Summary: Phase transition in minimizing spin glass energies

Antti Hyvärinen

October 16, 2007

Abstract

This summary describes an algorithm for determining the ground state of a given Ising spin glass instance, and presents some results on locating the phase transition point between Ising spin glass phase and ferromagnet phase by simulations on a large collection of random z-regular graphs of different sizes. The algorithm is based on transforming the problem of determining the ground state of a weighted graph representing an Ising spin glass to the problem of determining the maximum cut in a graph, formulating the maximization problem as an integer linear problem and solving the problem using a sophisticated branch-and-cut algorithm, a backtracking search algorithm for constraint problems. The results are compared to results from Bethe-Peierls approximation on larger z-regular random graphs obtained by Monte-Carlo simulation and the cavity method. Slightly above the phase transition point, the run time of the Branch-and-Cut algorithm shows a sharp decrease, indicating that there is a relation between the physical phenomenon of phase transitions and the difficulty of determining the ground state of an instance.

1 Introduction

The great progress achieved in efficiency of combinatorial algorithms during the last two decades has rendered possible the study of non-trivial physical systems using computers. Many such systems previously deemed practically unsolvable due to the famous NP-completeness result of S. A. Cook have been lately solved within reasonable time limits despite the potentially huge search space associated with such problems.

However, from many domains of NP-complete problems, we find certain instances which resist the efficient search space pruning techniques of the modern algorithms. While it is not surprising that there are difficult instances of NP-complete problems as it is widely believed that P $\neq$ NP, not every instance is equally difficult. In fact, given two seemingly similar problems, one might be solved by the same algorithm within milliseconds, while the other might take days. Consequently, we are faced with the question of what makes a problem hard.

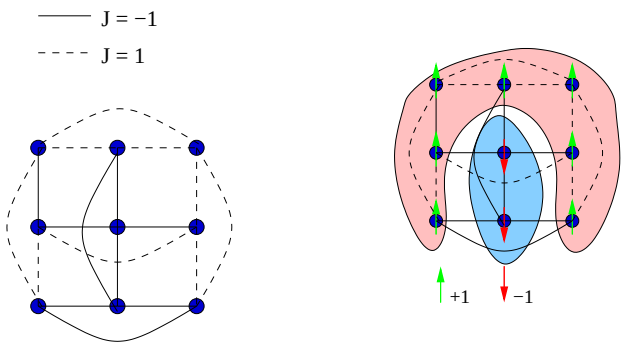

Figure 1: An Ising spin glass instance with interactions and a spin configuration minimizing the energy of the instance

One such NP-complete problem originating from statistical physics is determining the ground state of an Ising spin glass. In experiments, there is a clear transition from the spin glass phase were the magnetization is approximately zero to the ferromagnet phase where almost every spin in the system points to the same direction, resulting in high magnetization. It is assumed, and later shown empirically in this summary, that close to this transition a similar change can be seen in the run time of a modern algorithm when solving the corresponding minimization problem.

This summary is based on two related articles, of which the first describes a Branch-and-Cut algorithm designed for solving difficult Ising spin glass problems [1], and the second presents results on a set of regular graphs [2].

The summary is structured as follows. In Sect. 2 we describe the Ising spin glass, the mapping to a max-cut problem and describe what are Integer Linear Programs (ILPs). In Sect. 3 we describe the branch-and-cut algorithm for max-cut problems and describe certain heuristics developed for Ising spin glasses. In Sect. 4 we describe the results from [2], and finally conclude in Sect. 5.

2 Basics

Given a finite set of vertices $V \subset \mathbb{N}$ and an interaction $J_{ij} \in \mathbb{R}$ between vertices $i, j \in V$, we construct a weighted graph $G = (V, E)$, where there is an edge, or interaction, between two vertices i, j if $J_{ij} \neq 0$. The graph G is called an Ising spin glass. Each vertex i is occupied by a spin $S_i = \pm 1$. Any spin might interact with an external field of strength h, represented by a "ghost spin" S_0, which can be fixed due to symmetry reasons to be $S_0 = +1$. An example of an Ising spin glass (without the external field) is given in Fig. 1.

The problem of determining the ground state of the spin glass G is the problem of placing spins $S_i = \pm 1$ to every vertex i such that the

Hamiltonian of the spin configuration $\vec{S}$

$$\mathcal{H}(\vec{S}) = -\sum_{(i,j)\in E} J_{ij} S_i S_j - h\sum_{i=1}^{n} S_0 S_i \tag{1}$$

is minimized.

A spin configuration $\vec{S}$ partitions the vertices into two sets $V^+ = \{i \in V \mid S_i = +1\}$ and $V^- = \{i \in V \mid S_i = -1\}$. We define the cut $\delta(V^+)$ to be the subset of edges E with one endpoint in the set V^+ and the other endpoint in the set V^-. We can now reformulate the Hamiltonian of the spin configuration $\vec{S}$ as

$$\mathcal{H}(\vec{S}) = -2\sum_{(i,j)\in\delta(V^+)} (-J_{ij}) - \sum_{(i,j)\in E} J_{ij} \tag{2}$$

where the right summation corresponds to assuming that all the spins point to the same direction and the left summation corrects the mistake made in the right summation by substracting twice the interactions between spins pointing to different directions.

To formulate the ground state calculation of an Ising spin glass instance as a max-cut problem in weighted graphs, we use $c_{ij} = -J_{ij}$ as the weight of the edges (i, j) and finally solve the problem of maximizing

$$\sum_{(i,j)\in\delta(V^+)} c_{ij} \tag{3}$$

over all cuts $\delta(V^+)$.

We will formulate this problem as an Integer Linear Program (ILP). The formulation is presented in Sect. 3, but we mention here the definition of an ILP problem.

Definition *The* integer linear programming problem *is the maximization problem*

$$max_{\vec{x}}\{\vec{c}^T\vec{x} \mid A\vec{x} \leq \vec{b}\},$$

where $\vec{c} \in \mathbb{R}^n$ is the cost vector of n elements, $\vec{x}$ is the unknown in the equation, a vector of elements $x_1, \ldots, x_n$, and each x_e is an integer, and $A \in \mathbb{R}^{m\times n}$ is a matrix of $m \times n$ elements corresponding to the constraints of the problem together with the vector $\vec{b} \in \mathbb{R}^m$

The problem of determining whether there exists a vector $\vec{x}$ such that the constraints are satisfied can easily be shown NP-complete by reducing for example 3-SAT to it.

Theorem *ILP is NP-complete*

Proof ILP is in NP, which can be seen by nondeterministically assigning integer values for x_i and verifying (in polynomial time) that each equation holds. We show the NP-completeness by reducing 3-SAT to ILP. Given any clause $l_1 \vee \cdots \vee l_n$ of n literals, the clause can be translated to an ILP by setting $p_1 + \cdots + p_n \leq b$, where $p_i = -x_i$ if $l_i = x_i$, and $p_i = -1 + x_i$

```
function Branch-and-Bound(P)
    A ← {P}                          /* Initialize set of sub-
                                        problems */
    x* ← Any cut of P                /* Initial guess */
    while A ≠ ∅
        C ← An element from A
        A ← A \ {C}
        if c(optimal solution for C) > c(x*)
                                     /* Update optimal value */
            x* ← optimal solution
        else if C has no feasible solution
            break
        else if c(best solution for C) ≤ c(x*)
                                     /* Bound */
            break
        else
            split C into subproblems and add them to A
        end if
    end while
    return x*.
```

Algorithm 1: The Branch-and-Bound algorithm for maximization problems

if $l_i = \neg x_i$, for $x_1, \ldots, x_n \in \{0, 1\}$, and $0 > b > -1$. If we would have an algorithm for solving ILP in polynomial time, the same algorithm could be used for solving 3-SAT, and thus any problem in NP. □

If the integral condition on $\vec{x}$ is lifted, there are efficient algorithms for solving the problem (e.g. the Simplex algorithm).

3 The Branch-and-Cut Algorithm

The Branch-and-Cut algorithm is based on a simpler Branch-and-Bound algorithm given in Algorithm 1. The algorithm takes as input the original optimization problem P, and computes $\vec{x}^*$, the optimum solution for the problem P. It is a general purpose optimization algorithm based on backtrack search.

Given a set $\mathcal{F}$ of all possible cuts, we present the cut as a characteristic vector $\vec{x} = x_{e_1}, \ldots, x_{e_n}$, where n is the number of edges in graph G, $x_{e_i} = 1$ if $e_i \in \delta(V^+)$ and otherwise $x_{e_i} = 0$.

The cost of a cut $F \in \mathcal{F}$ is denoted as $\vec{c}(F) = \sum_{e \in F} c_e = \vec{c}^T \vec{x}$. Now the problem of determining the maximum cut in the graph G is that of determining

$$\max_{\vec{x}} \{\vec{c}^T \vec{x}\},$$

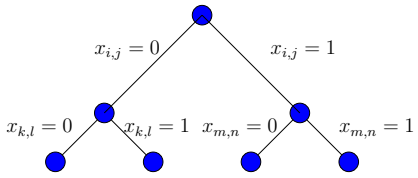

Figure 2: A branch-and-cut search tree

where $\vec{x}$ ranges over characteristic vectors of cuts. We now need to convert the problem of representing cuts as ILP. From graph theory, we have the following

Theorem *Let F be any set of edges in graph G. Now F is a cut if and only if F contains an even number of edges from every cycle in G.*

The necessity of the theorem is clear, since starting a cycle from a vertex $v \in V$, we must cross even number of times the cut in order to return to v. The proof of sufficiency is not obvious, and the reader is referred to the answer of Exercise 4.1.26. in [3].

We can now formulate the max-cut problem as

$$\max\{\vec{c}^T\vec{x} \mid \vec{x} \text{ is integral, } \vec{x}(Q) - \vec{x}(C \setminus Q) \leq |Q| - 1\} \tag{4}$$

where the equations on the right side are called *cycle inequalities*, and are written for every cycle C in graph G and for every odd-cardinality subset Q of C. An equation of this form holds, since if $\vec{x}$ is a cut, C is a cycle and Q is an odd-cardinality subset of the cycle C, the cardinality of the intersection of the cut $\vec{x}$ and the set $C \setminus Q$, $\vec{x}(C \setminus Q)$, must either be empty in which case also $\vec{x}(Q) = 0$ (by odd-cardinality of Q), or it must be at least one.

Since there is in the worst case an exponential number of cycles in a graph, and there are exponentially many subsets Q, the formulation of the cycle inequalities will be worst-case exponential in the number of vertices in G. Consequently, the full set of cycle inequalities for a typical non-trivial instance would not fit in the memory of any computer. Thus, certain relaxation techniques will have to be used.

Since the ILP in (4) cannot be solved in general case using the full set of cycle inequalities, we use a relaxation of the problem where most of the inequalities are left out. Solutions to this problem are not elements of $\mathcal{F}$, but elements of a superset $\mathcal{F}' \supset \mathcal{F}$. By set inclusion, we can guarantee that any maximum solution to that problem must be at least as good as the actual optimum solution to the original problem. This information can be efficiently used in the bound step of branch-and-bound algorithm, resulting in the branch-and-cut algorithm, as follows. Assuming that the search process has found a solution $\vec{x}^*$ corresponding to some real cut somewhere in the branch $x_{i,j} = 0, x_{k,l} = 0$ (see Fig 2). When the algorithm finally returns from this branch it tries the branch $x_{i,j} = 0, x_{k,l} = 1$. Now the branch-and-cut algorithm, working on some subset $\mathcal{F}'$ in this branch, solves the problem by using standard linear equation solving techniques, resulting in a possibly real-valued optimal

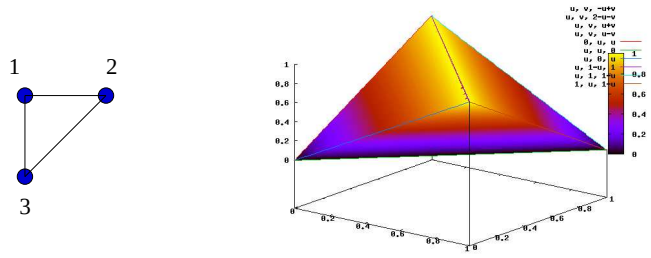

Figure 3: The graph K_3 and the corresponding cycle inequalities

solution for that subproblem. If the cost of this solution is less than that for $\vec{x}^*$, the algorithm can prune this branch $x_{i,j} = 0, x_{k,l} = 1$ without further checking, since every optimum solution below this branch is over a more constrained subset and thus cannot be higher than $\vec{x}^*$.

Finally, the algorithm will have obtained the optimum solution, when the value of the cut $\vec{x}^*$ can be shown to be equal to the upper bound from the relaxation.

Example *As an example of the linear programming description of a cut, we study the cuts over the complete graph K_3, or triangle (in left of Fig. 3). The triangle has two cycles, $C_1 = \emptyset, C_2 = \{(1,2),(2,3),(3,1)\}$. The empty cycle C_1 has no odd-cardinality sub-sets, and no cycle inequalities can be constructed. The cycle C_2 has four odd-cardinality sub-sets, $Q_1 = \{(1,2)\}, Q_2 = \{(2,3)\}, Q_3 = \{(3,1)\}, Q_4 = \{(1,2),(2,3),(3,1)\}$. The corresponding cycle inequalities are*

$$\begin{array}{lll} x_{12} - x_{23} - x_{31} & \leq 0, & \textit{for } Q_1 \\ -x_{12} + x_{23} - x_{31} & \leq 0, & \textit{for } Q_2 \\ -x_{12} - x_{23} + x_{31} & \leq 0, & \textit{for } Q_3 \\ x_{12} + x_{23} + x_{31} & \leq 2, & \textit{for } Q_4 \end{array}$$

The reader can verify that all actual cuts ($\emptyset$, $\{(1,2),(2,3)\}$, $\{(1,2),(3,1)\}$, $\{(2,3),(3,1)\}$) satisfy these equations. The 3-dimensional representation over which the optimization problem is solved is given in right of Fig. 3.

The key problem in branch-and-cut is to find small number of inequalities which are of good quality, in the sense that they cut off significant areas of the search space. This is where good heuristics are necessary. The article [1] presents several domain-specific heuristics, of which some check the current solution for being a legal cut and produce cycle inequalities in the negative case, and others produce cycle inequalities based on the static structure of the problem. The paper gives a polynomial-time complete algorithm for producing cycle inequalities from a violating solution based on shortest path computation in a weighted graph. In addition, the paper describes a fast heuristic algorithm of run time $\mathcal{C}(|E| \log |V|)$ based on graph two-coloring. Furthermore, certain domain-specific cycles can be formed from two-dimensional grids and three-dimensional cubes. It is

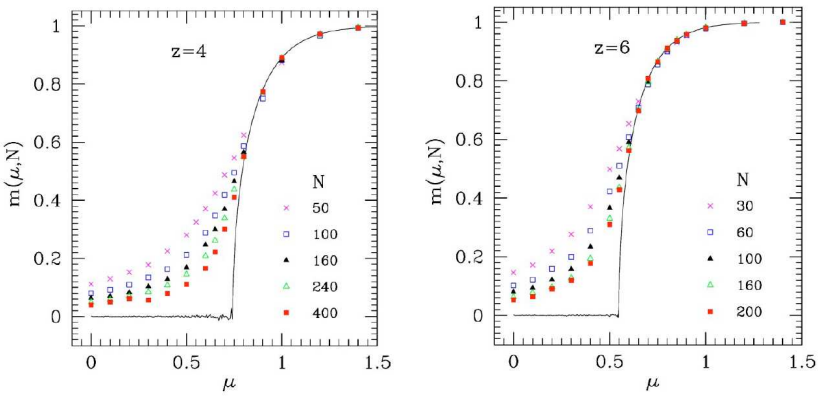

Figure 4: Average magnetization for different system sizes and μ, for $z = 4$-regular and $z = 6$-regular graphs [2]

also possible to use maximum-weight spanning trees where the weights are constructed from the incidence vector of the violating solution, and introduce an additional edge from G to the spanning tree to form a cycle inequality possibly violated by the solution.

4 Experiments

The Ising spin glass instances studied in [2] are constructed by creating a random z-regular graph (in which every vertex has exactly z neighbors), and assigning a random interaction J_{ij} between two neighbors i, j from the Gaussian distribution with mean μ and variance of one. The magnetization of a system of size N with spins S_i on vertices i, defined as

$$m(\mu, N) = \left[\frac{\sum_i S_i}{N} \right]_J \tag{5}$$

is computed for different system sizes N and different means μ.

The results in Fig. 4 show the behaviour of the magnetization as a function of μ, for $z = 4$ and $z = 6$. The solid line in the figures is obtained from Bethe-Peierls approximation by the cavity method for system sizes between 10^3 and 10^5 vertices. The results show the phase transition at $\mu_c = 0.742 \pm 0.005$ for $z = 4$ and $\mu_c = 0.546 \pm 0.005$ for $z = 6$, and suggest that the exact results for similar sized systems follow the behaviour of Bethe-Peierls approximation.

The exact point of phase transition in branch-and-cut results is difficult to see from Fig. 4, but can be determined using the Binder cumulant $g(\mu, N)$ defined using the second and fourth moments of the distribution

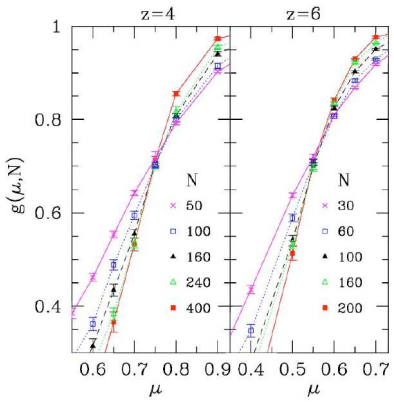

Figure 5: Determining μ_c from Branch-and-Cut results using Binder cumulant [2]

from the magnetization by

$$g(\mu, N) = \frac{1}{2}\left(3 - \frac{\left[\left(\frac{\sum_i S_i}{N}\right)^4\right]_J}{\left[\left(\frac{\sum_i S_i}{N}\right)^2\right]_J^2}\right) \tag{6}$$

The results for different system sizes should intersect at the phase transition point and are presented in Fig. 5 for $z = 4$ and $z = 6$. They show the intersection at $\mu_c = 0.77 \pm 0.02$ and $\mu_c = 0.56 \pm 0.02$ respectively, a result compatible with the results from the cavity approach.

In Fig. 6, the run time of the branch-and-cut algorithm shows a dramatical decrease slightly above μ_c. The spin glass problems seem often difficult, whereas the ferromagnet problems are always easy. One can also see from the picture that the difficulty of solving the spin glass phase increases superpolynomially with regards to the system size, whereas the growth seems linear deep in the ferromagnet phase. This confirms the existence of a "run time phase transition".

5 Conclusions

Determining the ground state or minimum energy-state of an Ising spin glass is a prototypical example from statistical physics, where combinatorial techniques can be used to find an exact solution to a finite problem.

The article [1] describes a method for transforming the problem of determining the ground state of an Ising spin glass to a max-cut problem and presents an efficient algorithm for computing optimum solutions of the max-cut problem by using an elaborate variation of branch-and-cut. The article [2] reports experimental evaluation of the branch-and-cut algorithm on certain spin glass instances, and confirms the existence of a clear phase transition observed also in the laboratory experiments.

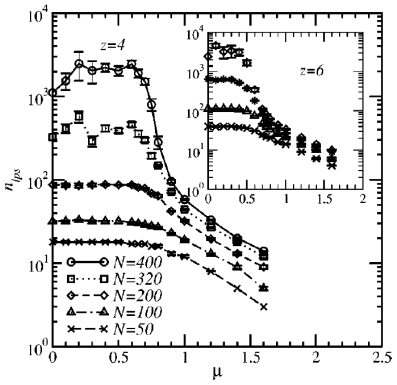

Figure 6: Run time of the branch-and-cut algorithm near μ_c [2]

The algorithm produces results which are compatible with analytical results from nearly infinite-sized systems, showing a clear change in the behaviour of the studied system at the phase transition point. Surprisingly, there is a corresponding change in the behaviour of the run time of the algorithm slightly above the physical phase transition. This is a phenomenon observed in solving of several NP-complete problems and provides certain insight to the question of what makes a problem hard.

References

[1] Liers, F., Jünger, M., Reinelt, G., Rinaldi, G.: Computing Exact Ground States of Hard Ising Spin Glass Problems by Branch-and-Cut. In: New Optimization Algorithms in Physics. (2004) 47–68

[2] Liers, F., Palassini, M., Hartmann, A.K., Jünger, M.: Ground state of the bethe lattice spin glass and running time of an exact optimization algorithm. Physical Review B **68**(9) (2003) 094406

[3] West, D.B.: Introduction to Graph Theory. 2 edn. Prentice Hall (2005)