Approximate optimization of MaxCut with a local spin algorithm

Aniruddha Bapat[1, *] and Stephen P. Jordan[2, 3, †]

[1] *Joint Center for Quantum Information and Computer Science, University of Maryland, College Park, Maryland 20742, USA*
[2] *Microsoft, Redmond, WA 98052, USA*
[3] *University of Maryland, College Park, MD 20742, USA*

(Dated: April 12, 2021)

Local tensor methods are a class of optimization algorithms introduced in [Hastings, arXiv:1905.07047v2][1] as a classical analogue of the quantum approximate optimization algorithm (QAOA). These algorithms treat the cost function as a Hamiltonian on spin degrees of freedom, and simulate the relaxation of the system to a low energy configuration using local update rules on the spins. Whereas the emphasis in [1] was on theoretical worst-case analysis, we here investigate the underlying dynamics of the algorithm, and find that the local tensor method closely follows discretized, imaginary-time dynamics of the system under the problem Hamiltonian. Then, we study practical performance through benchmarking experiments on instances of the MAXCUT problem. Through heuristic arguments we propose formulas for choosing the hyperparameters of the algorithm which are found to be in good agreement with the optimal choices determined from experiment. We observe that the local tensor method is closely related to gradient descent on a relaxation of MAXCUT to continuous variables, but consistently outperforms gradient descent in all instances tested. We find that time to solution achieved by the local tensor method is highly uncorrelated with that achieved by a widely used commercial optimization package; on some MAXCUT instances the local tensor method beats the commercial solver in time to solution by up to two orders of magnitude and vice-versa.

Binary unconstrained optimization, i.e., the maximization of an objective function on the configuration space of binary variables, is an important NP-hard optimization problem whose restrictions include several problems from Karp's list of 21 NP-complete problems [2]. Due to the hardness of the problems, many solution approaches rely on finding approximately optima in the shortest possible time, or constructing algorithms that have an optimality guarantee but without guarantees on runtime. Algorithms in the latter category are often referred to as exact solvers, and include approaches that use linear, quadratic, or semi-definite programming (LP/QP/SDP) relaxations of the problem instance with techniques to obtain optimality bounds such as cutting planes, branch-and-bound, or Langrangian dual-based techniques. While the design of exact algorithms may be well-suited to theoretical analysis, the runtime scaling in instance size is often poor.

In some cases, it is possible to design polynomial-time algorithms with guarantees on the *approximation ratio*, i.e., the ratio of the optimum obtained to the global maximum. These are, however, ultimately limited by hardness results on achieving approximation ratios above a certain threshold value [3].

In the absence of runtime or optimality guarantees, problem-specific heuristics can nevertheless perform better than expected, exhibiting superior performance in runtime, optimality or both. An important class of heuristics takes inspiration from physical processes seen in nature, and those in this category that mimic the evolution of quantum systems are known as *quantum-inspired optimization* methods.

Quantum-inspired (or "dequantized") algorithms have arisen in recent years of out a rich interplay between physics and algorithms research in the context of quantum computing. Thus, as notions of complexity now find analogues in many-body systems, so do quantum dynamics inform the design of quantum and classical algorithms. In the area of classical optimization, two quantum algorithms have generated considerable interest: quantum annealing and the quantum approximate optimization algorithm (QAOA). Promising developments in quantum annealing have inspired classical heuristic algorithms such as simulated quantum annealing [4] and sub-stochastic monte carlo [5], both of which mimic the evolution of the quantum state under an adiabatically evolving Hamiltonian.

Recent results on the performance of shallow-depth QAOA on the problem of MAX-E3-LIN2 led to improved approximations of corresponding classical algorithms for the same problem [6, 7]. More recently, a new classical heuristic known as Local Tensor (LT) was introduced in [1], taking inspiration from QAOA and closely related to previously

* ani@umd.edu
† stephen.jordan@microsoft.com

known classical heuristics for distributed computing [8]. It was shown in [1] that LT has average-case performance better than single-layer QAOA for triangle-free MAXCUT and MAX-K-LIN2 by tuning only one global hyperparameter, in contrast to the two-parameter tuning required for QAOA. Currently, it is unknown whether LT may be useful as a heuristic more broadly, and if so, how the hyperparameters should be set in practice. In this paper, we address this question by implementing a version of LT and benchmarking it on the problem of MAXCUT. We find that on the instances studied, the performance of LT can be considerably enhanced by hyperparameter tuning, and that it is possible to provide good initial guesses on the hyperparameters as function of the instance description. Under such settings, the performance of LT is comparable to the performance of the commercially available solver, Gurobi.

Our setup is described in Sections I and II, followed by a discussion of hyperparameter tuning and the underlying physics of the algorithm in Sections IV to VI, respectively. Then, we provide a brief description of the problem instances studied (Section III), and compare the performance of tuned LT with those of Gurobi (Section VII) and gradient descent (Section VIII).

I SPIN PROBLEMS

We refer to binary unconstrained optimization problems on spin degrees of freedom $s_i \in \{-1, 1\}$ as spin problems. Most generally, one can express the objective function as a polynomial in the variables. Furthermore, since higher powers of the binary variables are trivial, the polynomial is guaranteed to be degree at most one in each variable. The objective function to be maximized can therefore be viewed (up to a negative sign) as a Hamiltonian of a spin system, and the optimization problem maps to sampling from the ground state of the Hamiltonian. The cost Hamiltonian for a system of n spins with indices $\{1, 2, \ldots, n\}$ can be written as

$$H = \sum_{\alpha} w_{\alpha} \prod_{i\in\alpha} s_i. \tag{1}$$

Therefore, H is a sum of monomials, or *clauses*, where a clause α is supported on the subset of spins $\alpha \subseteq \{1, \ldots, n\}$. The sum is weighted by clause weights w_α. The problem can be fully specified as a weighted hypergraph $G = (V, E, W)$, on vertices $V = \{1, 2, \ldots, n\}$, hyperedges $E = \{\alpha, \cdots\}$, and clause weights $W = \{w_\alpha, \ldots\}$.

A wide range of optimization problems be cast as binary unconstrained maximization problems [9], making this problem description very versatile. A simple (and commonly studied) case is one where the polynomial (1) is quadratic, i.e. $|\alpha| \le 2$ for every α. This case captures several interesting physical systems such as Ising spin glasses, as well as a wide range of graph optimization problems. In this work, we focus on a particular quadratic spin problem, MAXCUT, defined in the following manner. Given a weighted graph $G = (V, E, W)$, we define a *cut* to be a partition of the vertices of the graph into two sets. The weight of the cut (or simply the cut) is then defined as the sum of weights of edges going across the cut. Therefore, for any $A \subset V$, the cut is

$$F(A) := \sum_{i\in A, j\in \bar{A}} w_{ij}. \tag{2}$$

Then, given a graph G, MAXCUT asks for the largest cut of the graph. To show that MAXCUT can be written as a quadratic spin problem, we consider the following encoding: Assign a spin s_i to vertex i. Then, there is a one-to-one mapping between bipartitions of V, $(A, \bar{A})$, and spin configuration, $\mathbf{s} = (s_1, s_2, \ldots, s_n)$, namely, by setting $s_i = +1$ if $i \in A$ and -1 otherwise. Edges ij that lie wholly in either A or $\bar{A}$ do not count towards the cut, while edges between A and $\bar{A}$ do. In terms of the spins, edge ij will count towards the cut iff the spins s_i, s_j have opposite sign. Therefore, we may express the MAXCUT Hamiltonian in the following manner:

$$H_{\text{MAXCUT}} = \frac{1}{4}\sum_{i,j} w_{ij} \cdot (s_i s_j - 1). \tag{3}$$

$$\equiv \frac{1}{2}\mathbf{s}^T \cdot J \cdot \mathbf{s} \tag{4}$$

where $J_{ij} := w_{ij}/2$ with zero diagonal terms, $J_{ii} = 0$, and the last equivalence is an equality up to a constant offset $-\frac{1}{4}\sum_{i,j} w_{ij}$. The cut size for any configuration is the negation of the energy under H_{MAXCUT}. Notice that the ground state of H_{MAXCUT} corresponds to the largest cut in G. Since we are ultimately interested in the maximization problem, we will denote the negation of the energy by E.

Despite its simple statement (and apparent similarity to the polynomial-time solvable problem of MINCUT), MAXCUT is known to be NP-hard [2]. In fact, assuming the unique games conjecture holds, approximating MAXCUT to within a fraction 0.878.. is NP-hard. This is also the best known performance guarantee, achieved by the exact classical algorithm due to Goemans and Williamson (GW) on MAXCUT with non-negative weights. Custom solvers for MAXCUT that that improve on practical performance while sometimes preserving optimality are known [10–12].

MAXCUT is a well-studied problem and often used as a benchmark for new classical, quantum, and quantum-inspired solvers. Benchmarking of certain quantum-inspired optimization methods such as the coherent Ising machine [13] and the unified framework for optimization or UFO (see, e.g., [14]) has yielded promising results. In the following section, we discuss the LT heuristic framework and set up our implementation of the algorithm.

II LOCAL TENSOR FRAMEWORK

Before describing our implementation, we review the local tensor (LT) algorithm framework laid out in [1]. The LT framework provides a general prescription for a class of *local* algorithms for the optimization of a Hamiltonian on spin variables. In a local algorithm, the state (e.g., a spin configuration) is encoded into the nodes of a graph, and the update rule at every node is local in the graph structure, depending only on nodes that are at most a bounded distance away. Local state updates therefore require information transfer among small neighborhoods and not the entire graph. If the graph has bounded degree, this can provide polynomial savings in the running cost of the algorithm. Additional speedup can be obtained in a true distributed model of computing where each node is an individual processor, and communication among nodes is slow compared to the internal operations of each processor.

LT is a local algorithm framework for optimization problems on spin degrees of freedom (such as MAXCUT). In LT, we first relax the domain of every spin variable from the binary set $\{-1, 1\}$ to a continuous superset such as the real interval $[-1, 1]$. By convention, we denote *soft* spins (i.e. those in the continuous domain) by letters u, v, etc. and hard spins by letters r, s, etc. Then, LT simulates dynamics of a soft spin vector $\mathbf{v}$ in discrete time steps, and, at the end of a total number of steps p, retrieves a hard spin configuration $\mathbf{s}$ from the final state via a rounding procedure applied to the soft spins. There is considerable flexibility in this setup, and for ease of study, we construct a specific instance of LT here.

Suppose we are given a MAXCUT instance whose corresponding Hamiltonian (as in (4)) is H. Denote the state of spin i at time t by $v_{i,t}$, and the full state vector by $\mathbf{v}_t = (v_{1,t}, v_{2,t}, \ldots, v_{N,t})$. Then, we perform the following steps in order, simultaneously for all spins $i = 1, \ldots, N$.

1. Initialize all spins uniformly at random, $v_{i,0} \in [-1, 1]$.

2. For $t = 0, 1, \ldots, p-1$, update $v_{i,t} \mapsto v_{i,t+1}$ as follows:

   (a) $v_{i,t.5} = v_{i,t} + cF_{i,t}$ where $F_{i,t} := -\partial H / \partial v_{i,t}$ and c is a real constant.

   (b) $v_{i,t+1} = \tanh(\beta v_{i,t.5})$, where β is a positive constant.

3. After p rounds, round each spin to its sign, $v_{i,p} \mapsto s_i^* = \operatorname{sgn}(v_{i,p}) \in \{-1, 1\}$. Return s_i^*.

The final configuration $\mathbf{s}^*$ is a feasible solution candidate. As there the initial configuration $\mathbf{v}_0$ is sampled at random, an outer loop carries out several independent runs of the algorithm and selects the best solution.

For an instance of size n, the domain of feasible solutions corresponds to the vertices of an n-dimensional hypercube. The relaxation in LT extends the domain to the full hypercube, which allows for small, incremental updates and a well-behaved cost function, at the cost of making the search space infinite. However, the rounding step at the end of the algorithm offsets this drawback in the form of a lenient rounding rule: Return the nearest vertex of the hypercube. Therefore, the final state of the graph is only required to lie in the correct quadrant (or 2^n-ant, to be precise) in order to produce the optimal solution.

The spin update sequence is carried out for a total of p rounds, each consisting of two steps. The *force* $F_i = \partial H / \partial v_i$, calculated for each spin, displaces the spin by an amount proportional to it. We refer to the constant of proportionality c as the *response*. Next, we apply the nonlinear function $\tanh(\beta v)$ to the spin, with a rescaling factor β which we will call the *inverse temperature*. This terminology is motivated by analogy to classical thermodynamics. As discussed in the next section, the soft spin v can be inferred as the expectation of an ensemble of spin configurations. When β is small, the expected value v is close to zero (i.e., random), while for large β the spins are "frozen" (expected value close to ± 1). The number of rounds p, response c, and the inverse temperature β form the hyperparameters of the algorithm, which must be fixed (ideally by optimization) before the algorithm is run on an instance. In theory, the factors c, β can also be made to vary by round under a predetermined or adaptive schedule, in a manner similar to simulated annealing. Here, however, we will consider them to be constant in time.

III SPIN MODEL INSTANCES

In order to test the dynamics and the performance of LT, we choose one of several online repositories of MAXCUT instances, the "Biq Mac" library [15]. Each instance therein is a random graph with edge weights drawn from a particular probability distribution. In addition to the weight distribution, the instances is parameterized by the number of variables n and edge density d (i.e. the expected number of non-zero weight edges). The instance data is tabulated in Table I.

Instance type	**Tag**	**Weight distribution**	**Instance size (n)**	**Clause density (d)**
`g05_`n	`g05`	$w_{ij} \in \{0,1\}$	$60, 80, 100$	0.5
`pm1s_`n	`pm1s`	$w_{ij} \in \{-1,0,1\}$	$80, 100$	0.1
`pm1d_`n	`pm1d`	$w_{ij} \in \{-1,0,1\}$	$80, 100$	0.5
`w`d`_`n	`w`	$w_{ij} \in [-10,10]$	100	$0.1, 0.5, 0.9$
`pw`d`_`n	`pw`	$w_{ij} \in [0,10]$	100	$0.1, 0.5, 0.9$
`ising2.5-`n	`ising2.5`	$w_{ij} \propto \frac{\epsilon_{ij}}{\lvert j-i\rvert^{2.5}}$, $\epsilon_{ij} \sim \mathcal{N}(0,1)$	$100, 150, 200, 250, 300$	–
`ising3.0-`n	`ising2.5`	$w_{ij} \propto \frac{\epsilon_{ij}}{\lvert j-i\rvert^{3.5}}$, $\epsilon_{ij} \sim \mathcal{N}(0,1)$	$100, 150, 200, 250, 300$	–
`t2g`L	`torus`	2-dim. toroidal grid, $w_{\langle ij\rangle} \in \{-1,1\}$	$L^2, L = 5, 6, 7$	$\frac{4}{n-1}$
`t3g`L		3-dim. toroidal grid, $w_{\langle ij\rangle} \in \{-1,1\}$	$L^3, L = 5, 6, 7$	$\frac{6}{n-1}$

TABLE I. The benchmarking instances. Each instance is a random graph on n vertices whose edge weights are chosen from the given distribution. The first column specifies the formatting of instance names, while the second column provides a shorter tag for all instances of a given type. In the case of the `w` instances, we sometimes group the instance type `w` by clause density d, in which case the instances are tagged as `w`d, where $d = 0.1, 0.5, 0.9$. The instance types `g05`, `pm1s`, `pm1d`, `w`, and `pw` are random graphs on n vertices with clause density d, where edge weights are drawn from the distrubtion in column 3. The `ising` instances are a 1-dimensional Ising model with long-ranged interactions falling off as a power (2.5 or 3.0) of the inter-spin distance, with a numerator sampled from the normal distribution. The `torus` instances are periodic, D-dimensional ($D = 2, 3$) spin lattices with random couplings ± 1 along the edges of the lattice (denoted by $\langle ij\rangle$ for two neighboring vertices i, j).

IV LT AS A DISCRETIZED, IMAGINARY-TIME SCHRÖDINGER EVOLUTION

LT describes a particular discrete-time evolution of a spin system under a Hamiltonian H. In this section, we provide a physical underpinning to these dynamics by showing that evolution under LT is closely related to imaginary-time Schrödinger evolution under H.

Given any initial state $|\psi\rangle$ and Hamiltonian H, time-evolution of $|\psi\rangle$ under H is given by the Schrödinger equation $d|\psi\rangle/dt = -iH|\psi\rangle$. The evolution applies a phase to the eigenstates of H proportional to the energy of the state times time, so that low-energy states rotate slowly while highly excited states rotate fast. An analytical tool often employed to access the low-energy spectrum of H is that of analytic continuation to imaginary time. In this, one replaces the time by an imaginary time parameter $\tau := it$, and the (unnormalized) imaginary time Schrödinger equation reads

$$|\dot{\psi}\rangle \equiv d|\psi\rangle/d\tau = -H|\psi\rangle\,. \tag{5}$$

The formal solution to this equation is $|\psi(\tau)\rangle = e^{-H\tau}|\psi(0)\rangle$. Note that $|\psi(\tau)\rangle$ is unnormalized, but we keep track of the normalization $\mathcal{N}(|\psi(\tau)\rangle) \equiv \mathcal{N}(\tau) := \sqrt{\langle\psi(\tau)|\psi(\tau)\rangle}$. In the limit $\tau \to \infty$, and assuming that the ground state of H is non-degenerate, the exponential $e^{-\tau H}$ suppresses contributions from all but the lowest-energy state $|\psi_0\rangle$ of H, which implies that $\lim_{\tau\to\infty}|\psi(\tau)\rangle = |\psi_0\rangle$.

The normalization $\mathcal{N}(\tau)$ has τ-dependence

$$\dot{\mathcal{N}} = \frac{1}{2\sqrt{\langle\psi|\psi\rangle}} \cdot \left(\langle\dot{\psi}|\psi\rangle + \langle\psi|\dot{\psi}\rangle\right) \tag{6}$$

$$= -\frac{\langle H\rangle}{\mathcal{N}} \tag{7}$$

where $\langle H\rangle := \langle\psi|H|\psi\rangle$ is the *unnormalized* expectation value of operator H. The normalized expectation value is given by $\langle\langle H\rangle\rangle := \langle H\rangle/\mathcal{N}^2$.

Next, let H be a Hamiltonian acting on n qubits that is diagonal in the Z basis. Any state $|\psi\rangle$ in this Hilbert space can be mapped to a vector of normalized expectation values of the Pauli operators Z_i, where the index i runs over all spins:

$$\begin{aligned}|\psi(\tau)\rangle &\mapsto (\langle\langle Z_1\rangle\rangle, \langle\langle Z_2\rangle\rangle, \ldots, \langle\langle Z_n\rangle\rangle)\\ &=: (v_1, v_2, \ldots, v_n)\ ,\end{aligned}$$

where $v_i \in [-1,1]$ is the classical spin variable that tracks the normalized expectation of Z_i. The imaginary time-evolution of the spins is given by

$$\dot{v}_i = \frac{d}{d\tau}\left(\frac{\langle Z_1\rangle}{\mathcal{N}^2}\right) \tag{8}$$

$$= \frac{-2\dot{\mathcal{N}}}{\mathcal{N}^3}\langle Z_i\rangle + \frac{1}{\mathcal{N}^2}\frac{d}{d\tau}\langle\psi|Z_i|\psi\rangle \tag{9}$$

$$= 2v_i\langle\langle H\rangle\rangle - \langle\langle HZ_i + Z_iH\rangle\rangle\,. \tag{10}$$

This is essentially an imaginary-time analogue of the Ehrenfest theorem. Since Pauli operators Z_i square to the identity, a diagonal Hamiltonian H can always be written as

$$H = R_iZ_i + S_i\,, \tag{11}$$

for every site i, for some operators R_i, S_i that are not supported on site i. Then, $HZ_i = Z_iH = R_i + S_iZ_i$. Next, we make a mean-field assumption, $\langle\langle H_{\bar{i}}Z_i\rangle\rangle \approx \langle\langle H_{\bar{i}}\rangle\rangle\cdot\langle\langle Z_i\rangle\rangle$, where $H_{\bar{i}}$ is any local operator not supported on site i. Then, it follows that $\langle\langle HZ_i\rangle\rangle \approx \langle\langle R_i\rangle\rangle + v_i\cdot\langle\langle S_i\rangle\rangle$, and $\langle\langle H\rangle\rangle \approx v_i\cdot\langle\langle R_i\rangle\rangle + \langle\langle S_i\rangle\rangle$, which gives

$$\dot{v}_i = 2\left(v_i^2\cdot\langle\langle R_i\rangle\rangle + v_i\langle\langle S_i\rangle\rangle - \langle\langle R_i\rangle\rangle + v_i\cdot\langle\langle S_i\rangle\rangle\right) \tag{12}$$

$$= -2\left(1 - v_i^2\right)\cdot\langle\langle R_i\rangle\rangle\,. \tag{13}$$

Next, we make a substitution $u_i := \tanh^{-1} v_i$ which maps the real line onto the open interval $(-1, 1)$. Then, $\dot{v}_i = 1 - \text{sech}^2(u_i)\dot{u}_i = (1 - v_i^2)\dot{u}_i$, therefore we can write

$$\dot{u}_i = -2\langle\langle R_i\rangle\rangle. \tag{14}$$

Note now that the term $\langle\langle R_i\rangle\rangle$ is precisely the negative expected value of the force on spin i, $dH/dZ_i = R_i = -F_i$. Finally, an imaginary time evolution discretized into small time steps $\delta\tau$ obeys (in mean field)

$$v_i(\tau + \delta\tau) = \tanh\left(2\delta\tau F_i + \tanh^{-1} v_i\right)\,. \tag{15}$$

This equation bears similarity to the update rule for LT. In fact, for v_i sufficiently small and close to steady state v_i^*, we can expand the inverse tangent as $\tanh^{-1} v_i \approx v_i + v_i^3/3 + \ldots \approx -2v_i^{*3}/3 + v_i\cdot(1 + v_i^{*2})$, which looks linear with a modified slope. In principle, one could directly evolve the u_i variables in time as per Eq. (15). However, the v_i have the advantage of being bounded in $[-1, 1]$, while the u_i are unbounded and may suffer from issues of convergence. The above analysis reveals a surprising connection between LT and a discretized, mean-field imaginary time evolution of classical spin expectation values.

Our analysis suggests a generalization of LT to situations where the Hamiltonian is not diagonal in the Z basis. In this context, we can represent each spin i as a 3D rotor $\mathbf{r}_i = (x_i, y_i, z_i) = (\langle\langle X_i\rangle\rangle, \langle\langle Y_i\rangle\rangle, \langle\langle Z_i\rangle\rangle)$ of Pauli expectation values. Since the Paulis square to the identity, a general spin Hamiltonian H can always be written as

$$H = P_iX_i + Q_iY_i + R_iZ_i + S_i\,, \tag{16}$$

for every site i, where P_i, Q_i, R_i, S_i are some Hermitian operators that do not take support on site i. Then, $H_iZ_i + Z_iH_i = 2R_i + 2S_iZ_i$ (and analogously for X_i, Y_i), and therefore

$$\dot{x}_i = -2(1 - x_i^2)\cdot\langle\langle P_i\rangle\rangle\,, \tag{17}$$

and similarly for the other coordinates. More succinctly, if we define $\boldsymbol{\rho}_i := \left(\tanh^{-1} x_i, \tanh^{-1} y_i, \tanh^{-1} z_i\right)$, then the imaginary time evolution becomes

$$\dot{\boldsymbol{\rho}}_i = -2\mathbf{F}_i \tag{18}$$

where $\mathbf{F}_i = \left(\langle\langle\frac{dH}{dX_i}\rangle\rangle, \langle\langle\frac{dH}{dY_i}\rangle\rangle, \langle\langle\frac{dH}{dZ_i}\rangle\rangle\right) = (\langle\langle P_i\rangle\rangle, \langle\langle Q_i\rangle\rangle, \langle\langle R_i\rangle\rangle)$. Then, we can imagine a generalization of LT that discretizes the above equation and simulates the evolution of a 3D rotor. By "rounding" the expectation values of the final state, we arrive at a product state estimate of the ground state. The study of this generalized algorithm will be left as a subject of future work.

V HYPERPARAMETER OPTIMIZATION

In order to talk about the performance of LT on any given instance, we must first consider variations in performance due to parameter setting and randomness. LT (as implemented here) is a family of algorithms in the hyperparameters c, β, p. Moreover, for fixed hyperparameters, any run of the algorithm has randomness due to the choice of initial spin configuration. Therefore, the energy output at the end of a single run of LT is a random variable dependent on (c, β, p). The median final energy with fixed hyperparameters, however, is a determinate quantity, which we denote $E_{c,\beta,p}$.

$$E_{c,\beta,p} = \text{median}_{\mathbf{v}_0\in[-1,1]^{\times n}}\,\text{LT}_{c,\beta,p}(\mathbf{v}_0) \tag{19}$$

where, abusing notation, LT$(\mathbf{v})$ denotes the output energy of LT with input configuration $\mathbf{v}$. By definition, half of the runs of LT are expected to produce an optimum with energy lower than E, making the median energy a useful figure of merit. The true median energy can be approximated in practice by the median value of M independent runs of LT$_{c,\beta,p}$,

$$\text{median}\,\{E_1, E_2, \ldots, E_M\} = \tilde{E} \approx E_{c,\beta,p} \tag{20}$$

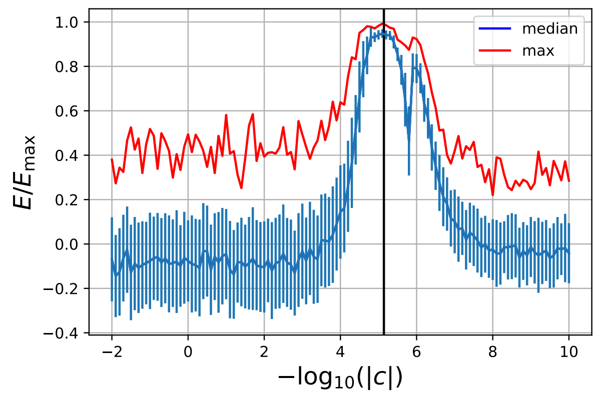


FIG. 1. Hyperparameter sweep for the instance `ising2.5-100` (seed 5555). We vary c over multiple orders of magnitude, and plot the median (lower line) and max (upper line) value of optimum found, normalized by the global optimum, for several runs of the algorithm. It can be seen that peak performance occurs when $c \sim \bar{c}$ (indicated by the vertical black line).

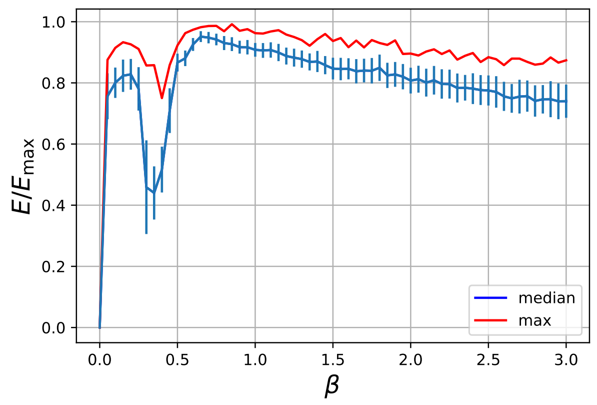


FIG. 2. Hyperparameter sweep for the instance `ising2.5-100` (seed 5555). We plot the median (blue, lower) and max (red, upper) performance as a function of β (arb. units), for a fixed value of $c \sim \bar{c}$. The performance is sensitive to order 1 variation in β, varying from sub-random (cut fraction 0.5) to close to optimal at $\beta \simeq 0.7$. This behavior is typical across all instances studied.

Since we ultimately wish to study the performance of LT as a whole, the hyperparameters must be fixed via a well-defined procedure that takes as input the instance description and returns an (ideally optimal) hyperparameter setting. The most rigorous criterion is global optimization of the performance with respect to each hyperparameter independently. This is important, e.g. to avoid spurious trends in the runtime scaling that arise from imperfect hyperoptimization.

Since this is a computationally expensive task, we focus first on gaining a better understanding of the effect of the hyperparameter on the algorithm performance and providing formulas to minimize the resources needed for hyperparameter optimization.

Response c. The response c is the sensitivity of the spins to force. Intuitively, setting c too large or too small would make the spins too responsive to displacement or frozen, respectively.

Therefore, we expect a regime for c values where the spins are optimally sensitive to the force, and the algorithm should also perform well in this regime. Given a typical length of spins $v_i \sim 1$ and maximum possible force on spin i, $F_i \sim \sum_{j=1}^{N} |J_{ij}|$, a natural guess for c is the inverse of the maximum force. We define

$$\bar{c} = 2 \left\langle \sum_{j=1}^{N} |J_{ij}| \right\rangle_i^{-1} \tag{21}$$

where the brackets $\langle \cdot \rangle_i$ denote a mean over all sites in the graph. The factor of two is chosen purely empirically. As shown in Fig. 1, we find that $\bar{c}$ is indeed a natural scale for the response, and optimal performance is typically found to be within an order 1 factor of $\bar{c}$. Hereafter, we use a rescaled hyperparameter $\eta := c/\bar{c}$.

Inverse temperature β. The β parameter scales the value of the input to the tanh activation function. Intuitively, this enables mapping the displaced spin to the linear response region of the tanh function for maximum sensitivity. This also ensures that the spin stays of order 1 and therefore sensitive to forces applied in subsequent rounds.

In Fig. 2, we show how the choice of β affects mean and best-case performance for a particular instance. The peaks suggest an optimal setting for the value of β. We now make an educated guess for β as we did for the response. For a fixed response $\eta = c/\bar{c}$, spin v_i is displaced as $v_i \to \tanh \beta(v_i + cF_i)$. Since $v_i \leq 1$ for all i, the argument of the tanh function must lie between $[-\beta(1+\eta), +\beta(1+\eta)]$. In order to be maximally sensitive to displacement, we should set β such that $\beta(1+\eta) \sim O(1)$. This gives us a functional dependence between β and η as

$$\beta = \frac{a}{1 + b\eta}, \tag{22}$$

where we have introduced two fitting parameters a, b. From the available instances, this relationship can be checked by extracting the locus of optimal settings for (η, β) and fitting them to the above functional form. The results are shown in Fig. 3. The quality of fits suggests that the functional dependence given in Eq. (22) is accurate. In Fig. 4, we

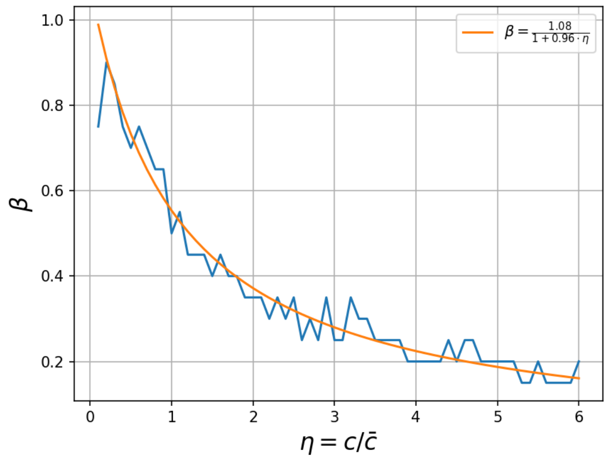


FIG. 3. Optimal β (arb. units) for a range of η values, plotted for a `torus` instance. For each η, the optimal β was found by grid search. The fit to the functional form given in (22) is given by the smooth curve in the figure. The curve profile and quality of fit seen here are typical to all instances studied.

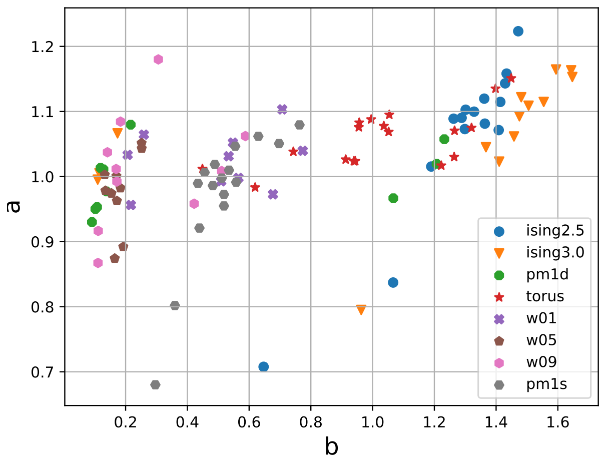


FIG. 4. Clustering in the fit coefficients a, b in (22) for different instances (units arbitrary). We see that the fitting numerator a is close to 1 for most instances, while b varies considerably. There is a reasonable degree of clustering by instance type in b.

show how the coefficients a, b cluster for different problems.

From this analysis, we see that given a problem instance, the hyperparameters η, β can be guessed with very little optimization, and tuned further, if necessary, by local search in a range of order 1 in each parameter.

Number of rounds p. The number of rounds p required by the algorithm is dictated by the convergence to steady state. Qualitatively, this may be connected to the rate of information propagation in

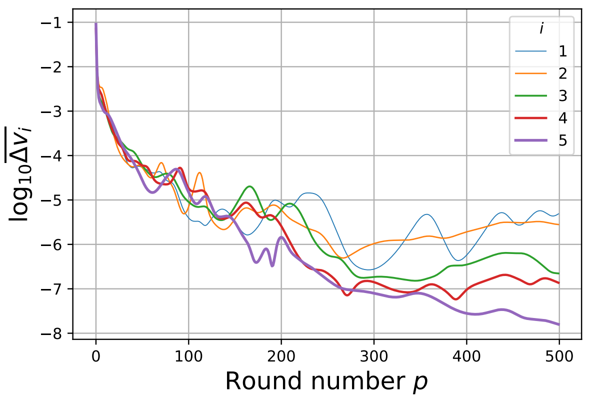


FIG. 5. Displacement between successive rounds as a function of round number, for five arbitrarily chosen spins from the random, 100-spin instance of type `w05`. The log displacement approaches floating point precision after a short number of rounds, indicating that the updates can be terminated early for the rounding step.

the graph, via quantities such as the girth. Unlike for c, β however, a direct guess for p may be harder to obtain.

Instead, we use a dynamic criterion to set the value of p. Since LT is iterative and closely related to gradient descent, we expect that at some point during the algorithm, the spin vector attains a steady state such that all subsequent displacements are smaller than a given threshold. As the final state is determined by the quadrant containing the vector and not the exact value of the vector, small displacements have a small or no effect on the outcome.

In Fig. 5, we plot displacements between successive rounds of a subset of spins in a fixed instance. It is seen that displacements quickly become small; for instance, at $p = 50$, the displacements are of order 10^{-4}. This convergence in the spin values is seen across all spins in a given instance, and all instances studied. Therefore, once the displacement of the state falls below a set threshold, we terminate the algorithm. While this threshold is an additional parameter, it can be set to be sufficiently smaller than the size of the hypercube.

VI DEPENDENCE OF LT DYNAMICS ON THE HYPERPARAMETERS

The implementation of LT studied here allows three hyperparameters, namely, the number of rounds p, the response to force c, and the "inverse temperature" β. We have discussed how to set these hyperparameters for a given MAXCUT instance, giving (in the case of c, β) good initial approximations that depend on the instance description, or (in the case of

p) a dynamic criterion based on the convergence of the state vector. Here we give a physical description of the system dynamics and show, qualitatively, why it is reasonable to expect such behaviors.

A Behavior for a small instance. For a small instance on 5 spins, with real weights on every edge chosen at random, we show the evolution of the full spin configuration as a function of p. We see that the system always finds the same steady-state configuration regardless of initial state (some examples shown in Fig. 6). We can therefore plot how the steady state values of each spin change as a function of β and η, Fig. 7. What we see is that the system undergoes a transition in both parameters, from a phase with zero magnetization on each spin to a phase where the spins have a preferred direction. This is broadly consistent with the interpretation of β and η as being analogous to inverse temperature and interaction strength. At high temperetures or low interaction strength, the spins prefer an unmagnetized configuration while at low temperatures or higher interaction, they find non-zero steady state configurations that correspond to low-energy solutions of the optimization problem. From a computational standpoint, the latter regime is of most interest.

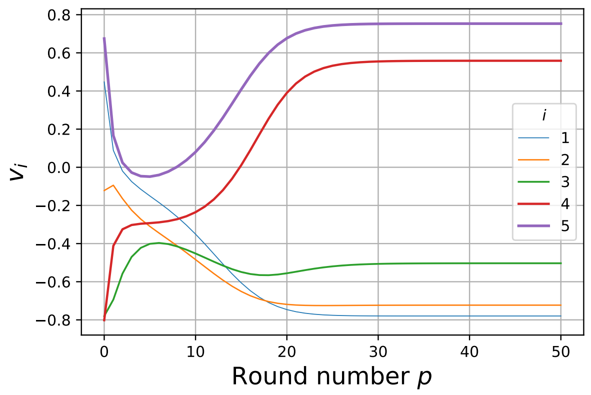
B Dynamics near steady-state. We will analyze the behaviour of LT near a steady state solution $\mathbf{v}^*$ that satisfies $v_i^* = \tanh\left[\beta\left(v_i^* + cF_i^*\right)\right]$ for all spins i. Note that a steady state always exists: the all-zero state $v_i^* = 0$ is an example. More generally, the transcendental equation for steady state, while not guaranteed to have other solutions, can be approximated as a linear equation when $v_i^* \ll 1$, which has non-zero solutions for particular values of β, c. Generically, we expect other steady solutions lying within the hypercube, and find this to be true in our numerics (e.g., Fig. 6).

Suppose, for a given run of the algorithm, the system tends to a particular steady state v^* at long times, with the state at some finite time t given by $\mathbf{v}_t = \mathbf{v}^* + \boldsymbol{\delta}_t$, where $|\delta_{i,t}| \ll |v_i^*|$. Then, to first order in the displacement, we have

$$
\begin{aligned}
v_{i,t+1} &= \tanh\left[\beta\left[(\mathbb{1} + cJ)\cdot(\mathbf{v}^* + \boldsymbol{\delta}_\mathbf{t})\right]_i\right] && (23)\\
&= \tanh\left[\beta\left(v_i^* + cF_i^*\right) + \beta\left[(\mathbb{1} + cJ)\cdot\boldsymbol{\delta}\right]_i\right] && (24)\\
&\simeq \tanh\left[\beta\left(v_i^* + cF_i^*\right)\right] && (25)\\
&+\beta\left[(\mathbb{1} + cJ)\cdot\boldsymbol{\delta}\right]_i \operatorname{sech}^2\left[\beta\left(v_i^* + cF_i^*\right)\right] && (26)\\
&= v_i^* + \beta\left(1 - v_i^{*2}\right)\left[(\mathbb{1} + cJ)\cdot\boldsymbol{\delta}\right]_i && (27)
\end{aligned}
$$

where we used the steady-state condition, and $\tanh'(x) = 1 - \tanh^2(x)$. Therefore, the displacement at time $t+1$ is

$$\boldsymbol{\delta}_{t+1} \simeq \beta\left(\mathbb{1} - V^{*2}\right)\cdot(\mathbb{1} + cJ)\cdot\boldsymbol{\delta}_t\,, \qquad (28)$$



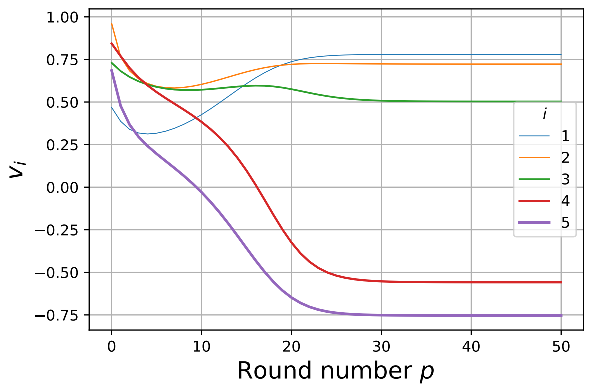


FIG. 6. Evolution of spins for a small instance ($N = 5$) for two randomly chosen initial configurations (units arbitrary). Despite different initial states, the spins are seen to approach the same steady state solution, up to an overall sign. Larger instances may have multiple steady states.

where we defined the diagonal matrix $V_{ii}^* = v_i^*$. Therefore, the norm of the displacement vector close to steady state is bounded as

$$|\boldsymbol{\delta}_{t+1}| \lesssim |\beta|\cdot||\left(\mathbb{1} - V^{*2}\right)||\cdot||(\mathbb{1} + cJ)||\cdot|\boldsymbol{\delta}_t|\,. \quad (29)$$

Since $||\left(\mathbb{1} - V^{*2}\right)|| \leq 1$, and $||\mathbb{1} + cJ|| \leq 1 + |c|\cdot||J||$, it follows that

$$|\boldsymbol{\delta}_{t+1}| \lesssim \beta\cdot(1 + c||J||)\cdot|\boldsymbol{\delta}_t|\,, \qquad (30)$$

assuming $c, \beta \geq 0$. Finally, consider the following properties:

1. Since J has zero diagonal, then by the Gershgorin circle theorem, all eigenvalues of J lie within a disc of radius $\max_i \sum_{j=1}^{n} |J_{ij}|$.

2. If $c = \eta\bar{c}$, where $\bar{c} = \max_i \sum_{j=1}^{n} |J_{ij}|$, then $1 + c||J|| \leq 1 + \eta$.

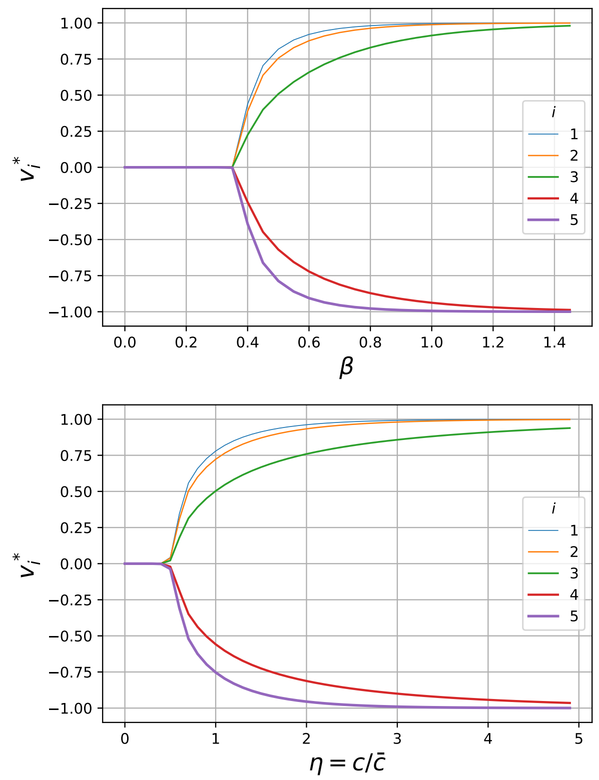


FIG. 7. Steady state spin configurations as a function of β (top, $\eta = 2$) and η (bottom, $\beta = 0.7$) for the same instance as Fig. 6 (units arbitrary). In both cases, the system transitions from an "unmagnetized" phase for low η, β to a "magnetized" phase at high η, β.

Therefore, for a choice $\beta \gtrsim \frac{1}{1+\eta}$, we expect

$$|\boldsymbol{\delta}_{t+1}| \lesssim |\boldsymbol{\delta}_t| \tag{31}$$

giving the condition for dynamics converging to a steady state. Three observations can be drawn from this:

1. $\bar{c}$ provides a natural unit for the response c.

2. For optimal convergence, we expect the dependence between β and $\eta = c/\bar{c}$ to be given by $\beta \simeq a/(1 + b\eta)$ for some parameters a, b.

3. Under the above circumstances, the trajectory near steady state is stable and follows an exponential convergence towards the steady state solution. Therefore, the algorithm can be "safely" terminated when the displacement is under a certain threshold.

These three observations closely match our empirically derived rules for good performance of LT. This indicates that the steady state solutions may also

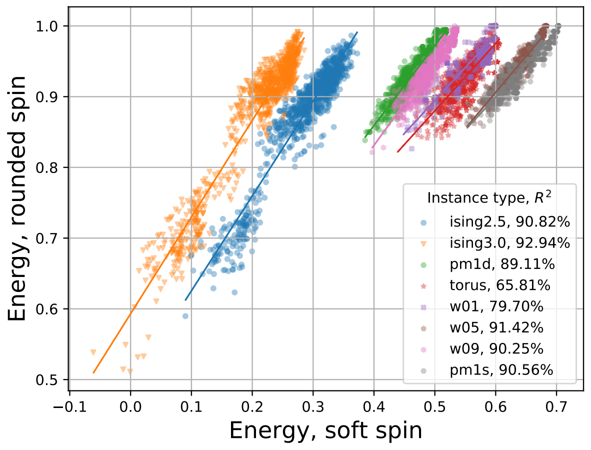


FIG. 8. The obbjective function, normalized by the global optimum, evaluated at the soft spin (y axis) and the corresponding rounded spin obtained at the end of an LT run, for several independent runs of the algorithm on eight instances. Each instance is picked from a different instance type. The correlation between the two quantities (given by R^2 values in the legend) indicates that better steady state soft spin configurations tend to map to better feasible solutions.

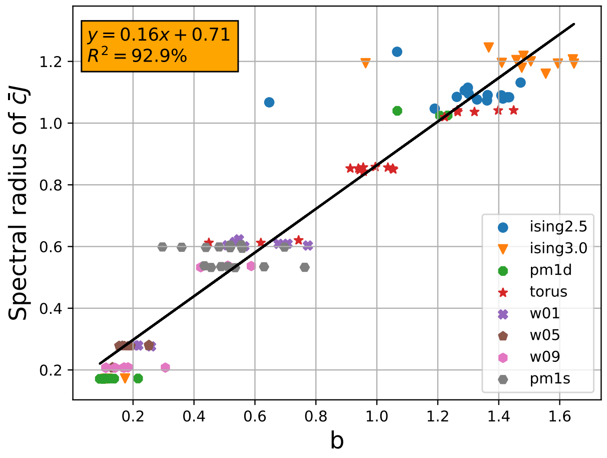


FIG. 9. Dependence of the fitting parameter b and the spectral radius of the the normalized coupling matrix $\bar{c}J$ (units arbitrary). A linear regression fits the data with $R^2 = 92.9\%$, indicating a strong linear relationship between the two quantities. Therefore, the spectral norm of the coupling matrix can give a good estimate on the fitting parameter b and hence β.

correlate with the locations of good feasible solutions (given by the nearest hypercube vertex). This can be seen in Fig. 8.

C Optimal parameters by instance type. In fact, the quantity $\bar{c}$ is defined not as the maximum but as a mean, Eq. (21). However, the functional relationship $\beta = \frac{a}{1+b\eta}$ is still seen to hold for some

a, b.

We can study the dependence of a, b by instance type. As shown in Fig. 4, the parameters form clusters by instance type, and the value of a is close to 1 for all instances studied. The value of b varies considerably from instance to instance. Looking at the functional form, it is reasonable to guess that b is related to the spectral radius of the coupling matrix J. While we bound the magnitude of the largest eigenvalue by $\max_i \sum_{j=1}^{n} |J_{ij}|$, the actual value may be smaller, and b may be understood to reflect this correction. We check this conjecture by plotting the relationship between $||J||$ and b for every problem instance in Fig. 9.

This relationship can be particularly useful if the largest eigenvalue of the matrix can be calculated or estimated quickly. Then, by inverting the linear regression shown in Fig. 9, one obtains a good initial guess for b. The initial guess for a, on the other hand, is simply 1. This potentially reduces the hyperparameter optimization to local minimization in the single parameter η, which is computationally inexpensive.

Having given a physical description of the algorithm, we now turn to studying its performance on practical problem instances.

VII COMPARISON WITH GUROBI

Gurobi is a commercial optimization software that solves a broad range of problems including quadratic programming (QP), linear programming (LP), and mixed integer programming. Additionally, the software includes in-built heuristics to find good initial solution candidates quickly, as well as "pre-solve" subroutines and simplify the problem description by eliminating redundant variables or constraints.

In order to use Gurobi, a MAXCUT instance can be relaxed to either a linear program or a quadratic program. Mapped to an LP, the instance is specified by real variables $x_{ij} \in \mathbb{R}$ to each edge, with the inequality constraints $x_{ij} \leq 1$, where $x_{ij} = 1$ if and only if the edge ij is in the cut. Additional constraints follow by observing that not all configurations are feasible: for example, for three edges ij, jk, ki, at most two may be part of a cut. Any feasible solution must satisfy such *cycle* constraints as well, expressible as inequalities of the form $x_{ij} + x_{jk} + x_{ki} \leq 2$. Then, the LP is formulated as maximization of the objective function $\mathbf{w}^T\mathbf{x}$, subject to the above inequalities, where $\mathbf{w}$ represents the vector of edge weights.

While the number of inequalities to fully characterize the feasible region are exponential in the input, smaller sets of constraints can suffice for good approximate solutions. The *separation problem*, which asks whether a candidate solution is in the polytope or, if not, to give a hyperplane separating it from the polytope, is shown to be polynomial-time in n for a particular choice of constraints [16]. This allows a polynomial-time, cutting-planes algorithm for finding and including violated constraints dynamically for each successive iteration of the LP until success. Theoretical results suggest, however, that solutions to the LP relaxation can be far from the optimal value in general, due to a large *integrality gap* (see, e.g, [17]). The formulation is somewhat unnatural for MAXCUT, and is also seen to perform poorly in practice due to a blowup in the number of variables.

A more natural formulation is as a QP with linear constraints, where every vertex i is assigned to one variable s_i, and the problem is expressed as

$$\begin{aligned} &\max \ \frac{1}{4}\mathbf{s}^T \cdot W \cdot \mathbf{s} \\ &\text{s.t.} \ -1 \leq s_i \leq 1. \end{aligned}$$

Here, the matrix W is the weighted graph Laplacian, with $w_{ii} = \sum_{k \neq i} w_{ik}$ and $W_{ij} = -w_{ij}$ for $i \neq j$. The QP formulation is readily generalized to a semidefinite program by promoting the spin variables to vectors of unit length on a m-dimensional sphere. This forms the basis for the (optimal) Goemans-Williamson algorithm, [18], and is generally the formulation of choice for exact solvers for MAXCUT. While Gurobi does not support an SDP formulation, the QP formulation is supported, with solutions via interior point methods (specifically, a parallel barrier method) and the simplex method. We therefore input our instances into Gurobi as QPs.

Then, we can compare the time to find optimal solution for Gurobi and LT on our benchmarking instances. The time has to be carefully defined in each case for a fair comparison. Gurobi is a deterministic algorithm, except for an initial (optional) heuristic step for proposing an initial solution candidate which takes a small fraction of the total runtime. The algorithm terminates when the optimum is found and proved. The latter typically requires additional time to improve the upper bound on the optimum until it matches the best optimum found. Since we use benchmarking instances that have known optima, we define runtime leniently as the time to find (but not necessarily prove) the optimal solution.

On the other hand, LT is randomized due to the random choice of initial state, and multiple runs are necessary to gather statistics on the performance. Therefore, we define runtime as the median performance over 30 independent runs of LT for each instance. Furthermore, since LT requires parameter

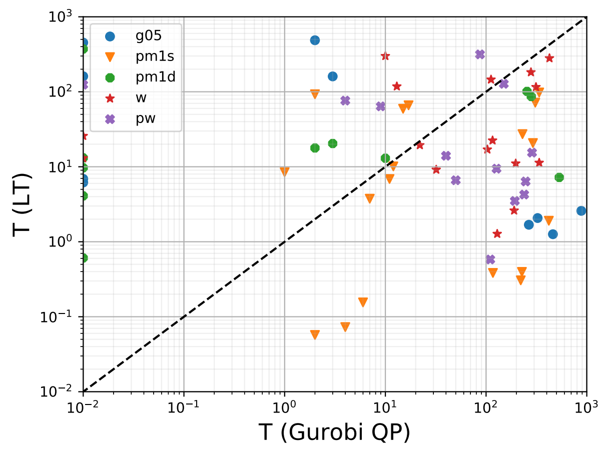


FIG. 10. Performance of LT compared against Gurobi for several benchmarking instances. The performance metric used is median time (in seconds) taken to find the optimum (over 10 runs), with a timeout of 10^3 seconds. Timed out instances are not shown: Out of 130 instances, LT and Gurobi timed out on 47 and 22 instances, respectively, including 7 instances where both timed out. Times faster than a certain threshold are reported by Gurobi as 0s (corresponding to points along the left edge). LT and Gurobi find optima faster than each other in an equal number of instances, with no clear instance-dependent advantange. The speedup on either side is in some cases up to three orders of magnitude.

tuning, we allow up to 20 s of hyperparameter tuning by grid search in β, η that is not considered part of the runtime. Note that the results of Section V suggest that the parameters can be set automatically, either adaptively as for p or by a well-motivated formula for η, β, without the need for a full grid search.

Then, as shown in Fig. 10, the runtime performance of LT and Gurobi can be compared directly on every instance. We see that there is significant spread in performance for every problem type, for both LT and Gurobi. Promisingly, there are instances in every problem class for which LT is significantly faster than Gurobi.

The comparison with Gurobi illustrates that there may be cases where properly tuned LT can outperform state-of-the-art solvers at a fraction of the time cost. It is pertinent to ask whether the success of LT over other solvers can be predicted in advance, using instance data (or quantities derived from it). We briefly address this question.

The most obvious performance indicator is the number of variables n. The instances used in the time comparison with Gurobi were of size 60, 80, or 100. Another elementary indicator is clause density, or the mean number of clauses per variable, which for a weighted instance is the average row sum of the graph adjacency matrix, $m := \sum_{i,j} J_{ij}/n$. We also compute the average row sum of the absolute value of the weight matrix $\bar{m} := \sum_{i,j} |J_{ij}|/n$. Finally, the misfit parameter μ measures the degree of frustration in the model. More precisely, it is the ratio of the ground state energy of the model to the ground state energy of a frustration-free reference system. For a given MAXCUT instance, a reference system with all weights J_{ij} replaced by their negative absolute values $-|J_{ij}|$ is frustration-free, with a ground state energy of $-\sum_{i<j} |J_{ij}|$. On the other hand, the ground state energy of the original instance is bounded below by $-\sum_{i<j} J_{ij}$. Therefore, we define misfit as

$$\mu := \frac{\sum_{i<j} J_{ij}}{\sum_{i<j} |J_{ij}|} . \tag{32}$$

Then, we ask: How well does a given performance indicator predict the runtime of LT (or Gurobi) on a randomly chosen instance? More formally, treating the runtime and indicator as random variables X, Y respectively, the predictive power can be expressed as the conditional entropy $H(Y|X)$, defined as

$$H(Y|X) := -\sum_{x\in\mathcal{X}, y\in\mathcal{Y}} p(x,y) \log \frac{p(x,y)}{p(x)} , \tag{33}$$

where the sum is taken over the support sets of X, Y. Informally, $H(Y|X)$ quantifies the number of additional bits needed to specify Y given knowledge of X. The largest possible value of $H(Y|X)$ is $\log|\mathcal{X}|$ for a discrete sample space $\mathcal{X}$, corresponding in our case to the number of bins used to group the runtimes. We report the conditional entropy normalized by this maximum, so that a normalized entropy of 0 (1) corresponds to perfect (no) predictability. The results are presented in Table II. Relative to Gurobi, the performance of LT is marginally more predictable using the instance data. However, clearly discernable relationships between the performance and any of the indicators studied here could not be obtained using the instance data available, suggesting the need for further systematic study.

VIII COMPARISON WITH GRADIENT DESCENT

An inspection of the LT implementation reveals that the algorithm is operationally very similar to a gradient descent algorithm. The difference lies only in the fact that we apply a nonlinear tanh wrapper to

Predictor	Gurobi	LT
n	0.73	0.69
m	0.68	0.63
$\bar{m}$	0.66	0.59
μ	0.56	0.53

TABLE II. A tabulation of the normalized conditional entropy (as defined in Eq. (33)) of different performance predictors with the runtime of Gurobi and LT on the benchmarking instances. Zero indicates perfect prediction, while 1 corresponds to no predictability. The real-valued predictors $m, \bar{m}, \mu$ were binned into 20 equally spaced intervals, and the runtime was binned into 20 logarithmic intervals spanning the range 0.01s to 1000 s, with an additional bin for timed-out instances ($t > 1000$s).

each spin value in every step, while gradient descent is fully linear. This raises a natural question: does LT offer any advantage to gradient descent?

We formalize this comparison. The MAXCUT Hamiltonian does not have an extremum over $\mathbb{R}^n$, as all of its second (and higher-order) derivatives in any single variable vanish. Put differently, the Hessian of the cost function in the spin variables has zero on-diagonal entries, and is therefore trace zero. So, the Hessian is indefinite everywhere, implying that no point can take an extremal value. This implies that a gradient descent algorithm must constrain the state vector to lie within a closed region of $\mathbb{R}^n$; then the optima are guaranteed to lie on the boundary of this region. The natural choice of region is the n-dimensional hypercube $H_n := [-1, 1]^{\times n}$, whose vertices correspond to feasible solutions to the MAXCUT problem. Then, any step that displaces the state vector outside H_n must be modified to obey the constraint. We implement this by applying a cutoff function to each spin at the end of every displacement step. The form of this function is as follows:

$$\mathrm{cutoff}(x) = \mathrm{sgn}(x) \cdot \min\{1, |x|\}\,. \tag{34}$$

When applied to each spin as $\mathrm{cutoff}(\beta v_i)$, this function has the effect of projecting every spin component that exceeds an allowed range $[-1/\beta, 1/\beta]$ onto the closest boundary of the range. The free parameter β controls how wide the allowed range should be.

The full algorithm may then be written down:

1. Initialize all spins uniformly at random, $v_i \in [-1, 1]$.
2. Apply displacement to spin $v_i \mapsto v_i + c \cdot F_i$ where $F_i = \partial H / \partial v_i$.
3. $v_i \mapsto \mathrm{cutoff}(\beta v_i)$.

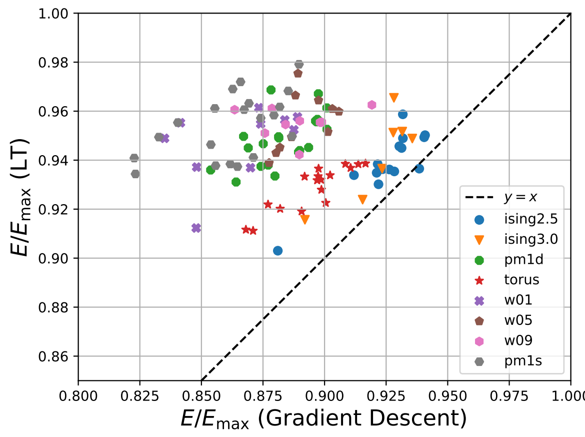


FIG. 11. Performance of LT and gradient descent, given by the energy obtained as a fraction of the maximum (with 1 being optimal), across different benchmarking instances. For each instance shown, the performance has been averaged over 100 trials after a pre-optimization of the hyperparamaters.

4. After p rounds, round each spin to its sign, ± 1.

It is now apparent that GD mirrors LT, with the difference lying in the choice of onsite activation function used: LT uses the tanh function while GD uses a hard cutoff function. Both algorithms have identical free parameters p, c, β that play the same or similar functional roles in each case. Then, we can compare the performances of these algorithms on the same instances. In Fig. 11, we see that LT beats GD on average for the instances studied. This suggests that the specific form of LT that uses a tanh function offers an advantage over a hard cutoff function. This choice also corresponds to the underlying physics described in Section IV.

IX DISCUSSION

The benchmarking of our implementation of LT on the MAXCUT instances gives evidence that LT can perform well in certain practical problem settings. We find that the LT hyperparameters can be set using simple rules that obviate the need for a full, global hyperoptimization, making the algorithm particularly lightweight.

It remains to be seen how well LT fares on problems other than MAXCUT. We expect LT to show similar performance in closely related quadratic unconstrained binary (QUBO) problems. More generally, we remark that the algorithm itself is specified by a domain relaxation, and a notion of derivative of the objective function with respect to each variable. These are minimal requirements found in many optimization problems, for example mixed integer linear programs. An interesting open question is whether LT can be adapted for use in these settings as well. The analysis in Section IV suggests an alternative description of the algorithm as a discretized simulation of imaginary-time dynamics in a spin system. It is interesting whether this picture can be pursued to design improvements or variations to the algorithm, or generalize it to other settings, for instance, on problems like quantum SAT where the problem Hamiltonian is not diagonalizable in any local basis.

Acknowledgments: We thank Helmut Katzgraber, Matt Hastings, and Brad Lackey for useful discussions.

[1] M. B. Hastings, arXiv preprint arXiv:1905.07047 (2019).
[2] R. M. Karp, in *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, The IBM Research Symposia Series, edited by R. E. Miller and J. W. Thatcher (Plenum Press, New York, 1972) pp. 85–103.
[3] J. Håstad, Journal of the ACM (JACM) **48**, 798 (2001).
[4] E. Crosson and A. W. Harrow, in *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)* (IEEE Computer Society, Los Alamitos, CA, USA, 2016) pp. 714–723.
[5] M. Jarret, S. P. Jordan, and B. Lackey, Phys. Rev. A **94**, 042318 (2016).
[6] E. Farhi, J. Goldstone, and S. Gutmann, arXiv preprint arXiv:1411.4028 (2014).
[7] B. Barak, A. Moitra, R. ODonnell, P. Raghavendra, O. Regev, D. Steurer, L. Trevisan, A. Vijayaraghavan, D. Witmer, and J. Wright, in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2015)*, Leibniz International Proceedings in Informatics (LIPIcs), Vol. 40, edited by N. Garg, K. Jansen, A. Rao, and J. D. P. Rolim (Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2015) pp. 110–123.
[8] J. Hirvonen, J. Rybicki, S. Schmid, and J. Suomela, The Electronic Journal of Combinatorics , P4 (2017).
[9] A. Lucas, Front. Physics **2**, 5 (2014).
[10] S. Burer, R. D. Monteiro, and Y. Zhang, SIAM Journal on Optimization **12**, 503 (2002).
[11] F. Liers, M. Jünger, G. Reinelt, and G. Rinaldi, "Computing exact ground states of hard Ising spin glass problems by branch-and-cut," in *New Optimization Algorithms in Physics* (John Wiley & Sons, Ltd, 2004) Chap. 4, pp. 47–69.
[12] F. Rendl, G. Rinaldi, and A. Wiegele, Mathematical Programming **121**, 307 (2010).
[13] Z. Wang, A. Marandi, K. Wen, R. L. Byer, and Y. Yamamoto, Phys. Rev. A **88**, 063853 (2013).
[14] S. Mandrà, Z. Zhu, W. Wang, A. Perdomo-Ortiz, and H. G. Katzgraber, Phys. Rev. A **94**, 022337 (2016).
[15] A. Wiegele, "Binary Quadratic and Max Cut (Biq Mac) Library," `http://biqmac.uni-klu.ac.at/biqmaclib.html` (2007).
[16] F. Barahona and A. R. Mahjoub, Mathematical programming **36**, 157 (1986).
[17] W. F. de la Vega and C. Kenyon-Mathieu, in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07 (Society for Industrial and Applied Mathematics, USA, 2007) p. 5361.
[18] M. X. Goemans and D. P. Williamson, Journal of the ACM (JACM) **42**, 1115 (1995).