Quantum-Inspired Hamiltonian Descent for Mixed-Integer Quadratic Programming

Shreya Chaudhary[◇,1] **Jinglei Cheng**[◇,2] **Samuel Kushnir**[⋆,3] **Jiaqi Leng**[4]
Pengyu Liu[5] **Yuxiang Peng**[◇,6] **Hanrui Wang**[◇,2] **Xiaodi Wu**[†,7,8]
(Authors are listed in alphabetical order)
[1]Department of Electrical Engineering and Computer Science, MIT
[2]Eigen AI [3]Google
[4]Simons Institute for the Theory of Computing and Department of Mathematics, UC Berkeley
[5]School of Computer Science, Carnegie Mellon University
[6]Department of Computer Science, Purdue University
[7]Department of Computer Science, University of Maryland
[8]Joint Center for Quantum Information and Computer Science, University of Maryland
[†]xiaodiwu@umd.edu

Abstract

Large-scale non-convex optimization problems, often involving mixed-integer variables, arise naturally in domains such as finance and medical imaging. Conventional CPU-based solvers rely on sequential decision-making, which limits their ability to leverage modern hardware accelerators like GPUs. Motivated by recent advances in quantum optimization, particularly Quantum Hamiltonian Descent (QHD), we introduce Quantum-Inspired Hamiltonian Descent (QIHD), a family of algorithms designed for efficient large-scale optimization on GPU clusters. QIHD reformulates optimization tasks as classical dynamical systems and exploits massively parallel GPU simulations to accelerate computation. Despite being classical, QIHD inherits distinctive QHD-like properties—such as tunneling—and demonstrates similar empirical behavior. We provide a scalable GPU implementation in JAX for mixed-integer quadratic programming (MIQP) problems, capable of handling millions of nonzero elements within seconds. Extensive benchmarks on large-scale MIQP problems show that QIHD consistently outperforms the state-of-the-art CPU-based solver Gurobi when time budgets are limited, highlighting its potential as a practical and scalable optimization framework.

1 Introduction

Mixed-integer quadratic programming (MIQP) is an optimization framework characterized by a quadratic objective function defined over a feasible region with both continuous and binary variables:

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & f(x) = \frac{1}{2}x^\top Q x + w^\top x, \\ \text{subject to} \quad & Ax \le b, \quad Cx = d, \quad i \in \mathcal{I}\colon x_i \in \{0,1\}, \quad j \in \mathcal{J}\colon x_j \in [\ell_j, u_j]. \end{aligned} \tag{MIQP}$$

Here, the disjoint index sets $\mathcal{I}$ and $\mathcal{J}$ form a partition of the index set of variables. Such rich problem structures enable MIQP to serve as a unifying framework for a wide range of practical applications, including quantitative finance [30], energy systems [14], and machine learning [7]. Meanwhile, as a generalization of mixed-integer programming, MIQP is NP-hard in general, implying that finding approximate global solutions to large-scale MIQP instances is computationally intractable in practice.

[◇] Jinglei Cheng, Yuxiang Peng, and Hanrui Wang conducted most of this work at Artephi Computing Inc.
[⋆] Samuel Kushnir conducted most of this work at University of Maryland, College Park.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: ScaleOPT: GPU-Accelerated and Scalable Optimization.

Conventional approaches to MIQP problems generally fall into two categories: global methods and metaheuristic methods. Commercial solvers such as Gurobi primarily adopt global strategies like branch-and-bound [33], which iteratively reduce the gap between upper and lower bounds of the objective function. These methods not only provide solutions but also offer certificates of optimality. In contrast, metaheuristic solvers generate locally optimal solutions within shorter runtimes, often drawing inspiration from natural phenomena—for example, genetic algorithms [19] and simulated annealing [20]. A common feature of both categories is their reliance on sequential decision-making, where each optimization step depends heavily on previous choices. This paradigm has aligned well with the strengths of CPUs, which excel at performing complex, sequential operations, enabling these methods to succeed across a broad range of applications over the past decades [15, 3, 35].

However, the era of big data presents new challenges. Many contemporary optimization tasks involve millions of variables and dense representations derived from terabyte- or even petabyte-scale datasets [27, 37]. Processing problems at this scale exceeds the capabilities of traditional CPU-based clusters. Over the past decade, fueled by advances in artificial intelligence and machine learning, graphics processing units (GPUs) have emerged as powerful alternatives [34]. GPUs excel at executing massive numbers of simple operations in parallel, yet fully exploiting this capability requires algorithmic paradigms fundamentally different from those designed for sequential CPU execution. Existing optimization algorithms designed for CPUs, including gloabl solvers and metaheuristic solvers, have heavy dependencies on sequential condition-branching, cross-core communications, random memory access, or high-quality random number generation, while none can be easily engineered on GPUs.

In parallel, quantum computing has been explored since the early 2000s as a promising framework for tackling complex optimization problems [9]. Under certain structural conditions—such as hidden symmetries or algebraic properties—quantum algorithms can achieve super-polynomial speedups over classical approaches for both combinatorial and continuous optimization tasks [26, 16, 24]. Building on these insights, quantum-inspired algorithms have gained traction as classical approaches that borrow principles from quantum computation. Co-evolving with their quantum counterparts, these methods replicate distinctive features of quantum algorithms without requiring large-scale quantum hardware [8]. Importantly, many quantum-inspired algorithms are naturally parallelizable, making them particularly well-suited for GPU acceleration. This positions them as practical and scalable candidates for tackling real-world optimization problems today, while serving as a bridge toward the eventual deployment of full-scale quantum computing.

In this work, motivated by a quantum algorithm named Quantum Hamiltonian Descent (QHD) originally designated for continuous optimization [23], we propose Quantum-Inspired Hamiltonian Descent (QIHD) for mixed-integer quadratic programming. To the best of our knowledge, QIHD is the first quantum-inspired algorithmic framework for optimization with mixed-integer type decision variables. Our main contributions are threefold: (1) inspired by the quantum Hamiltonian dynamics from QHD used to explore non-convex optimization landscapes, we propose a classical time-dependent Hamiltonian dynamics framework in QIHD, whose long-term stable configurations correspond to global solutions of (MIQP); (2) we discretize the proposed dynamics using the symplectic Euler method, yielding an iterative process that can be efficiently simulated on GPUs; and (3) we conduct extensive benchmarking experiments on various subclasses of MIQP problems, demonstrating that QIHD substantially outperforms state-of-the-art classical commercial solvers (e.g., Gurobi) as well as several other classical and quantum-inspired algorithms.

Related work. Leveraging GPUs for optimization has recently attracted significant attention. The primal–dual hybrid gradient (PDHG) algorithm [4], for instance, has been applied to large-scale linear programming [2] and convex quadratic programming [29], where its reliance on matrix multiplications enables efficient GPU acceleration and scalable first-order solutions. In contrast, second-order methods such as interior-point algorithms have fewer GPU-friendly subroutines, yet notable successes have been reported [17, 36], including applications in medical imaging and treatment planning [28]. More recently, neural network–based approaches have also been explored for solving mixed-integer quadratic programming problems [5].

The development of quantum computing has inspired new hardware and algorithmic approaches, particularly through formulations based on Ising models. Simulated bifurcation [12, 18, 11], derived from nonlinear Hamiltonian dynamics, offers an example of a quantum-inspired algorithm that can also be efficiently implemented on GPUs [1]. Similarly, coherent Ising machines [31, 22]—a class of photonic and silicon-based hardware designed to solve Ising problems—have motivated classical simulation frameworks that now serve as practical combinatorial optimizers [32, 21].

Organization. In Section 2, we introduce a classical Hamiltonian dynamics framework whose long-term equilibria correspond to global solutions of MIQP. The discretization of this dynamics using symplectic integrators and its efficient GPU implementation are discussed in Section 3. Finally, Section 4 presents our benchmarking experiments and results.

2 Quantum-inspired Hamiltonian dynamics

2.1 Motivation: Quantum Hamiltonian Descent for Quadratic Programming

Quantum Hamiltonian Descent (QHD) is a quantum algorithm designed for continuous optimization [23, 25]. Drawing inspiration from gradient flows in classical optimization, QHD can be viewed as a quantum analogue that exploits quantum tunneling to more effectively navigate the optimization landscape and accelerate convergence to the global minimum. Its quantum advantage is supported by both theoretical analysis and empirical results [26]. In particular, QHD has been proven to deliver super-polynomial speedups over *any* classical methods for certain black-box non-convex optimization problems [24].

To exemplify the formulation of QHD, we consider a classic nonlinear optimization problem with a quadratic objective and a box-shaped feasible set, namely, box-constrained quadratic programming:

$$\min_{x\in[0,1]^n} \quad f(x) = \frac{1}{2}x^\top Qx + w^\top x. \tag{BoxQP}$$

We denote the feasible set as $\Omega = [0,1]^n$. QHD is described by the following continuous-time dynamics:

$$i\partial_t \Psi(t,x) = H(t)\Psi(t,x), \quad H(t) = \frac{1}{\lambda(t)}\left(-\frac{1}{2}\Delta\right) + \lambda(t)f(x), \tag{1}$$

where $\Psi(t,x)\colon [0,\infty)\times\Omega \to \mathbb{C}$ is a quantum wave function, subject to the Dirichlet boundary condition $\Psi(t,x) = 0$ for all $t \geq 0$ and $x \in \partial\Omega$; $\lambda(t)\colon [0,\infty) \to \mathbb{R}^+$ is an increasing schedule function. $H(t)$ is a quantum Hamiltonian operator, where $(-\Delta/2)$ is the quantum kinetic operator ($\Delta = \sum_{j=1}^n \partial_{jj}^2$ the Laplacian), and the objective f plays the role of a potential operator. Eq. (1) is a time-dependent Schrödinger equation (TDSE) and it preserves the L^2-norm of $\Psi(t,x)$. We require the initial wave function $\Psi(0,x)$ to have a unit L^2-norm, therefore $|\Psi(t,x)|^2$ can be interpreted as a probability density function that evolves with time t.

As $\lambda(t)$ increases over time, the dynamics (1) resemble those of a quantum heavy ball transitioning from wave-like (non-local) motion to particle-like (localized) motion. With a suitable choice of the schedule function $\lambda(t)$ and an appropriate evolution time $t = T$, the resulting probability distribution $|\Psi(T,x)|^2$ becomes concentrated in a neighborhood of the global minimizer of $f(x)$ within the feasible set Ω. Measuring the quantum state in the computational basis (equivalent to sampling from $|\Psi(T,x)|^2$) then yields a point $X \in \Omega$ that serves as an approximate global solution to the problem (BoxQP).

The TDSE (1) can be efficiently simulated in a quantum computer [6], but is classically intractable unless $\texttt{BPP} = \texttt{BQP}$ [38]. While this observation confirms the inherent *quantumness* of QHD, it also presents substantial challenges for studying QHD through numerical methods. Given the limited scale of current quantum hardware, the largest instance solved by QHD to date involves 75 continuous variables [23]. This is certainly a non-trivial problem size, yet still within the capabilities of commercial solvers such as Gurobi.

In physics, the Hamiltonian formalism is a bridge between classical and quantum mechanics: a continuous-variable quantum Hamiltonian operator corresponds to a classical Hamiltonian function that governs a Lagrangian-mechanical system. Specifically, the classical Hamiltonian function corresponding to (1) takes the following form:

$$H(t,X,P) = \frac{1}{2\lambda(t)}\|P\|^2 + \lambda(t)V(X), \quad V(X) = \begin{cases} f(X) & X \in \Omega \\ \infty & X \notin \Omega \end{cases}, \tag{2}$$

where X and P are position and momentum variables, respectively. This Hamiltonian gives rise to the dynamical evolution of a classical particle via Hamilton's equations.[1] In this work, we mainly

[1]The boundary condition of this system of ODEs needs to be specified to be consistent with the infinite potential well outside of Ω.

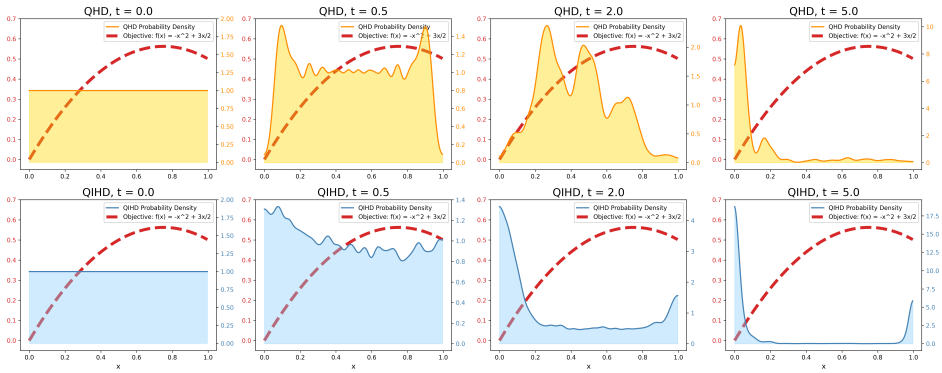


Figure 1: Dynamical behaviors of QHD (top row) and QIHD (bottom row). Success probabilities of QHD and QIHD are 0.86 and 0.79, respectively; both are higher than uniformly random guesses (0.75).

consider (1) absorbing boundary conditions and (2) reflecting boundary conditions.

$$\dot{X}(t) = \frac{\partial H}{\partial P} = \frac{1}{\lambda(t)} P(t), \quad \dot{P}(t) = -\frac{\partial H}{\partial X} = -\lambda(t)\left(QX + w\right), \quad (X(0), P(0)) \sim \rho_0(x, p). \tag{3}$$

Note that (3) describes a stochastic process, where the randomness comes from the initial data $(X(0), P(0))$ that follows the distribution ρ_0.

Although their Hamiltonians are closely related, the process (3) is generally not equivalent to the quantum Hamiltonian dynamics (1), even when initialized with the same distribution $\rho(0, x) = |\Psi(0, x)|^2$. In Figure 1, we compare the evolution of the probability density under QHD and QIHD at various times t, applied to the minimization of a quadratic function $f(x) = -x^2 + \frac{3}{2}x$ over the unit interval $x \in [0, 1]$, whose global minimum lies at $x = 0$. For reference, vanilla projected gradient descent with uniformly random initialization succeeds with probability 0.75. Both QHD and QIHD achieve higher success probabilities than random guessing, with QHD outperforming QIHD. This suggests that QIHD can be viewed as a "weaker" counterpart of QHD, while still inheriting a partial tunneling effect from its quantum origin.

2.2 Quantum-Inspired Hamiltonian Descent

Now, we extend the quantum-classical correspondence to handle binary decision variables and linear constraints. Define the following classical Hamiltonian function, where $A, b, C, d, \mathcal{I}$ are the same as in (MIQP):

$$H_{\mathrm{QIHD}}(t, X, P) = \underbrace{\frac{1}{2\lambda(t)}\|P\|^2}_{\text{Kinetic Energy}} + \underbrace{\lambda(t)V(X)}_{\text{Potential Energy/Objective}} + \underbrace{V_{\mathrm{penalty}}(t, X)}_{\text{Penalty Function}}, \tag{4}$$

where X and P are the position and momentum variable, respectively; $V(X)$ is the same as in (2), and the penalty function reads:

$$V_{\mathrm{penalty}}(t, X) = \begin{cases} \alpha(t)\underbrace{\mathrm{Relu}(AX - b)}_{\text{Ineq. Constraint}} + \beta(t)\underbrace{\|CX - d\|}_{\text{Eq. Constraint}} + \gamma(t)\underbrace{\langle X \odot (e - X), 1_{\mathcal{I}}\rangle}_{\text{Binary Constraint}} & X \in \Omega \\ \infty & X \notin \Omega \end{cases}.$$

Here, $\mathrm{Relu}(z) = \max(0, z)$, $\odot$ represents the element-wise (Hadamard) product, e is the all-one vector, and $1_{\mathcal{I}}$ is a binary vector whose i-th element is 1 only if $i \in \mathcal{I}$. When $X \in \Omega$, the binary constraint equals to $\sum_{i \in \mathcal{I}} X_i(1 - X_i) \geq 0$, and the zero values are obtained if and only if $X_i \in \{0, 1\}$ for all $i \in \mathcal{I}$. For positive $\alpha(t)$, $\beta(t)$, and $\gamma(t)$, the penalty function $V_{\mathrm{penalty}}(t, X)$ is non-negative and the zero values are obtained only when X satisfies all the constraints in (MIQP).

Similar to (3), the Hamiltonian function $H_{\mathrm{QIHD}}(t, X, P)$ generates a Hamiltonian dynamics described by Hamilton's equations:

$$\dot{X} = \nabla_P H_{\mathrm{QIHD}}(t, P, X), \quad \dot{P} = -\nabla_X H_{\mathrm{QIHD}}(t, P, X). \tag{5}$$

With random initializations of the generalized coordinate variable (X, P), this dynamics gives rise to a stochastic process. With appropriate choices of the schedules functions $\{\lambda(t), \alpha(t), \beta(t), \gamma(t)\}$, this process may converge to an equilibrium that is localized near the solutions to (MIQP). We refer to the process generated by (4) as **Quantum-Inspired Hamiltonian Descent**, or simply **QIHD**.

In practice, we take all time-dependent schedule functions (i.e., $\lambda(t)$, $\alpha(t)$, $\beta(t)$, and $\gamma(t)$) to be monotonically increasing. As a result, the Hamiltonian dynamics undergo a phase transition as the evolution progresses. When both $\lambda(t)$ and the penalty term $V_{\mathrm{penalty}}(t)$ are small, the system exhibits chaotic behavior, exploring the full solution space Ω due to its large kinetic energy (i.e., $\|P\|^2/2\lambda(t)$). As time evolves, the potential energy V and the penalty term V_{penalty} increase, while the kinetic energy dissipates, causing the dynamics to become progressively more structured and localized near the low-energy manifold of the effective energy surface $\lambda(t)V(X) + V_{\mathrm{penalty}}(t, X)$. This manifold corresponds to the global minimum of the problem (MIQP).

3 Large-Scale Implementation with GPU

3.1 Symplectic integration

Hamiltonian dynamics preserves the symplectic form, which is essentially equivalent to the volume in the phase space. Standard numerical integration schemes, such as Runge-Kutta methods, do not preserve the symplectic structure of Hamiltonian dynamics, usually leading to distortion of the phase space and poor stability in numerical computation. Instead, we need to incorporate numerical methods that preserve the symplectic structure of the Hamiltonian dynamics, known as *symplectic integrators* [13, 10].

To simulate QIHD, we consider the (first-order) symplectic Euler method. The corresponding update rule is given below. We denote an integer k as the iteration number, and $s > 0$ is a pre-fixed step size. The update rule is ($t_k = ks$, effective evolution time at step k):

$$\begin{cases} p_{k+1} &= p_k - s\nabla_x H(t_k, x_k, p_{k+1}) = p_k - s\left[\lambda(t_k)(Qx_k + w) + \nabla_x V_{\mathrm{penalty}}(t_k, x_k)\right], \\ x_{k+1} &= x_k + s\nabla_p H(x_k, p_{k+1}) = x_k + sp_{k+1}/\lambda(t_k). \end{cases} \tag{6}$$

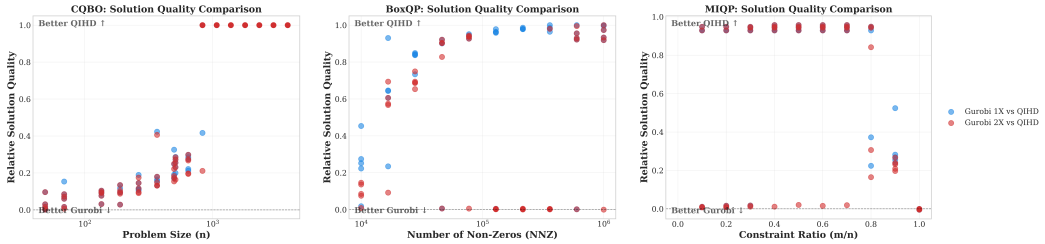
Note that while the symplectic Euler method is implicit for general Hamiltonian systems, it turns out to be an explicit scheme in our case since the QIHD Hamiltonian is separable. This nice structure significantly simplifies the numerical update step.

QIHD begins by drawing random initial positions and momenta for each sample. As a metaheuristic, it generates multiple initial samples that evolve independently under the same dynamics. Since the update equations consist solely of matrix multiplications and vector operations for MIQP problems, QIHD naturally parallelizes across all samples, enabling efficient large-scale GPU acceleration.

3.2 GPU implementation

Our implementation of QIHD leverages JAX, a Python library for array computation and program transformation targeting multiple hardware accelerators like GPUs and TPUs. With JAX, our QIHD implementation is batchable and can be distributed to multiple GPUs easily. These features are especially desired when solving multiple optimization problems of the same size. JAX's native just-in-time compilation ensures the array computation is optimized towards the hardware.

QIHD's outputs are not necessarily locally optimal due to the remainder of kinetic energy in the system. To ensure the local optimality of the outputs, we post-process QIHD's outputs with an implementation of restarted accelerated primal-dual hybrid gradient (rAPDHG) from MPAX, a JAX-based solver for mathematical programming. Since rAPDHG is for continuous optimization, we fix the integer variables in the outputs of QIHD and feed the vectors to MPAX as warm-starts. Although there is no performance guarantee of rAPDHG for non-convex quadratic programming problems, rAPDHG exhibits great performance in finding local minimum in our empirical study.



(a) CQBO benchmark sweeping problem size

(b) BoxQP benchmark sweeping density

(c) MIQP benchmark sweeping the number of binary variables

Figure 2: Benchmarking QIHD against Gurobi on randomly generated problem instances. Gurobi is given time limits of 1× and 2× QIHD's runtime. QIHD performs comparably to or better than Gurobi on nearly all cases.

4 Benchmarking Experiments

We evaluate QIHD on randomly synthesized benchmarks across multiple problem types. All QIHD experiments are executed on a machine equipped with four NVIDIA RTX A5000 GPUs. As a baseline, we compare against Gurobi, a commercial global solver that guarantees optimality, running on two AMD EPYC 7302 CPUs with 256GB RAM. To highlight QIHD's advantages, we limit Gurobi's runtime to either 1× or 2× the end-to-end runtime of QIHD with post-processing, with the number of samples fixed at 1000.

To compare solution quality, we define a metric called the *relative gap*. Let f_Q and f_G denote the objective function values of solutions obtained by QIHD and Gurobi, respectively. The relative gap is defined as: $Rel(f_Q, f_G) := \frac{f_G - f_Q}{\max\{|f_Q|, |f_G|\}}$. A positive value, $Rel(f_Q, f_G) > 0$, indicates that QIHD produces a better solution than Gurobi.

Constrained Quadratic Binary Optimization (CQBO). CQBO is a special case of (MIQP) where all variables are binary, i.e., $\mathcal{J} = \emptyset$. We randomly synthesize dense problems with sizes ranging from 50 to 5000 binary variables and 25% of that number of constraints to showcase how QIHD performs over different sizes of problems. The benchmark compares QIHD against Gurobi on CQBO in Figure 2a. Results in Figure 2a show that QIHD consistently outperforms Gurobi across all problem sizes, with the performance gap widening as problem size increases. This improvement arises because global methods (e.g., branch and bound) are not designed to efficiently produce high-quality suboptimal solutions under tight time limits.

Box-constrained Quadratic Programming (BoxQP). BoxQP is arguably the simplest form of continuous non-convex optimization problems. It is a special case of (MIQP) with $\mathcal{I} = \emptyset$ and A is the zero matrix. The problem is still NP-hard when Q is not positive semi-definite. We synthesize problems with dimension 1000 but with different densities, where the number of non-zero entries ranges from 10^4 to 10^6. As shown in Figure 2b, QIHD consistently produces solutions comparable to or better than Gurobi. Importantly, Gurobi's performance degrades significantly as density increases, while QIHD benefits from GPU-accelerated dense matrix multiplications, leading to growing advantages at higher densities.

Mixed-Integer Quadratic Programming (MIQP). Finally, we evaluate the general mixed-integer quadratic programming (MIQP) problem. Benchmarks are conducted on dense problems with 1000 variables and 250 linear constraints. We specify the first m variables as binary, varying m to study the effect of increasing discrete variables. Results in Figure 2c show that QIHD consistently performs at least as well as Gurobi and often outperforms it, especially when the proportion of continuous variables is large. However, the relative advantage of QIHD drops as the proportion of binary variables increases, likely because Gurobi performs better for quadratic objectives when binary variables are the majority.

Acknowledgments and Disclosure of Funding

We thank the anonymous reviewers for their constructive feedback and thank Lei Fan, Wotao Yin, Haihao Lu, Jinwen Yang, and Chaoyue Zhao for their helpful discussions.

References

[1] Romain Ageron, Thomas Bouquet, and Lorenzo Pugliese. Simulated Bifurcation (SB) algorithm for Python, November 2023.

[2] David Applegate, Mateo Díaz, Oliver Hinder, Haihao Lu, Miles Lubin, Brendan O'Donoghue, and Warren Schudy. Practical large-scale linear programming using primal-dual hybrid gradient. *Advances in Neural Information Processing Systems*, 34:20243–20257, 2021.

[3] M Asghar Bhatti. *Practical optimization methods: With Mathematica® applications*. Springer Science & Business Media, 2012.

[4] Antonin Chambolle and Thomas Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of mathematical imaging and vision*, 40(1):120–145, 2011.

[5] Ziang Chen, Xiaohan Chen, Jialin Liu, Xinshang Wang, and Wotao Yin. Expressive power of graph neural networks for (mixed-integer) quadratic programs. *arXiv preprint arXiv:2406.05938*, 2024.

[6] Andrew M Childs, Jiaqi Leng, Tongyang Li, Jin-Peng Liu, and Chenyi Zhang. Quantum simulation of real-space dynamics. *Quantum*, 6:860, 2022.

[7] Edwin KP Chong, Wu-Sheng Lu, and Stanislaw H Zak. *An Introduction to Optimization: With Applications to Machine Learning*. John Wiley & Sons, 2023.

[8] Elizabeth Crosson and Aram W Harrow. Simulated quantum annealing can be exponentially faster than classical simulated annealing. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 714–723. IEEE, 2016.

[9] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum computation by adiabatic evolution. *arXiv preprint quant-ph/0001106*, 2000.

[10] Kang Feng and Mengzhao Qin. *Symplectic geometric algorithms for Hamiltonian systems*, volume 449. Springer, 2010.

[11] Hayato Goto, Kotaro Endo, Masaru Suzuki, Yoshisato Sakai, Taro Kanao, Yohei Hamakawa, Ryo Hidaka, Masaya Yamasaki, and Kosuke Tatsumura. High-performance combinatorial optimization based on classical mechanics. *Science Advances*, 7(6):eabe7953, 2021.

[12] Hayato Goto, Kosuke Tatsumura, and Alexander R Dixon. Combinatorial optimization by simulating adiabatic bifurcations in nonlinear hamiltonian systems. *Science advances*, 5(4):eaav2372, 2019.

[13] Ernst Hairer, Christian Lubich, and Gerhard Wanner. Structure-preserving algorithms for ordinary differential equations. *Geometric numerical integration*, 31, 2006.

[14] Benjamin F Hobbs. Optimization methods for electric utility resource planning. *European Journal of Operational Research*, 83(1):1–20, 1995.

[15] Michael D Intriligator. *Mathematical optimization and economic theory*. SIAM, 2002.

[16] Stephen P Jordan, Noah Shutty, Mary Wootters, Adam Zalcman, Alexander Schmidhuber, Robbie King, Sergei V Isakov, Tanuj Khattar, and Ryan Babbush. Optimization by decoded quantum interferometry. *arXiv preprint arXiv:2408.08292*, 2024.

[17] Jin Hyuk Jung and DIANNE P O'Leary. Implementing an interior point method for linear programs on a cpu-gpu system. *Electronic Transactions on Numerical Analysis*, 28(174-189):37, 2008.

[18] Taro Kanao and Hayato Goto. Simulated bifurcation assisted by thermal fluctuation. *Communications Physics*, 5(1):153, 2022.

[19] Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. A review on genetic algorithm: past, present, and future. *Multimedia tools and applications*, 80(5):8091–8126, 2021.

[20] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[21] Timothée Leleu, Farad Khoyratee, Timothée Levi, Ryan Hamerly, Takashi Kohno, and Kazuyuki Aihara. Scaling advantage of chaotic amplitude control for high-performance combinatorial optimization. *Communications Physics*, 4(1):266, 2021.

[22] Timothée Leleu, Yoshihisa Yamamoto, Peter L McMahon, and Kazuyuki Aihara. Destabilization of local minima in analog spin systems by correction of amplitude heterogeneity. *Physical review letters*, 122(4):040607, 2019.

[23] Jiaqi Leng, Ethan Hickman, Joseph Li, and Xiaodi Wu. Quantum hamiltonian descent. *arXiv preprint arXiv:2303.01471*, 2023.

[24] Jiaqi Leng, Kewen Wu, Xiaodi Wu, and Yufan Zheng. (sub) exponential quantum speedup for optimization. *arXiv preprint arXiv:2504.14841*, 2025.

[25] Jiaqi Leng, Yufan Zheng, Zhiyuan Jia, Lei Fan, Chaoyue Zhao, Yuxiang Peng, and Xiaodi Wu. Quantum hamiltonian descent for non-smooth optimization. *arXiv preprint arXiv:2503.15878*, 2025.

[26] Jiaqi Leng, Yufan Zheng, and Xiaodi Wu. A quantum-classical performance separation in nonconvex optimization. *arXiv preprint arXiv:2311.00811*, 2023.

[27] Zhen-Chun Li and Ying-Ming Qu. Research progress on seismic imaging technology. *Petroleum Science*, 19(1):128–146, 2022.

[28] Felix Liu, Albin Fredriksson, and Stefano Markidis. A gpu-accelerated interior point method for radiation therapy optimization. *arXiv preprint arXiv:2405.03584*, 2024.

[29] Haihao Lu and Jinwen Yang. A practical and optimal first-order method for large-scale convex quadratic programming. *arXiv preprint arXiv:2311.07710*, 2023.

[30] Renata Mansini, WŁ ,odzimierz Ogryczak, M Grazia Speranza, and EURO: The Association of European Operational Research Societies. *Linear and mixed integer programming for portfolio optimization*, volume 21. Springer, 2015.

[31] Peter L McMahon, Alireza Marandi, Yoshitaka Haribara, Ryan Hamerly, Carsten Langrock, Shuhei Tamate, Takahiro Inagaki, Hiroki Takesue, Shoko Utsunomiya, Kazuyuki Aihara, et al. A fully programmable 100-spin coherent ising machine with all-to-all connections. *Science*, 354(6312):614–617, 2016.

[32] Naeimeh Mohseni, Peter L McMahon, and Tim Byrnes. Ising machines as hardware solvers of combinatorial optimization problems. *Nature Reviews Physics*, 4(6):363–379, 2022.

[33] David R Morrison, Sheldon H Jacobson, Jason J Sauppe, and Edward C Sewell. Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Optimization*, 19:79–102, 2016.

[34] John Nickolls and William J Dally. The gpu computing era. *IEEE micro*, 30(2):56–69, 2010.

[35] Donald A Pierre. *Optimization theory with applications*. Courier Corporation, 2012.

[36] Edmund Smith, Jacek Gondzio, and Julian Hall. Gpu acceleration of the matrix-free interior point method. In *International Conference on Parallel Processing and Applied Mathematics*, pages 681–689. Springer, 2011.

[37] Paul Suetens. *Fundamentals of medical imaging*. Cambridge university press, 2017.

[38] Yufan Zheng, Jiaqi Leng, Yizhou Liu, and Xiaodi Wu. On the computational complexity of schrödinger operators. *arXiv preprint arXiv:2411.05120*, 2024.

Appendices

A Simulated Bifurcation

SB is inspired by the quantum adiabatic evolution of a network of Kerr-nonlinear parametric oscillators (KPOs), whose Hamiltonian is given by

$$H(t) = \hbar \sum_{i=1}^{N} \left[\frac{K}{2} a_i^{\dagger 2} a_i^2 - \frac{p(t)}{2} (a_i^{\dagger 2} + a_i^2) + \Delta_i a_i^\dagger a_i \right] - \hbar \xi_0 \sum_{i=1}^{N} \sum_{j=1}^{N} J_{i,j} a_i^\dagger a_j. \tag{7}$$

By approximating the expectation value of a_i (and $a_i^\dagger$) as a complex amplitude $x_i + iy_i$ (and $x_i - iy_i$), we derive the classical approximation of the quantum system:

$$H_c(x,y,t) = \sum_{i=1}^{N} \left[\frac{K}{4} (x_i^2 + y_i^2)^2 - \frac{p(t)}{2} (x_i^2 - y_i^2) + \frac{\Delta_i}{2} (x_i^2 + y_i^2) \right] - \frac{\xi_0}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} J_{i,j} (x_i x_j + y_i y_j). \tag{8}$$

To achieve faster numerical implementation, the Hamiltonian (8) can be further simplified to the following one, which we will refer to as the original *adiabatic SB* (aSB) [12],

$$H_{\mathrm{aSB}} = \frac{a_0}{2} \sum_{i=1}^{N} y_i^2 + V_{\mathrm{aSB}}, \quad V_{\mathrm{aSB}} = \sum_{i=1}^{N} \left(\frac{x_i^2}{4} + \frac{a_0 - a(t)}{2} x_i^2 \right) - \frac{c_0}{2} \sum_{i,j}^{N} J_{i,j} x_i x_j, \tag{aSB}$$

The aSB dynamics is defined over the full Euclidean space and sometimes the state vector $x(t)$ can move away from the unit box. To improve the numerical accuracy, two modified SB dynamics are proposed in a follow-up work [11], namely, the *ballistic SB* (bSB) and the *discrete SB* (dSB),

$$H_{\mathrm{bSB}} = \frac{a_0}{2} \sum_{i=1}^{N} y_i^2 + V_{\mathrm{bSB}}, \quad V_{\mathrm{bSB}} = \begin{cases} \sum_{i=1}^{N} \frac{a_0 - a(t)}{2} x_i^2 - \frac{c_0}{2} \sum_{i,j}^{N} J_{i,j} x_i x_j & |x| \le 1 \\ +\infty & \text{otherwise} \end{cases} \tag{bSB}$$

$$H_{\mathrm{dSB}} = \frac{a_0}{2} \sum_{i=1}^{N} y_i^2 + V_{\mathrm{dSB}}, \quad V_{\mathrm{dSB}} = \begin{cases} \sum_{i=1}^{N} \frac{a_0 - a(t)}{2} x_i^2 - \frac{c_0}{2} \sum_{i,j}^{N} J_{i,j} x_i \mathrm{sign} x_j) & |x| \le 1 \\ +\infty & \text{otherwise} \end{cases} \tag{dSB}$$

Note that the Hamiltonian dynamics generated by both bSB and dSB are restricted to the unit box $[0,1]^N$ due to the infinite potential wall (*inelastic wall* in [11]) on the boundary.

Reinterpreting SB as a Hamiltonian descent. An Ising problem can be reformulated as the following continuous optimization problem:

$$\min_x f(x) = -\frac{1}{2} \sum_{i,j}^{N} J_{i,j} x_i x_j, \tag{9}$$

$$\text{subject to} \quad 1 - x_i^2 = 0, -1 \le x_i \le 1 \quad \forall i \in [N]. \tag{10}$$

And the time-dependent part in V_{bSB} can be regarded as a "soft" penalty added to the objective function:

$$H_{\mathrm{bSB}} = \underbrace{\frac{a_0}{2} \sum_{i=1}^{N} y_i^2}_{\text{kinetic energy}} + \underbrace{\frac{a(t) - a_0}{2} \left(\sum_{i=1}^{N} -x_i^2 \right)}_{\text{time-varying penalty}} + \underbrace{\left(-\frac{c_0}{2} \sum_{i,j}^{N} J_{i,j} x_i x_j \right)}_{\text{objective function}} \tag{11}$$

B QIHD Dynamics

Consider the Hamiltonian:

$$H(t) = \frac{a_0}{2} \sum_i y_i^2 + V(t,x), \tag{12}$$

where the potential function is defined by

$$V(t,x) = \begin{cases} \frac{a_0 - a(t)}{2} \sum_{i \in \mathcal{I}} x_i(x_i - 1) + g(t)\mathrm{ReLu}(Ax - b) + c_0 f(x) & x \in [0,1]^N \\ +\infty & \text{otherwise} \end{cases} \tag{13}$$

The corresponding Hamilton's equation is

$$\begin{cases} \dot{x} & = a_0 y, \\ \dot{y} & = -\nabla_x V(t) = -\left[\frac{a_0 - a(t)}{2} r + g(t) \sum_{j \in [m]} 1_{>0}(a_j^T x - b_j) a_j + c_0 \left(Qx + w\right) \right] \end{cases} \tag{14}$$

where r is a vector defined by

$$r_i = \begin{cases} 1 - 2x_i & i \in \mathcal{I} \\ 0 & i \in \mathcal{J}. \end{cases} \tag{15}$$

If we want to have a dSB-style numerical integration, the last part in the $\nabla_x V(t,x)$ should be

$$c_0 \left(Q\tilde{x} + w\right), \quad \tilde{x}_j = \begin{cases} 1_{>1/2}(x_j) & j \in \mathcal{I}, \\ x_j & j \in \mathcal{J}, \end{cases} \tag{16}$$

and the second part in $\nabla_x V(t,x)$ should be

$$g(t) \sum_{j \in [m]} 1_{>0}(a_j^T \tilde{x} - b_j) a_j \tag{17}$$

We may choose c_0 as follows:

$$c_0 = \frac{0.5}{C\sqrt{N}}, \quad C = \sqrt{\frac{\sum_{i,j} Q_{i,j}^2 + \sum_j w_j^2}{N(N+1)}}. \tag{18}$$