over diversifying” the portfolio. This type of investment strategy can be challenging to implement in practice: portfolios composed of a large number of assets are difficult to manage and may incur in high transaction costs. Therefore, several restrictions are usually imposed on the allocation of capital among assets, as a consequence of market rules and conditions for investment or to reflect investor profiles and preferences. For instance, constraints can be included to control the amount of desired diversification, i.e., modifying bound limits per asset \( i \), denoted by \{\( l_i, u_i \)\}, to the proportion of capital invested in the investment on individual assets or a group of assets, thus the constraint \( l_i < w_i < u_i \) could be considered.

Additionally, a more realistic and common scenario is to include in the optimization task a cardinality constraint, which limits directly the number of assets to be transacted to a pre-specified number \( \kappa < N \). Therefore, the number of different sets to be treated is \( M = \binom{N}{\kappa} \). In this scenario, the problem can be formulated as a Mixed-Integer Quadratic Program (MIQP) with the addition of binary variables \( x_i \in \{0,1\} \) per asset, for \( i = 1,...,N \), which are set to “1” when the \( i \)-th asset is included as part of the \( \kappa \) assets, or “0” if it is left out of this selected set. Therefore, valid portfolios would have a number \( \kappa \) of 1’s, as specified in the cardinality constraint. For example, for \( N = 4 \) and \( \kappa = 2 \), the six different valid configurations can be encoded as \{0011, 0101, 0110, 1001, 1010, 1100\}.

The optimization task can then be described as follows:

\[
\begin{align*}
\min_{w, x} \{ & \sigma^2(w) : \\
& \langle r(w) \rangle = \rho, \\
& l_i x_i < w_i < u_i x_i \quad i = 1,...,N, \\
& \sum_i x_i = \kappa \}.
\end{align*}
\]

(A2)

In this reformulated problem we denote by \( \sigma^\ast_{\rho, \kappa} \) the minimum portfolio risk outcome from Eq. (A2) for a given return level \( \rho \) and cardinality \( \kappa \). The optimal solution vectors \( w^\ast \) and \( x^\ast \) define the portfolio investment strategy. Adding the cardinality constrain and the investment bound limits transforms a simple convex optimization problem (Eq. [A1]) into a much harder non-convex NP-hard problem . For all the problem instance generation in this work we chose \( \kappa = N/2 \) and the combinatorial nature of the problems lies in the growth of the search space associated with the binary vector \( x \), which makes it intractable to exhaustively explore for a number of assets in the few hundreds. The size of the search space here is \( M = \binom{N}{N/2} \).

It is important to note that given a selection of which assets belong to the portfolio by instantiating \( x \) (say with a specific \( x^{(1)} \)), solving the optimization problem in Eq. (A2) to find the respective investment fractions \( w^{(1)} \) and risk value \( \sigma^\ast_{\rho,N/2} \) can be efficiently achieved with conventional quadratic programming (QP) solvers. In this work we used the python module cvxopt [25] for solving this problem. Note that we exploit this fact to break this constrained portfolio optimization problem into a combinatorial intractable one (find best asset selection \( x \)), which we aim to solve with QEO, and a tractable subroutine which can be solved efficiently with available solvers.

The set of pairwise \( (\sigma^\ast_{\rho, \kappa}, \rho) \), dubbed as the efficient frontier, is no longer convex neither continuous in contrast with the solution to problem in Eq. (A1).
2. Quantum-Inspired Generative Model in TN-QEO

The addition of a probabilistic component is inspired by the success of Bayesian Optimization (BO) techniques, which are among the most efficient solvers when the performance metric aims to find the lowest minimum possible within the least number of objective function evaluations. For example, within the family of BO solvers, GPyOpt [21] uses a Gaussian Process (GP) framework consisting of multivariate Gaussian distributions. This probabilistic framework aims to capture relationships among the previously observed data points (e.g., through tailored kernels), and it guides the decision of where to sample the next evaluation with the help of the so-called acquisition function. GPyOpt is one of the solvers we use to benchmark the new quantum-enhanced strategies proposed here.

Although the GP framework in BO techniques is not a generative model, we explore here the powerful unsupervised machine learning framework of generative modeling in order to capture correlations from an initial set of observations and evaluations of the objective function (step 1-4 in Fig. 1).

For the implementation of the quantum-inspired generative model at the core of TN-QEO we follow the procedure proposed and implemented in Ref. [13]. Inspired by the probabilistic interpretation of quantum physics via Born’s rule, it was proposed that one can use the Born probabilities \( |\Psi(x)|^2 \) over the \( 2^N \) states of an \( N \) qubit system to represent classical target probability distributions which would be obtained otherwise with generative machine learning models. Hence,

\[
P(x) = \frac{|\Psi(x)|^2}{Z}, \quad \text{with} \quad Z = \sum_{x \in S} |\Psi(x)|^2, \tag{A3}
\]

with \( \Psi(x) = \langle x | \Psi \rangle \) and \( x \in \{0, 1\}^N \) are in one-to-one correspondence with decision variables over the investment universe with \( N \) assets in our combinatorial problem of interest here. In Ref. [13] these quantum-inspired generative models were named as Born machines, but we will refer to them hereafter as tensor-network Born machines (TNBM) to differentiate it from the quantum circuit Born machines (QCBM) proposal [11] which was developed independently to achieve the same purpose but by leveraging quantum wave functions from quantum circuits in NISQ devices. As explained in the main text, either quantum generative model can be adapted for the purpose of our QEO algorithm.

On the grounds of computational efficiency and scalability towards problem instances with large number of variables (in the order of hundreds or more), following Ref. [13] we implemented the quantum-inspired generative model based on Matrix Product States (MPS) to learn the target distributions \( |\Psi(x)|^2 \).

MPS is a type of TN where the tensors are arranged in a one-dimensional geometry. Despite its simple structure, MPS can efficiently represent a large number of quantum states of interest extremely well [26]. Learning with the MPS is achieved by adjusting its parameters such that the distribution obtained via Born’s rule is as close as possible to the data distribution. MPS enjoys a direct sampling method that is more efficient than other Machine Learning techniques, for instance, Boltzmann machines, which require Markov chain Monte Carlo (MCMC) process for data generation.

The key idea of the method to train the MPS, following the algorithm on paper [13], consists of adjusting the value of the tensors composing the MPS as well as the bond dimension among them, via the minimization of the negative log-likelihood function defined over the training dataset sampled from the target distribution. For more details on the implementation see Ref. [13] and for the respective code see Ref. [27].

3. Classical Optimizers

a. GPyOpt Solver

GPyOpt [21] is a Python open-source library for Bayesian Optimization based on GP and a Python framework for Gaussian process modelling. For the comparison exercise in TN-QEO as a stand-alone solver here are the hyperparameters we used for the GPyOpt solver:

- Domain: to deal with the exponential growth in dimensionality, the variable space for \( n \) number of assets was partitioned as the cartesian product of \( n \) 1-dimensional spaces.
- Constraints: we added two inequalities in the number of assets in a portfolio solution to represent the cardinality condition.
- Number of initial data points: 10
- Acquisition function: Expected Improvement

b. Simulated Annealing Solver

For simulated annealing (SA) we implemented a modified version from Ref. [28]. The main change consists of adapting the update rule such that new candidates are within the valid search space with fixed cardinality. The conventional update rule of single bit flips will change the Hamming weight of \( x \) which translates in a portfolio with different cardinality. The hyperparameters used are the following:

- Max temperature in thermalization: 1.0
- Min temperature in thermalization: 1e-4

c. Conditioned Random Solver

This solver corresponds to the simplest and most naive approach, while still using the cardinality information of the problem. In the conditioned random solver, we generate, by contraction, bitstrings which satisfy the cardinality constrain. Given the desired cardinality \( \kappa = N/2 \) used here, one starts
from the bitstring with all zeros, $x_0 = 0 \cdots 0$, and flips only $N/2$ bits at random from positions containing 0’s, resulting in a valid portfolio candidate $x$ with cardinality $N/2$.

4. Algorithm Methodology for TN-QEO as a booster

As explained in the main text, in this case it is assumed that the cost of evaluating the objective function is not the major computational bottleneck, and consequently there is no practical limitations in the number of observations to be considered.

Following the algorithmic scheme in Fig. 1, we describe next the details for each of the steps in our comparison benchmarks:

0 Build the seed data set, $\{x^{(i)}\}_{\text{seed}}$ and $\{\sigma^{(i)}_{\rho,N/2}\}_{\text{seed}}$.

For each problem instance defined by $\rho$ and a random subset with $N$ assets from the S&P 500, gather all initial available data obtained from previous optimization attempts with classical solver(s). In our case, for each problem instances we collected 10,000 observations from the SA solver. These 10,000 observations corresponding to portfolio candidates $\{x^{(i)}\}_{\text{init}}$ and their respective risk evaluations $\{\sigma^{(i)}_{\rho,N/2}\}_{\text{init}}$ were sorted and only the first $n_{\text{seed}} = 1,000$ portfolio candidates with the lowest risks were selected as the seed data set. This seed data set is the one labeled as $\{x^{(i)}\}_{\text{seed}}$ and $\{\sigma^{(i)}_{\rho,N/2}\}_{\text{seed}}$ in the main text and hereafter. The idea of selecting a percentile of the original data is to provide the generative model inside QEO with samples which are the target samples to be generated. This percentile is a hyperparameter and we set it 10% of the initial data for our purposes.

1 Construct of the softmax surrogate distribution: Using the seed data from step 0, we construct a softmax multinomial distribution with $n_{\text{seed}}$ classes - one for each point on the seed data set. The probabilities outcome associated with each of these classes in the multinomial is calculated as a Boltzmann weight, $p_i = \frac{e^{-\sigma^{(i)}_{\rho,N/2}}}{\sum_{j=1}^{n_{\text{seed}}} e^{-\sigma^{(j)}_{\rho,N/2}}}$. Here, $\sigma^{(i)}_{\rho,N/2} = \sigma_{\rho,N/2}(x^{(i)})/T$, and $T$ is a “temperature” hyperparameter. In our simulations, $T$ was computed as the standard deviation of the risk values of this seed data set. In Bayesian optimization methods the surrogate function tracks the landscape associated with the values of the objective function (risk values here). This softmax surrogate constructed here by design as a multinomial distribution from the seed data observations serves the purpose of representing the objective function landscape but in probability space. That is, it will assign higher probability to portfolio candidates with lower risk values. Since we will use this softmax surrogate to generate the training data set, this bias imprints a preference in the quantum-inspired generative model to favor low-cost configurations.

2 Sample from softmax surrogate. We will refer to these samples as the training set since these will be used to train the MPS-based generative model. For our experiments here we used $n_{\text{train}} = 10000$ samples.

3 Use the $n_{\text{train}}$ samples from the previous step to train the MPS generative model.

4 Obtain $n_{\text{MPS}}$ samples from the generative model which correspond to the new list of potential portfolio candidates. In our experiments, $n_{\text{MPS}} = 4000$. For the case of 500 assets, as sampling takes sensibly longer because of the problem dimension, this value was reduced to 400 to match the time in SA.

5 Select new candidates: From the $n_{\text{MPS}}$ samples, select only those who fulfill the cardinality condition, and which have not been evaluated. These new portfolio candidates $\{x^{(i)}\}_{\text{new}}$ are saved for evaluation in the next step.

6 Obtain risk value for new selected samples: Solve Eq. A2 to evaluate the objective function (portfolio risks) for each of the new candidates $\{x^{(i)}\}_{\text{new}}$. We will denote refer to the new cost function values by $\{\sigma^{(i)}_{\rho,N/2}\}_{\text{new}}$.

7 Merge the new portfolios, $\{x^{(i)}\}_{\text{new}}$, and their respective cost function evaluations, $\{\sigma^{(i)}_{\rho,N/2}\}_{\text{new}}$ with the seed portfolios, $\{x^{(i)}\}_{\text{seed}}$, and their respective cost values, $\{\sigma^{(i)}_{\rho,N/2}\}_{\text{seed}}$, from step 0 above. This combined super set is the new initial data set.

8 Use the new initial data set from step 7 to start the algorithm from step 1. If a desired minimum is already found or if no more computational resources are available, one can decide to terminate the algorithm here. In all of our benchmark results reported here when using TN-QEO as a booster from SA intermediate results, we only run the algorithm for this first cycle and the minima reported for the TN-QEO strategy is the lowest minimum obtained up to step 7 above.

5. Algorithm Methodology for TN-QEO as a stand-alone solver

This section presents the algorithm for the TN-QEO scheme as a stand-alone solver. In optimization problems where the objective function is inexpensive to evaluate, we can easily probe it at many points in the search for a minimum. However, if the cost function evaluation is expensive, e.g., tuning hyperparameters of a deep neural network, then it is important to minimize the number of evaluations drawn. This is the domain where optimization technique with a Bayesian flavour, where the search is being conducted based on new information gathered, are most useful, in the attempt to find the global optimum in a minimum number of steps.
The algorithmic steps for TN-QEO as a stand-alone solver follows the same logic as that of the solver as a booster described Sec. [A4]. The main differences between the two algorithms rely on step 0 during the construction of the initial data set and seed data set in step 0, the temperature use in the softmax surrogate in step 1, and a more stringent selection criteria in step 5. Since the other steps remain the same, we focus here to discuss the main changes to the algorithmic details provided in Sec. [A4].

\begin{align*}
\begin{cases}
(n_{\text{seed}} - 1) p_0 + p_1 = 1 \\
p_1 = 2 \cdot p_0
\end{cases} \Rightarrow \begin{cases}
p_0 = 1/(1 + n_{\text{seed}}) \\
p_1 = 2/(1 + n_{\text{seed}})
\end{cases}
\end{align*}

\begin{align*}
\mathcal{N} &= (n_{\text{seed}} - 1) e^{-\sigma(0)/T} + e^{-\sigma(1)/T} \\
p_1 &= e^{-\sigma(1)/T} / \mathcal{N} \\
p'_0 &= e^{-\sigma(0)/T} / \mathcal{N}
\end{align*}

\begin{align*}
\mathcal{N} &= (n_{\text{seed}} + 1) \cdot e^{-\sigma(1)/T} / 2 \\
\sigma(0) &= T \cdot \log 2 + \sigma(1)
\end{align*}

(A4)

It is important to note that $\sigma(0)$ us an artificial reference value which is calculated analytically and does not require a call to the objective function (in contrast to $\sigma(1)$). Here, $\mathcal{N}$ is the normalization factor of the multinomial and $T$ is the temperature hyperparameter which, as in the case of TN-QEO as a booster, can be adjusted later in the algorithm as more data is seen. Due to the lack of initial cost function values, in order to set a relevant typical “energy” scale in this problem, we follow the procedure in Ref. [29] where it is set to be the square root of the mean of the covariance matrix defined in Eq. [A1] as this matrix encapsulates the risk information (volatility) as stated in the Markowitz’s model.

0 Build the seed data set: since evaluating the objective function could be the major bottleneck (assumed to be expensive) then we cannot rely on cost function evaluations to generate the seed data set. The strategy we adopted is to initialize the algorithm with samples of bitstrings which satisfy the hard constraints of the problem. In our specific example, we can easily generate $n_{\text{seed}}$ random samples, $D_0 = \{x^{(i)}\}_{\text{seed}}$, which satisfy the cardinality constrain. Since all the elements in this data set hold the cardinality condition, then maximum length $n_{\text{seed}}$ of $D_0$ is $\binom{N}{\kappa}$. In our experiments, we set the number of samples $n_{\text{init}} = 2,000$, for all problems considered here up to $N = 100$ assets.

1 Construct the softmax surrogate distribution: start by constructing a uniform multinomial probability distribution where each sample in $D_0$ has the same probability. Therefore, for each point in the seed data set its probability is set to $p_0 = 1/n_{\text{seed}}$. As in TN-QEO as a booster, we will attempt to generate a softmax-like surrogate which favors samples with low cost value, but we will slowly build that information as new samples are evaluated. In this first iteration of the algorithm, we start by randomly selecting a point $x^{(1)}$ from $D_0$, and we evaluate the value of its objective function $\sigma^{(1)}$ (its risk value in our specific finance example). To make this point $x^{(1)}$ stand out from the other unevaluated samples, we set its probability to be twice that of any of the remaining $n_{\text{seed}} - 1$ points in $D_0$. Since we increase the probability of one of the points, we need to adjust the probability of the $n_{\text{seed}} - 1$ from $p_0$ to $p'_0$, and if we assume the probability weights for observing each point follows a multinomial distribution with Boltzmann weights, under these assumptions, and making by fixing the temperature hyperparameter we can solve for the reference “risk” value $\sigma(0)$ associated to all the other $n_{\text{seed}} - 1$ points as shown below.

2 Generate training set: same as in TN-QEO as a booster (see Appendix [A4]).

3 Train MPS: same as in TN-QEO as a booster (see Appendix [A4]).

4 Generate samples from trained MPS: same as in TN-QEO as a booster (see Appendix [A4]).

5 Select new candidates from trained MPS: In contrast to TN-QEO as a booster we cannot afford to evaluate all new candidates coming from the MPS samples. In our procedure we selected only two new candidates which must meet the cardinality constrain. For our procedure these two candidates correspond to the most frequent sample (“exploitation”) and the least frequent sample (“exploration”). If all new samples appeared with the same frequency, then we can select two samples at random. In the case where no new samples were generated, we choose them from the unevaluated samples of the original seed data set in $D_0$.

6 Obtain risk value for new selected samples: same as in TN-QEO as a booster (see Appendix [A4]).

7 Merge the new portfolios with seed data set from step 0 same as in TN-QEO as a booster (see Appendix [A4]).

8 Restart next cycle of the algorithm with the merge data set as the new seed data set: same as in TN-QEO as a booster (see Appendix [A4]).
FIG. 5. Relative quantum enhancement similar to those shown in the bottom panel of Fig. 2 in the main text. For these experiments, portfolio optimization instances with a number of variables ranging from $N = 30$ to $N = 100$ were used. Here, each panel corresponds to a different investment universe corresponding to a random subset of the S&P 500 market index. Note the trend for a larger quantum enhancement as the number of variables (assets) becomes larger, with the largest enhancement obtained in the case on instances with all the assets from the S&P 500 ($N = 500$), as shown in Fig. 2.
Appendix C: Generalization in Quantum Generative Models

In Fig. 6 we further demonstrate the generalization capabilities of the quantum-inspired generative model used in this work.

![Portfolio Counts vs. Risk](image)

FIG. 6. Further demonstration of the generalization capabilities of our quantum-inspired generative model, expanding on those cases presented in Fig. 3 in the main text. Left panel corresponds to another investment universe with \( N = 50 \) assets while the right panel corresponds to one with \( N = 100 \) assets. The blue histogram represents the number of observations from the classical solver (seed data set). In orange we represent samples coming from our quantum generative model at the core of TN-QEO. The inset emphasizes all the new outstanding samples (red sticks) corresponding to lower portfolio risk value (better minima) than that available from the classical solver by itself (blue stick at the lower edge of the classical histogram bars). The number of outstanding samples in the case of \( N = 50 \) is equal to 27, while 139 outstanding samples were obtained from the MPS generative model in the case of \( N = 100 \).