OPT2023: 15th Annual Workshop on Optimization for Machine Learning

K-Spin Ising Model for Combinatorial Optimizations over Graphs: A Reinforcement Learning Approach

Xiao-Yang Liu[1] XL2427@COLUMBIA.EDU
Ming Zhu[2,3*] ZHUMINGPASSIONAL@GMAIL.COM
[1]**Columbia University, New York, NY, USA.**
[2]**Institute of Automation, Chinese Academy of Sciences, Beijing, China.**
[3]**University of Chinese Academy of Sciences, Beijing, China.** *

Abstract

Many graph-based combinatorial optimization (GCO) problems are NP-complete and can be formulated by the Ising model. Reinforcement learning (RL) algorithms are promising due to their powerful search abilities. The sampling method in RL is the Monte Carlo Markov chain (MCMC), which collects many samples on a trajectory, while we can only obtain one objective value for a GCO problem if using the Ising model. In this paper, we propose a K-spin Ising model for GCO problems, which integrates well with RL algorithms. First, we propose a *K-spin Ising model* and use its Hamiltonian as the loss function, which collects samples on trajectories. Second, we give the K-spin Hamiltonian functions for several GCO problems. Third, we evaluate our RL approach for the graph maxcut problem on both synthetic and benchmark datasets. Our approach outperforms the commercial solver Gurobi [2] with a speedup of $100\times$ and $10\times$ in small-scale ($100 \sim 3,000$ nodes) and large-scale ($5,000 \sim 10,000$ nodes) graph instances, respectively. On the benchmark dataset, our approach obtains nearly the same best-known results over five compared methods.

1. Introduction

Motivation: Most graph-based combinatorial optimization (GCO) problems are NP-complete. Conventional methods include branch-and-bound [5], cutting plane [15], and randomized search algorithms such as simulated annealing [10]. Many GCO problems can be formulated using the Ising model [14]. We aim to integrate Ising model with reinforcement learning (RL) and define a loss function based on this method.

Challenges: First, the sampling method in RL is the Monte Carlo Markov chain (MCMC) [4], which collects many samples on a trajectory, while we can only obtain an objective value for a GCO problem if using the Ising model. Second, the performance of existing methods may not be good especially in large graph instances.

Contributions: In this paper, we propose a K-spin Ising model for GCO problems, which integrates well with RL algorithms. First, we propose a K-spin Ising model and use its Hamiltonian [13] as the loss function, that is defined on the sampled trajectories and is MCMC in RL. Therefore the Ising model and RL are integrated together in our approach. Second, we provide the K-spin Hamiltonian functions for three GCO problems (graph maxcut, graph partitioning, and minimum vertex cover). Third, we evaluate the K-spin Ising model based RL approach in the graph maxcut problem on both synthetic and benchmark datasets, which exhibits powerful ability in solving GCO problems.

* Corresponding author.

Existing Works: To solve GCO problems, researchers may first formulate it as mixed integer linear programming (MILP). Then the MILP is relaxed to linear programming (LP), and the simplex method [17] is used to obtain LP solutions. However, the LP solutions may generally not be all integers; and therefore, branch-and-bound or cutting plane is used to obtain integer solutions.

Current machine learning methods usually use DRL [11] or supervised/imitation learning methods [16] to generate policies. DRL methods do not require labels, but supervised/imitation learning methods do require. Dai et al. [11] proposed an approach to learn greedy algorithms that exploit the structure of recurring problems, which combines DRL and graph embedding [9]. The solution starts from a partial solution, and a new node with the maximum Q-value is added iteratively until a whole solution is obtained. Chen et al. [6] proposed a method to learn a local search for binary optimization using Monte Carlo policy gradient. Most of them build MDP models for GCO problems and then use DRL to obtain a policy. The expectation of cumulative rewards may diverge in GCO problems, which may lead to inaccurate evaluation of actions. Supervised/imitation learning [16] requires the labels calculated by some other algorithms or solvers, which highly affect the performance.

2. Reinforcement Learning Using K-Spin Ising Model

We show the basic denotations of graphs. Let $G = (V, E, w)$ denote a weighted graph, where V is the node set, E is the edge set, $|V| = N$, $|E| = M$, and $w : E \to \mathbb{R}^+$ is the edge weight function, i.e., $w(u, v)$ is the weight of edge $(u, v) \in E$. $w(u, v) > 0$ if (u, v) is an edge and 0 otherwise.

2.1. Ising Formulation

Consider a 1D Ising model with a ring structure and an external magnetic field h_i, as shown in Fig. 1, there are N nodes with $(N + 1) = 1 \bmod N$; a node i has a spin $\boldsymbol{x}_i \in \{+1, -1\}$ (where $+1$ for up and -1 for down). Two adjacent sites i and $i + 1$ have an energy $w(i, i + 1)$ or $-w(i, i + 1)$ if they have the same direction or different directions, respectively.

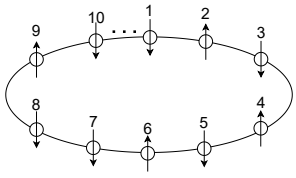

Figure 1: Ising model on a ring.

The whole system will evolve into the ground state with the minimum Hamiltonian [8] :

$$\arg\min_{\boldsymbol{x}} H(\boldsymbol{x}) = \underbrace{-\sum_{i=1}^{N} h_i \boldsymbol{x}_i}_{H_A} + \alpha \underbrace{\sum_{i=1}^{N} -w(i, i+1)\boldsymbol{x}_i \boldsymbol{x}_{i+1}}_{H_B} \quad (1)$$

where α is a weight, H_A is defined on each node's effect on its own, and H_B is defined on each two adjacent nodes' interactions.

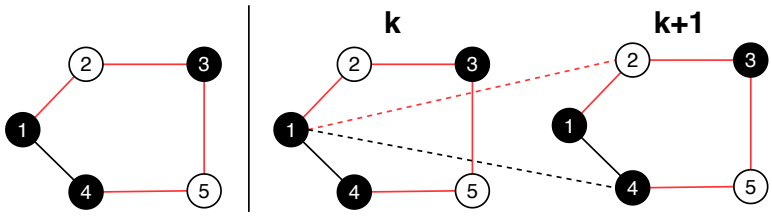

Figure 2: Graph maxcut problem, Ising model (left) vs. K-spin Ising model (right).

We take the **graph maxcut problem** as an example, which is defined as follows. Given a graph $G = (V, E, w)$, split V into two subsets V^+ (with edge set E^+) and V^- (with edge set E^-), and the the cut set is $\delta = \{(i,j)|i \in V^+, j \in V^-\}$. The goal is to maximize the cut value: $\max \sum_{(i,j)\in\delta} w(i,j)$.

We reformulate graph maxcut using an Ising model, which has the following Hamiltonian:

$$\begin{aligned}\min_{\boldsymbol{x}} H(\boldsymbol{x}) &= \sum_{(i,j)\in E} w(i,j)x_i x_j = \sum_{(i,j)\in E^+} w(i,j) + \sum_{(i,j)\in E^-} w(i,j) - \sum_{(i,j)\in\delta} w(i,j) \\ &= \sum_{(i,j)\in E} w(i,j) - 2\sum_{(i,j)\in\delta} w(i,j),\end{aligned} \tag{2}$$

where $\sum_{(i,j)\in E} w(i,j)$ is a constant, $x_i x_j = 1$ if $(i,j) \in E^+$ or E^- and $x_i x_j = -1$ otherwise.

For an illustrative example in the left graph of Fig. 2, the edge set is $E = \{(1,2),(1,4),(2,3),(2,4),(3,5)\}$ and the weights are $w(1,2) = w(1,4) = w(2,3) = w(2,4) = w(3,5) = w(4,5) = 1$. The edge set of black nodes is $E^+ = \{(1,4)\}$, and the edge set of white nodes is $E^- = \emptyset$. The edges connect the two subsets are $\delta = \{(1,2),(2,3),(2,4),(3,5),(4,5)\}$. The solution is $\boldsymbol{x} \in \{-1,+1\}^5$ and the Hamiltonian in (1) becomes

$$\min_{\boldsymbol{x}} H(\boldsymbol{x}) = \boldsymbol{x}_1\boldsymbol{x}_2 + \boldsymbol{x}_1\boldsymbol{x}_4 + \boldsymbol{x}_2\boldsymbol{x}_3 + \boldsymbol{x}_2\boldsymbol{x}_4 + \boldsymbol{x}_3\boldsymbol{x}_5 + \boldsymbol{x}_4\boldsymbol{x}_5. \tag{3}$$

2.2. K-spin Ising Formulation

We only present the K-spin Ising formulation for graph maxcut here, while the formulation of the other problems is moved to Appendix A and B. For the right graph in Fig. 2, we add edges for the k-th and $(k+1)$-th iterations. Take node 1 as an example, in the k-th iteration, node 1 connects node 2, so we set node 2 of the $(k+1)$-th iteration as the end point denoted by the red dash line; and in similar, we set node 4 of the $(k+1)$-th iteration as the end point denoted by the black dash line. The 1st and 2nd steps have the follow Hamiltonian

$$\begin{aligned}\min_{\boldsymbol{x}} H(\boldsymbol{x}^1, \boldsymbol{x}^2) = H(\boldsymbol{x}^1) + H(\boldsymbol{x}^2) + (&\boldsymbol{x}_1^1\boldsymbol{x}_2^2 + \boldsymbol{x}_1^1\boldsymbol{x}_4^2 + \boldsymbol{x}_2^1\boldsymbol{x}_1^2 + \boldsymbol{x}_2^1\boldsymbol{x}_3^2 + \boldsymbol{x}_2^1\boldsymbol{x}_4^2 \\ &+ \boldsymbol{x}_3^1\boldsymbol{x}_2^2 + \boldsymbol{x}_3^1\boldsymbol{x}_5^2 + \boldsymbol{x}_4^1\boldsymbol{x}_2^2 + \boldsymbol{x}_4^1\boldsymbol{x}_5^2 + \boldsymbol{x}_5^1\boldsymbol{x}_3^2 + \boldsymbol{x}_5^1\boldsymbol{x}_4^2).\end{aligned} \tag{4}$$

We consider the agent starts from the initial point $\boldsymbol{x}^1$, and moves (K - 1) steps to a new point $\boldsymbol{x}^K$. In each two consecutive steps, we use edges to connect their nodes and add weights for them. The weights decrease with iterations by a discount factor $\gamma \in (0,1]$. In this way, we obtain the generic K-spin Ising model, and the Hamiltonian is:

$$\begin{aligned}\min_{\boldsymbol{x}^1,\cdots,\boldsymbol{x}^K} H(\boldsymbol{x}^1,\cdots,\boldsymbol{x}^K) &= \sum_{k=1}^{K} H(\boldsymbol{x}^k) + \beta H_C(\boldsymbol{x}^1,\cdots,\boldsymbol{x}^K), \\ &= \sum_{k=1}^{K} H_A(\boldsymbol{x}^k) + \alpha\sum_{k=1}^{K} H_B(\boldsymbol{x}^k) + \beta H_C(\boldsymbol{x}^1,\cdots,\boldsymbol{x}^K),\end{aligned} \tag{5}$$

where β is a weight, and $H_C(\boldsymbol{x}^1,\cdots,\boldsymbol{x}^K)$ is a new term that measures the interactions along two successive iterations:

$$H_C(\boldsymbol{x}^{k-1}, \boldsymbol{x}^k) = -\sum_{i\in V^{k-1}}\sum_{j\in V^k} J_{i,j}^k \boldsymbol{x}_i^{k-1}\boldsymbol{x}_j^k, \tag{6}$$

Algorithm 1: Reinforcement Learning Using K-Spin Ising Model

Input: L (number of epochs), M (number of trajectories), T (length of a trajectory), η (learning rate)
K (number of steps in K-spin Ising model, $K < T$)
Training: train policy network π_θ
01: Build policy network π_θ and randomly initialize it;
02: **for** epoch $l = 1, \ldots, L$
03: **for** $m = 1, \ldots, M$
04: Randomly generate a binary vectors, $\boldsymbol{x}_{l,m}^1$;
05: **for** $t = 2, \ldots, T$
06: $\boldsymbol{p} = \pi_\theta(\boldsymbol{x}_{l,m}^{t-1})$;
07: Obtain $\boldsymbol{x}_{l,m}^t$ by sampling based on Bernoulli($\boldsymbol{p}$);
08: Calculate $H_A(\boldsymbol{x}_{l,m}^t)$, $H_B(\boldsymbol{x}_{l,m}^t)$, and $H_C(\boldsymbol{x}_{l,m}^{t-1}, \boldsymbol{x}_{l,m}^t)$;
09: **if** $t \geq K$
10: Calculate $H_C(\boldsymbol{x}_{l,m}^{t-K}, \cdots, \boldsymbol{x}_{l,m}^t)$ by (7);
11: Calculate Hamiltonian $H_{l,m}^t(\theta)$ of K-spin Ising model by (5);
12: $\theta = \theta + \eta \nabla_\theta \frac{1}{M(T-K+1)} \sum_{m=1}^{M} \sum_{t=K}^{T} H_{l,m}^t(\theta)$;
Testing:
13: Initialize a binary vector $\boldsymbol{x}$;
14: Obtain probability vector: $\boldsymbol{p} = \pi_\theta(\boldsymbol{x})$;
15: **for** $t = 1, \ldots, T$
16: $\boldsymbol{x}^t = \text{Bernoulli}(\boldsymbol{p})$
17: return $\boldsymbol{x}^T$;
Output: $\boldsymbol{x}^T$.

and

$$H_C(\boldsymbol{x}^1, \cdots, \boldsymbol{x}^K) = \sum_{k=2}^{K} H_C(\boldsymbol{x}^{k-1}, \boldsymbol{x}^k), \tag{7}$$

where $J_{i,j}^k = \gamma^{k-1} w(i,j)$.

2.3. Proposed Algorithm

We use policy gradient methods instead of Q-function based methods such as DQN [18] and PPO [20] for two reasons. First, the cumulative rewards may diverge in GCO problems. Second, the Q-function is the mathematical expectation of the cumulative rewards, and therefore is not well defined and may not exist.

The pseudocode of our algorithm is shown in Alg. 1. We use Bernoulli($\boldsymbol{p}$) to denote the Bernoulli distribution with probability vector $\boldsymbol{p}$. Lines $1 \sim 12$ show the training stage, and lines $13 \sim 17$ show the testing stage.

First, we describe the training stage. Line 1 initializes the policy network. Lines 2 and 3 mean that there are L epochs and M trajectories. Line 4 generates an initial solution (binary vector). Line 5 means the length of the sampling trajectory is T. Lines 6 and 7 obtains the probability vector $\boldsymbol{p}$ and the new solution $\boldsymbol{x}_{l,m}^t$ by sampling which follows the Bernoulli distribution, respectively. Line 8 $\sim$ 11 calculate the Hamiltonian. Line 12 updates the parameters of the policy network.

Second, we describe the testing stage. Line 13 generates an initial solution (binary vector). Line 14 obtains a probability vector. Line 15 means that there are T steps for obtaining a new solution. Line 16 obtains a solution by sampling based on Bernoulli distribution. Line 17 returns the new solution.

Table 1: Results for graph maxcut on synthetic instances

Nodes	Edges	Ours	Gurobi	Improvement	Speedup
100	460	**336** (4s)	**336** (494s)	+0%	123.50×
300	2036	**1419** (7s)	1405 (3600s)	+1.00%	514.29×
500	3624	**2508** (7s)	2477 (3600s)	+1.25%	514.29×
700	4036	**2896** (9s)	2862 (3600s)	+1.19%	400.00×
900	5122	**3690** (10s)	3600 (3600s)	+2.50%	360.00×
1000	6368	**4497** (11s)	4447 (3600s)	+1.12%	327.27×
3000	25695	**17340** (35s)	17123 (3600s)	+1.13%	102.86×
5000	50543	**33308** (60s)	30081 (3600s)	+10.73%	60.00×
7000	79325	**51648** (92s)	46514 (3600s)	+11.04%	39.13×
9000	96324	**61121** (109s)	58066 (3600s)	+10.80%	33.03×
10000	100457	**66337** (123s)	59694 (3600s)	+11.13%	29.27×
20000	205364	**134824** (211s)	– (3600s)	–	–

Table 2: Results for graph maxcut on the Gset dataset.

Graph	Nodes	Edges	BLS	DSDP	KHLWG	RUN-CSP	PI-GNN	Ours	Improvement
G14	800	4694	**3064**	-	2922	3061	2943	**3064**	+0%
G15	800	4661	**3050**	2938	**3050**	2928	2990	**3050**	+0%
G22	2000	19990	**13359**	12960	**13359**	13028	13181	**13359**	+0%
G49	3000	6000	**6000**	**6000**	**6000**	**6000**	5918	**6000**	+0%
G50	3000	6000	**5880**	**5880**	**5880**	**5880**	5820	**5880**	+0%
G55	5000	12468	**10294**	9960	10236	10116	10138	10242	−0.51%
G70	10000	9999	**9541**	9456	9458	-	9421	9530	−0.12%

3. Performance Evaluation

We implement our RL approach on a DGX-2 server with NVIDIA A100 GPUs. We only show the results of graph maxcut here. Table 1 shows the result on synthetic data with the number of nodes from 100 to $20,000$. We compare our approach with the SOTA solver Gurobi [2], and its time limit is set to 1 hour. For the instances with $100 \sim 3,000$ nodes, our approach has a little better performance than Gurobi with the speedup of $100\times \sim 500\times$. For the instances with $5,000 \sim 10,000$ nodes, the performance of our approach is about 10% better than Gurobi. For the instances with $20,000$ nodes, Gurobi cannot obtain any solution, but our approach obtains solutions within 4 minutes.

Table 2 presents results of our RL approach and 5 compared approaches in seven instances from Gset [1]. The compared methods include, SDP (DSDP) [7], breakout local search (BLS) [3], Tabu search (KHLWG) [12], recurrent GNN (RUN-CSP) [21], and physical-inspired GNN (PI-GNN) [19]. Compared to the best-known solution, our approach has the same performance in the first five instances, and has a little worse performance in the last two instances (G55 and G70).

4. Conclusion and Future Works

In this paper, we propose a K-spin Ising model for graph-based combinatorial optimizations (GCO). These problems are formulated by K-spin Ising model. We provide a novel loss function, the Hamiltonian of the K-spin Ising model. Experimental results show that our approach achieves good performance on both synthetic and benchmark datasets. We only implemented this method in the graph maxcut problem, and the implementation for other problems is not finished until now. While we provided the implementation process for them, including the Ising and K-spin Ising formulations in Appendix A and B. Implementing them will be future works.

References

[1] Gset dataset at stanford (2023). https://web.stanford.edu/~yyye/yyye/Gset/, .

[2] Homepage of gurobi (2023). https://www.gurobi.com, .

[3] Una Benlic and Jin-Kao Hao. Breakout local search for the max-cut problem. *Engineering Applications of Artificial Intelligence*, 26(3):1162–1173, 2013.

[4] Troels Arnfred Bojesen. Policy-guided monte carlo: Reinforcement-learning markov chain dynamics. *Physical Review E*, 98(6):063303, 2018.

[5] Michael J Brusco, Stephanie Stahl, et al. *Branch-and-bound applications in combinatorial data analysis*, volume 2. Springer, 2005.

[6] Cheng Chen, Ruitao Chen, Tianyou Li, Ruichen Ao, and Zaiwen Wen. Monte carlo policy gradient method for binary optimization. *arXiv preprint arXiv:2307.00783*, 2023.

[7] Changhui Choi and Yinyu Ye. Solving sparse semidefinite programs using the dual scaling algorithm with an iterative solver. *Manuscript, Department of Management Sciences, University of Iowa, Iowa City, IA*, 52242, 2000.

[8] Barry A Cipra. An introduction to the ising model. *The American Mathematical Monthly*, 94 (10):937–959, 1987.

[9] Hanjun Dai, Bo Dai, and Le Song. Discriminative embeddings of latent variable models for structured data. In *International Conference on Machine Learning (ICML)*, pages 2702–2711. PMLR, 2016.

[10] Kathryn Anne Dowsland and Jonathan Thompson. Simulated annealing. *Handbook of natural computing*, pages 1623–1655, 2012.

[11] Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.

[12] Gary A Kochenberger, Jin-Kao Hao, Zhipeng Lü, Haibo Wang, and Fred Glover. Solving large scale max cut problems via tabu search. *Journal of Heuristics*, 19:565–571, 2013.

[13] Xiao-Yang Liu, Zechu Li, Shixun Wu, and Xiaodong Wang. Stationary deep reinforcement learning with quantum k-spin hamiltonian regularization. In *ICLR 2023 Workshop on Physics for Machine Learning*, 2023.

[14] Andrew Lucas. Ising formulations of many np problems. *Frontiers in Physics*, 2:5, 2014.

[15] Hugues Marchand, Alexander Martin, Robert Weismantel, and Laurence Wolsey. Cutting planes in integer and mixed integer programming. *Discrete Applied Mathematics*, 123(1-3): 397–446, 2002.

[16] Vinod Nair, Sergey Bartunov, Felix Gimeno, Ingrid Von Glehn, Pawel Lichocki, Ivan Lobov, Brendan O'Donoghue, Nicolas Sonnerat, Christian Tjandraatmadja, Pengming Wang, et al. Solving mixed integer programs using neural networks. *arXiv preprint arXiv:2012.13349*, 2020.

[17] John C Nash. The (dantzig) simplex method for linear programming. *IEEE Computing in Science & Engineering*, 2(1):29–31, 2000.

[18] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. *Advances in Neural Information Processing Systems*, 29, 2016.

[19] Martin JA Schuetz, J Kyle Brubaker, and Helmut G Katzgraber. Combinatorial optimization with physics-inspired graph neural networks. *Nature Machine Intelligence*, 4(4):367–377, 2022.

[20] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[21] Jan Toenshoff, Martin Ritzert, Hinrikus Wolf, and Martin Grohe. Graph neural networks for maximum constraint satisfaction. *Frontiers in Artificial Intelligence*, 3:580607, 2021.

Appendix A. Problem Formulation for Graph Partitioning

Given a graph G with an even number $N = |V|$ of nodes, partition V into two subsets (V^+ and V^-) of equal size $N/2$ such that the number of edges connecting the two subsets is minimized.

We consider a node $i \in V$, and let x_i be a binary variable with +1 denoting in the subset V^+ and -1 denoting in the subset V^-. The Hamiltonian is

$$H_A = \left(\sum_i x_i\right)^2, \tag{8}$$

$$H_B = \sum_{(i,j)\in E} \frac{1 - x_i x_j}{2} = \sum_{i<j} w(i,j)\frac{1 - x_i x_j}{2}. \tag{9}$$

The 1st and 2nd steps of K-spin Ising formulation for graph partitioning have the follow Hamiltonian

$$\min_{\boldsymbol{x}} H(\boldsymbol{x}^1, \boldsymbol{x}^2) = H(\boldsymbol{x}^1) + H(\boldsymbol{x}^2) + \beta \sum_{i<j} w(i,j)\frac{1 - x_i^1 x_j^2}{2}. \tag{10}$$

Appendix B. Problem Formulation for Minimum Vertex Cover

Given a graph G, find the smallest number of vertices that can be colored such that every edge is incident to a colored vertex.

We consider a vertex i, and let x_i be a binary variable with 1 denoting the vertex is colored and 0 otherwise. The Hamiltonian is

$$H_A = \sum_i x_i, \tag{11}$$

$$H_B = \sum_{(i,j)\in E} (1 - x_i)(1 - x_j) = \sum_{i<j} w(i,j)(1 - x_i)(1 - x_j). \tag{12}$$

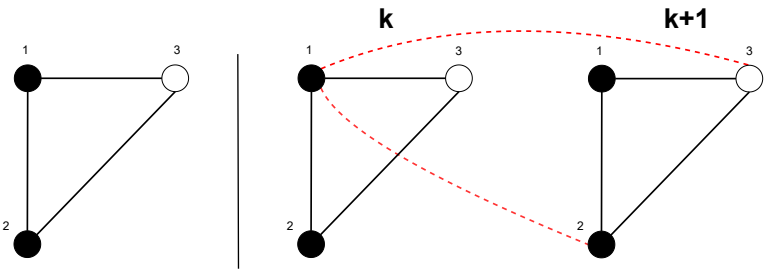

Figure 3: Minimum vertex cover problem, Ising model (left) vs. K-spin Ising model (right).

In Fig. 3, the selected vertices (vertex 1 and 2) are colored as black. The 1st and 2nd steps of K-spin Ising formulation for minimum vertex cover have the follow Hamiltonian

$$\min_{\boldsymbol{x}} H(\boldsymbol{x}^1, \boldsymbol{x}^2) = H(\boldsymbol{x}^1) + H(\boldsymbol{x}^2) + \sum_{i<j} w(i,j)(1 - x_i^1)(1 - x_j^2). \tag{13}$$