Alcom-FT Technical Report Series
ALCOMFT-TR-03-95

1 Computing Exact Ground-States of Hard Ising Spin-Glass Problems by Branch-and-Cut

Frauke Liers[1]
Michael Jünger[1]

Institut für Informatik
Universität zu Köln
Germany

Gerhard Reinelt[2]

Institut für Informatik
Ruprecht-Karls-Universität Heidelberg
Germany

Giovanni Rinaldi[3]

Istituto di Analisi dei Sistemi ed Informatica "Antonio Ruberti"
CNR
Italy

1.1 Introduction

When studying ground-state properties of Ising spin-glasses, the computation of exact ground-states, i.e., spin configurations with exact rather than approximate minimum energy, is often desirable. The Ising model is one of the most commonly used, both for its simplicity and its accuracy in describing many real world systems. When we refer to a "spin-glass" in this chapter, we always have the Ising model in mind.

The configurations that are mostly considered in the literature are the two- and three-dimensional Ising spin-glasses on a grid with nearest neighbor interactions and free or periodic boundary interactions, with or without an external magnetic field. The periodic boundary conditions are a standard way of approximating an infinite spin-glass with a structure that contains only a finite number of spins.

Unfortunately, there are no functions in closed form that, given all interactions between the spins and the external field, yield a ground state. Therefore, the only way to compute a

[1]Pohligstrasse 1, D-50969, Köln
[2]Im Neuenheimer Feld 368, D-69120 Heidelberg
[3]viale Manzoni 30, I-00185, Roma

ground-state is using a numerical algorithm. Since the total number of states for a structure with n spins is 2^n, as soon as n exceeds, say, 35, it is impossible, from a computational point of view, to find a ground-state by *brute force*, i.e., by enumerating all possible states and computing the energy for each of them.

A fundamental question from both a practical and a theoretical point of view in computer science is to determine whether it is possible to design an algorithm that, for any possible choice of the spin interactions and of the magnetic field, finds a ground-state in a number of elementary operations bounded by a polynomial function of n, or, more precisely, of the number of bits needed to store the problem data. A problem for which this is possible is called *polynomially solvable* and the procedure used is called a *polynomial algorithm*.

The theory of computational complexity has given quite negative results for the possibility of treating certain Ising spin-glass problems. Most cases are NP-hard. They are therefore generally considered as notoriously hard, and it is believed that there can be no polynomial algorithm for their solution.

Due to the difficulty of the problem, popular methods described in the literature compute an approximation of the value of the minimum energy of the spin-glass. Such methods usually use Monte Carlo simulation including simulated annealing, evolutionary and genetic algorithms. More recent algorithmic approaches are described in [19]. Despite the practical usefulness and efficiency of the heuristic methods used in the literature, two main drawbacks arise:

- It is not possible to estimate how far the produced solution, when the algorithm stops, is away from a real ground-state. Therefore it is not possible to determine reliably the degree of accuracy of the experimental results produced with these methods. For example, if we want to compute the expected ground-state energy as a function of the grid size, using heuristic methods, the values will always have a positive bias, no matter how large the grid size is, how many experiments are made, and how much computation time is invested.

- Two different states with almost the same energy may be very different. Therefore, it is not clear whether the state produced by these algorithms yields useful information on the structure of the ground-state; one has to keep this in mind when *real* ground states have to been analyzed.

In the seventies and early eighties, the NP-hardness of the ground state problem was believed to make its exact solution practically impossible, but the introduction of branch-and-cut techniques for NP-hard combinatorial optimization problems has made such computations practical. Nowadays, we can treat system sizes that allow for meaningful statistics in theoretical physics.

In this chapter we explain the branch-and-cut technique for exact ground-state computation and show its potential for the study of ground-state properties of Ising spin-glasses.

In Section 1.2 we show that the ground-state problem for Ising spin-glasses is equivalent to the max-cut problem that is well-known in combinatorial optimization and has a number of further applications. From then on, we can concentrate on properties of and algorithmic approaches to max-cut. We start by introducing a general scheme for solving hard max-cut problems in Section 1.3 which consists of specializing the well-known branch-and-bound method to max-cut. In branch-and-bound, relaxations play a crucial role, and as a prerequisite for the branch-and-cut algorithm we are aiming at, we study linear programming relaxations of max-cut in some detail in Section 1.4. After these preparations, we outline in Section 1.5 the main components of our branch-and-cut method. In Section 1.6 we present some recent results obtained with exact ground-state computations in the context of studying the nature of the spin-glass phase. Then we comment on advantages of branch-and-cut for exact ground-state computations in Section 1.7. We close in Section 1.8 with some challenges we see for the future.

1.2 Ground-States and Maximum Cuts

Finding the ground-state of a spin-glass is closely related to a well known problem in combinatorial optimization: the maximum cut problem in a weighted graph (max-cut problem for short).

The max-cut problem is the following. We are given a graph $G = (V, E)$ with weights $c_{ij} \in \mathbb{R}$ for all edges $ij \in E$. For each (possibly empty) subset W of the node set V, the *cut* $\delta(W)$ in G is the set of all its edges with one endpoint in W and the other in $V \setminus W := \{i \in V \mid i \notin W\}$. W and $V \setminus W$ are called the *shores* of the cut $\delta(W) = \delta(V \setminus W)$. The *weight of a cut* is given by the sum of the weights of all its edges. The *max-cut problem* is to find a cut of G with maximum weight.

Assume that we have a spin-glass consisting of n spins $S_1, S_2, \ldots, S_n$ where the variables S_i take values $+1$ or -1 and an external magnetic field of strength h.

Given a spin configuration ω, the Hamiltonian of this system is

$$H(\omega) = -\sum_{(ij)} J_{ij} S_i S_j - h \sum_{i=1}^{n} S_i, \tag{1.1}$$

where the sum $\sum_{(ij)}$ is over the coupled spins.

We identify the spins with the node set $V = \{1, \ldots, n\}$ of a graph $G = (V, E)$, the interaction graph associated with the system. For a pair i, j of nodes, G contains an edge ij if the interaction J_{ij} between two magnetic impurities is nonzero. We introduce an extra node "0" with 'ghost spin' S_0 for the external magnetic field that is connected to all other nodes i by an edge $0i$ of weight h. We rewrite

$$H(\omega) = -\sum_{ij \in E} J_{ij} S_i S_j - \sum_{i=1}^{n} h S_0 S_i. \tag{1.2}$$

Observe that each spin configuration ω induces a partition of the node set V of the interaction graph G into node sets V^+ and V^-, where $V^+ := \{i \in V \mid S_i = +1\}$ and $V^- = \{i \in V \mid S_i = -1\}$. So the energy of the spin configuration ω can be translated to the form

$$H(\omega) = -2 \sum_{ij \in \delta(V^+)} c_{ij} - C, \tag{1.3}$$

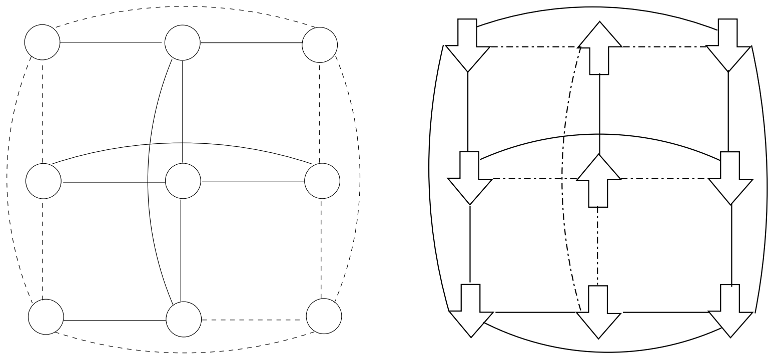
where $c_{ij} := -J_{ij}$ for all $ij \in E$ and $C := \sum_{ij \in E} J_{ij}$. Hence, the problem of minimizing H is equivalent to maximizing

$$c(\delta(V^+)) := \sum_{ij \in \delta(V^+)} c_{ij} \tag{1.4}$$

over all $V^+ \subseteq V$.

This problem is a max-cut problem in the interaction graph G associated with the spin-glass system. Thus, finding a ground-state in the Ising model of a spin-glass is equivalent to finding an optimum solution of the corresponding max-cut problem. The structure of the underlying graph and the type of the objective function determine the complexity of the problem.

In Fig. 1.1 we show an instance on a 3×3 grid with periodic boundary conditions, ± 1 interactions and no external field. Fig. 1.1(a) shows the max-cut problem. The solid lines have edge weight 1 (i.e., the coupling strength in the spin-glass instance is -1), the dashed lines weight -1. Fig. 1.1(b) shows an optimum solution. The dash-dotted lines correspond to the edges of the cut.



(a) A 3×3 instance. The solid lines have weight 1, the dashed lines weight -1.

(b) An optimal solution. The dash-dotted lines correspond to the cut edges.

Figure 1.1: Example for a 3×3 instance.

The max-cut problem is NP-hard in general, but is polynomially solvable for some classes of graphs like, e.g., planar graphs, graphs not contractable to K_5 (the complete graph on five nodes), weakly bipartite graphs and graphs with nonnegative edge weights. E.g., the standard 2-dimensional grid model with nearest neighbor interactions, no periodic boundary conditions and no magnetic field amounts to solving a max-cut problem in a planar graph, and is therefore polynomially solvable. Surprisingly, max-cut is already NP-hard for almost planar graphs [3], i.e., graphs where only one node has to be removed to obtain a planar graph. There are quite negative results for the possibility of solving the Ising spin-glass with periodic boundary conditions. Polynomially solvable cases are the two dimensional grid without a field and $\pm J$ interactions [30], and more generally the case in which the genus of the graph is bounded by a constant and the sizes of the integral edge weights are bounded in absolute value by a polynomial in the size of the graph [16]. For the Gaussian case the question is still open. As soon as we have an external field, the problem becomes NP-hard for all kinds of spin interactions [2].

Goemans and Williamson [20] presented a 0.878-approximation algorithm for the maximum cut problem, i.e., an algorithm with running time bounded by a polynomial in the input size that provably delivers a solution of at least 0.878 times the optimum value of a maximum cut. However, the bad news is that under the assumption P$\neq$NP there is no polynomial algorithm that provably delivers a solution of at least 98% of the optimum value of a maximum cut [10]. In the following we only deal with optimum solutions of the maximum cut problem.

The algorithmic approach in this chapter is applicable to any instance of the max-cut problem and therefore to other combinatorial optimization problems that, like the ground-state problem, can quite easily be transformed to max-cut. Here are some examples:

- A generic optimization problem whose equivalence to max-cut has been shown by Hammer, see [11] is *quadratic 0-1 optimization*, in which we are given a matrix $A \in \mathbb{R}^{n\times n}$ and a vector $a \in \mathbb{R}^n$ and wish to solve $\min\{x^T Ax + a^T x \mid x \in \{0,1\}^n\}$, see [7] for details.

- In [6] an application to the layout of electronic circuits is described: when the wiring occurs on two layers, the task is to assign each wire segment to one of these layers so as to minimize the number of *vias*, i.e., layer changes along the wires.

- An application to the vertical or horizontal polarization assignment to television broadcast antennas is described in [1]: The optimization goal is to minimize interference between the transmitters.

- In a tournament schedule for a sports league, a *break* consists of two consecutive "home" or two consecutive "away" games for an individual team. Breaks are considered undesirable. A reduction of the break minimization problem to max-cut is described in [14] along with a computational study.

1.3 A General Scheme for Solving Hard Max-Cut Problems

The most common algorithmic framework for solving NP-hard combinatorial optimization problems to optimality is the branch-and-bound scheme that has been introduced by [25] in the context of the general integer linear programming problem

$$\max\{c^T x \mid Ax \leq b,\, x \geq 0,\, x \text{ integer}\} \tag{1.5}$$

where $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$. We describe this framework here in the context of max-cut while keeping in mind that it is applicable in a much broader sense. Understanding branch-and-bound and some properties of the set of cuts in graphs is a prerequisite for the branch-and-cut algorithm we are aiming at.

Recall that max-cut is defined on an undirected graph $G = (V, E)$ with edge weights c_e for all $e \in E$. For a subset $S \subseteq E$ and a vector $a \in \mathbb{R}^E$, let $a(S)$ be a shorthand for $\sum_{e \in S} a_e$, where a_e is the e-th component of the vector a. Denote by $\mathcal{F}$ the set of all cuts in G. Then the max-cut problem becomes

$$\max\{c(F) \mid F \in \mathcal{F}\}. \tag{1.6}$$

For the subsequent discussion it is convenient to replace the cuts $F \in \mathcal{F}$, i.e. the *f*easible solutions by their characteristic vectors in $\{0,1\}^E$: With any subset $S \subseteq E$ we associate its characteristic vector $\chi^S \in \{0,1\}^E$ defined by

$$\chi_e^S = \begin{cases} 0 & \text{if } e \notin S, \\ 1 & \text{if } e \in S. \end{cases} \tag{1.7}$$

If we let $\mathcal{C} = \{x \in \{0,1\}^E \mid x = \chi^F \text{ for some } F \in \mathcal{F}\}$, then (1.6) can equivalently be written as

$$\max\{c^T x \mid x \in \mathcal{C}\}. \tag{1.8}$$

When $\mathcal{C}$ is replaced by a superset $\mathcal{C}_R \supseteq \mathcal{C}$, the solution value of

$$\max\{c^T x \mid x \in \mathcal{C}_R\} \tag{1.9}$$

is an upper bound for the solution value of (1.8), i.e., an upper bound for the weight of a maximum cut in G. Problem (1.9) is a *relaxation* of problem (1.4), and a crucial ingredient of branch-and-bound algorithms is to find relaxations that are computationally "easy" to solve yet give "good" bounds.

Before we look at suitable relaxations of max-cut, we outline the generic branch-and-bound scheme that is a simple variant of the generic divide-and-conquer principle. Namely, if the problem cannot be solved directly, it is split into (hopefully easier) subproblems. These subproblems are then either solved or split further until eventually very simple problems are generated. From the solution of all the generated subproblems a solution of the original problem can then be constructed.

algorithm branch-and-bound

begin

1. Initialize the list of active subproblems with the original problem.
2. **while** list of active subproblems is not empty **do**
   begin
3. choose some subproblem from the list of active problems and "solve" it as follows:
   (a) find an optimal solution for the subproblem, or
   (b) prove that the subproblem has no feasible solution, or

(c) prove using a relaxation that there is no feasible solution for the subproblem that has a higher objective function value than the best feasible solution that is already known, or

(d) split the subproblem into further subproblems and add them to the list of active problems, if none of the above is possible.

end

4. **return** the best feasible solution found so far as an optimal solution.

end

The splitting of problems into subproblems can be represented by the so-called *branch-and-bound tree*, the root of which represents the original problem. It is crucial for the efficiency of a branch-and-bound algorithm that this tree does not grow too large. Therefore subproblems have to be solved if possible by alternatives (a), (b) or (c) of step 3. Alternative (a) rarely occurs, for (b) and (c) relaxations are important. Namely for (b), if a relaxation of the subproblem is already infeasible, then also the subproblem itself is infeasible. To be able to finish the subproblem in (c), good lower and upper bounds must be available. Lower bounds are obtained by finding feasible solutions. These are either obtained by solving some subproblem to optimality or usually by determining good feasible solutions using heuristics. Upper bounds can be computed by using relaxations where in principle any type of relaxation discussed above can be employed. It is clear that the tighter a relaxation the better the performance of the algorithm will be. Without a suitable relaxation branch-and-bound would tend to completely enumerate the set of feasible solutions and thus become computationally infeasible. Of prime interest are linear programming relaxations that we will explain in the next section.

The easiest way to implement the splitting step 3(d) is to select a variable x_e and replace the current subproblem by two, in one of which the additional restriction $x_e = 0$ and in the other the additional restriction $x_e = 1$ is appended.

In the special context of max-cut, we can exploit the fact that the set of all cuts in a graph G is closed under taking symmetric differences. In other words, the symmetric difference of the cuts $F \in \mathcal{F}$ and $F' \in \mathcal{F}$ in G, denoted by $F \triangle F'$ (i.e., the set of edges that belong to one of the two cuts but not to both) is a cut in G. This simple combinatorial concept can be translated into algebraic terms and applied to the characteristic vectors of cuts. Let χ^F and $\chi^{F'}$ be the characteristic vectors of F and F', respectively. Then the map

$$s^F : \mathbb{R}^E \longrightarrow \mathbb{R}^E \tag{1.10}$$

called *switching along the cut* F and defined by

$$s^F(\chi^{F'})|_e = \begin{cases} \chi_e^{F'} & \text{if } e \notin F, \\ 1 - \chi_e^{F'} & \text{if } e \in F. \end{cases} \tag{1.11}$$

maps the characteristic vector of any cut F' in G into the characteristic vector of another cut in G. It maps χ^F to the 0 vector (the characteristic vector of the empty cut) and it is an automorphism of $\mathcal{F}$.

Given an instance of the max-cut problem defined by a graph G and an objective function vector c, for a fixed cut F the map $s^F(\cdot)$ provides a variable transformation $z = s^F(x)$ that translates the problem

$$\max\{c^T x \mid x \in \mathcal{C}\} \tag{1.12}$$

into the problem

$$\max\{c'^T z + d \mid z \in \mathcal{C}\} \tag{1.13}$$

where c' is defined by

$$c'_e = \begin{cases} c_e & \text{if } e \notin F \\ -c_e & \text{if } e \in F \end{cases} \tag{1.14}$$

and $d = c\chi^F$.

In addition to the switching operation, we need the concept of contraction of two nodes u and v in G. This operation consists of identifying u and v with a new node, deleting the edge $e = uv$, if $e \in E$, and making all edges incident with either u or v in G incident with the new node. Finally, double edges arising in this process are replaced by a single edge whose weight is the sum of the weights of the two edges, see Fig 1.2 for an illustration.

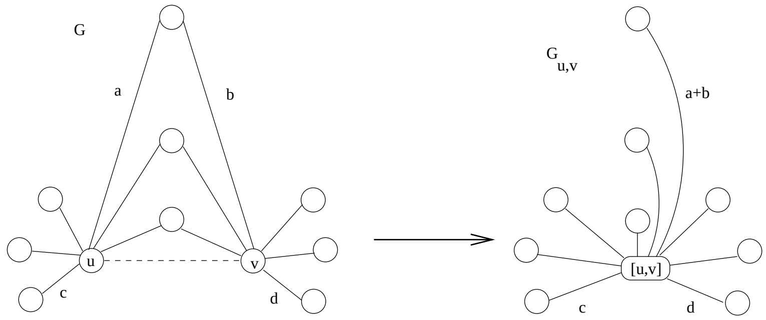


Figure 1.2: Contracting nodes u and v

In max-cut we can contract a pair of nodes u and v whenever we want to make sure that u and v belong to the same shore of the cut, thus reducing the number of edges and therefore the dimension of the space in which the optimization takes place.

Based on the switching and contraction operations, we can implement a branch-and-bound scheme for max-cut as follows:

Let us assume that we have a relaxation

$$\max\{c^T x \mid x \in X \subseteq [0,1]^E\} \tag{1.15}$$

of max-cut defined by the set X that contains all characteristic vectors of cuts in G and has the additional property that every integral vector of X is the characteristic vector of some cut in G, and that (1.15) is "easy" to solve. (We shall see later that such relaxations are readily available.)

Now let $\bar{x}$ be the characteristic vector of a cut in G. $\bar{x}$ serves as the incumbent solution: it corresponds to the best cut we know at any time during the execution of the branch-and-bound algorithm. Let x^* be an optimum solution of (1.15). If $c^T x^* < c^T \bar{x}$, we have case 3(c) and the subproblem is *fathomed*, i.e., needs no further consideration. If x^* is integral, it must be the characteristic vector of a cut, and we are again in case 3(c) after we have updated $\bar{x} \longleftarrow x^*$ in case $c^T x^* > c^T \bar{x}$, so we can consider the subproblem fathomed. Finally, we implement case 3(d) by choosing an edge $e = uv$ such that $0 < x_e^* < 1$, and replace the current subproblem by the following pair of subproblems:

(S1) The first subproblem is obtained by contracting u and v. This corresponds to forcing x_e to 0.

(S2) The second subproblem is obtained by switching x^* by $\delta(u)$, i.e., performing the variable transformation $s^{\delta(u)}(x^*)$ and recording this by multiplying all edge weights c_e for $e \in \delta(u)$ by -1, recording the additive constant $c\chi^{\delta(u)}$ for the objective function value, and finally contracting u and v. Contracting u and v amounts to forcing x_e to be equal to 0 in the transformed subproblem (after switching), which by switching corresponds to forcing x_e to 1 in the original subproblem (before switching).

Subproblems (S1) and (S2) are two new instances of max-cut. Therefore, any algorithm used to solve the original problem can be recursively applied to each subproblem.

There are various relaxations of max-cut that can be used in either implementation of the above branch-and-bound scheme, for an overview, see [22]. Here we want to concentrate on linear programming relaxations because their application has proven to be by far superior in the context of ground-state computations for spin-glasses with short-range interactions like in grids or similar structures.

1.4 Linear Programming Relaxations of Max-Cut

Before we introduce linear programming (LP) relaxations of max-cut, we derive an integer linear programming formulation of the form (1.5) for max-cut.

An edge set $C = \{v_0v_1, v_1v_2, \ldots, v_{k-1}v_0\} \subseteq E$ is called a *cycle* (of length k) in G. An edge $v_iv_j \in E$ where $j \neq (i+1) \bmod k$ is called a *chord* of the cycle C. The integer linear programming formulation is based on the observation that the intersection of a cut and a cycle in a graph G always contains an even number of edges and that this property characterizes those subsets of the edge set E that correspond to cuts. Using characteristic vectors of edge sets, this condition can be translated into algebraic terms by stipulating the *cycle inequality* $\chi(Q) - \chi(C \setminus Q) \leq |Q| - 1$ for all cycles C in G and each odd cardinality subset Q of C. (Recall that $\chi(S) = \sum_{e \in S} \chi_e$ for $S \subseteq E$.) For example, if C is a cycle consisting of an odd number of edges, the inequality for $Q = C$ says that any cut can contain at most $|C| - 1$ of the edges in C. If C is an arbitrary cycle and Q any odd subset of edges with $Q \neq C$, the inequality says that if a cut contains all edges of Q, i.e., $\chi(Q) = |Q|$, it must also contain at least one edge in $C \setminus Q$, i.e. $\chi(C \setminus Q) \geq 1$.

This leads to the following formulation of max-cut as an integer linear program:

$$\begin{aligned}\max\{c^T x \mid x(Q) - x(C \setminus Q) \leq |Q| - 1 &\text{ for each } Q \subseteq C,\ |Q| \text{ odd },\\ &\text{ for each cycle } C \text{ in } G,\\ 0 \leq x_e \leq 1 &\text{ for each } e \in E,\\ x_e &\text{ integer for each } e \in E\}\end{aligned} \tag{1.16}$$

Our first linear programming relaxation for max-cut is obtained by dropping the integrality conditions in (1.16):

$$\begin{aligned}\max\{c^T x \mid x(Q) - x(C \setminus Q) \leq |Q| - 1 &\text{ for each } Q \subseteq C,\ |Q| \text{ odd },\\ &\text{ for each cycle } C \text{ in } G,\\ 0 \leq x_e \leq 1 &\text{ for each } e \in E\}\end{aligned} \tag{1.17}$$

Obviously (1.17) posesses all features we required in the discussion of the previous section except that it is not immediate that it can be "easily" solved. After all, (1.17) has an exponential number of inequalities, thus simply feeding it into computer memory is out of the question. Nevertheless, (1.17) can indeed be solved practically efficiently in polynomial time, as we shall see later in this section. We call (1.17) the *cycle relaxation* of max-cut. As the feasible solutions of the cycle relaxation are the solutions of a system of linear inequalities, and they are all contained in the $|E|$-dimensional hypercube, they define a polytope which we call the *cycle polytope*

$$\begin{aligned}P^G_{\mathrm{CYCLE}} = \{x \in \mathbb{R}^E \mid x(Q) - x(C \setminus Q) \leq |Q| - 1 &\text{ for each } Q \subseteq C,\ |Q| \text{ odd },\\ &\text{ for each cycle } C \text{ in } G,\\ 0 \leq x_e \leq 1 &\text{ for each } e \in E\}.\end{aligned} \tag{1.18}$$

If we take the convex hull of all characteristic vectors of cuts, or equivalently, all integer points in the cycle polytope, we obtain the *cut polytope*

$$\begin{aligned}P^G_{\mathrm{CUT}} &= \mathrm{conv}\{\chi^F \mid F \in \mathcal{F}\}\\ &= \mathrm{conv}\{x \in P^G_{\mathrm{CYCLE}} \mid x \in \{0,1\}^E\}.\end{aligned} \tag{1.19}$$

The vertices of the cut polytope are the characteristic vectors of the cuts in G, therefore max-cut can be formulated as

$$\max\{c^T x \mid x \in P^G_{\mathrm{CUT}}\}. \tag{1.20}$$

By classical theorems of Weyl and Minkowski we know that there is a matrix A and a vector b such that

$$P^G_{\mathrm{CUT}} = \{x \in \mathbb{R}^E \mid Ax \leq b,\ x \geq 0\}. \tag{1.21}$$

However, complexity theory tells us that such an inequality system is enormously large and that its explicit computation is hopeless in practice.

Let us consider the triangle graph as shown in Fig. 1.3 as the smallest interesting example.

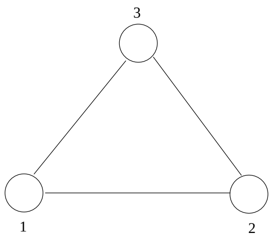


Figure 1.3: A graph consisting of a triangle.

The set of characteristic vectors of cuts is

$$\left\{ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \right\}, \tag{1.22}$$

where we order the coordinates as (x_{12}, x_{23}, x_{13}).

The cycle inequalities read

$$\begin{aligned} x_{12} + x_{23} + x_{13} &\leq 2 \\ x_{12} - x_{23} - x_{13} &\leq 0 \\ -x_{12} + x_{23} - x_{13} &\leq 0 \\ -x_{12} - x_{23} + x_{13} &\leq 0 \end{aligned} \tag{1.23}$$

The reader may easily verify that the vectors in (1.22) satisfy all four inequalities of (1.23) while other possible integer vectors violate at least one of them. E.g. the vector $(1, 1, 1)$ (not corresponding to a cut) violates the first inequality in (1.23). In Fig. 1.4 we show the cut polytope $P^{K_3}_{\mathrm{CUT}}$, i.e., the convex hull of the cut vectors (1.22) for the triangle graph of Fig. 1.3. The cycle inequalities (1.23) are hyperplanes in $\mathbb{R}^3$, and it is not hard to see that each of them corresponds to a face of the tetrahedron in Fig 1.4. Hence, $P^{K_3}_{\mathrm{CUT}} = P^{K_3}_{\mathrm{CYCLE}}$.

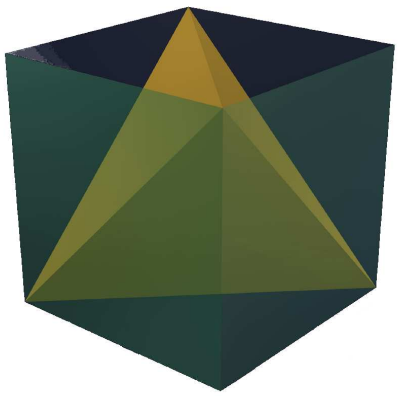


Figure 1.4: The cut polytope $P^{K_3}_{\mathrm{CUT}}$ in the cube $[0, 1]^3$

We shall see that even partial descriptions of P^G_{CUT} by linear inequalities lead to relaxations of max-cut such as (1.17) that are useful in practical computations.

If $P \subseteq \mathbb{R}^n$ is a polytope, we say that the inequality $a^T x \leq a_0$ is *valid* for P if $P \subseteq \{x \in \mathbb{R}^n \mid a^T x \leq a_0\}$. For any valid inequality $a^T x \leq a_0$ for P, the polytope $\{x \in P \mid a^T x = a_0\}$ is called a *face* of P. Faces of P different from P itself are called *proper faces* of P. The proper faces of minimum dimension 0 are the vertices of P, and the proper faces of maximum dimension $\dim P - 1$ are called *facets* of P.

It can be shown that $\dim P^G_{\mathrm{CUT}} = |E|$, therefore the facets of P^G_{CUT} are the $(|E| - 1)$-dimensional proper faces of P^G_{CUT}. The most compact description of a full-dimensional polytope consists of facet defining inequalities only, one for each facet. It turns out that the cycle inequalities define facets of P^G_{CUT} whenever the defining cycle C has no chord. A trivial inequality $x_e \geq 0$ or $x_e \leq 1$ defines a facet of P^G_{CUT} when e is not contained in a triangle of G. All non-facet defining inequalities can be eliminated from the linear description of P^G_{CYCLE} and P^G_{CUT}. Therefore, when G is the complete graph K_p, of all cycle and trivial inequalities, only the inequalities

$$\begin{aligned} x_{ij} + x_{ik} + x_{jk} &\leq 2 \\ x_{ij} - x_{ik} - x_{jk} &\leq 0 \\ -x_{ij} + x_{ik} - x_{jk} &\leq 0 \\ -x_{ij} - x_{ik} + x_{jk} &\leq 0 \end{aligned} \tag{1.24}$$

remain.

In the following we first introduce several classes of facets for the cut polytope. Later we concentrate on the cycle inequalities as they turn out to be especially important for practical spin-glass ground-state computations.

A graph is called a *bicycle p-wheel* if it consists of a cycle of length p and two nodes adjacent to each other and to every node of the cycle. Let (W, B) be a bicycle $(2k+1)$-wheel, $k \geq 1$, contained in G. Then the inequality

$$x(B) \leq 2(2k+1) \tag{1.25}$$

defines a facet of P^G_{CUT}, [9]. See Figure 1.5(a) for a bicycle 5-wheel together with a cut of maximum cardinality satisfying the inequality with equality.

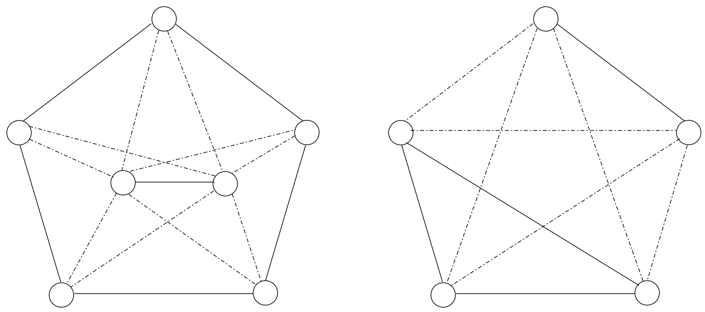
Let $K_p = (W, E_p)$ be a complete subgraph of order p of G. Then the *K_p-inequality*

$$x(E_p) \leq \left\lceil \frac{p}{2} \right\rceil \left\lfloor \frac{p}{2} \right\rfloor \tag{1.26}$$

is valid for P^G_{CUT}, see Figure 1.5(b); this inequality defines a facet of P^G_{CUT} if and only if p is odd [9].

The polytope P^G_{CUT} has been studied by Barahona and Mahjoub [9] and there are further classes of facets known. In particular there exist interesting methods to construct new facet-defining inequalities from given facet-defining inequalities.

The usage of any of the above mentioned and further facet defining inequalities tightens the relaxation. In practice, already the relaxation obtained by optimizing over the cycle polytope has proven useful, so let us first concentrate on the cycle polytope. At the beginning of this



(a) Bicycle $2(k+1)$-wheel with $k = 2$. A cut of maximum cardinality cuts all "spoke" edges.

(b) K_5. The dash-dotted edges correspond to a cut of maximum cardinality.

Figure 1.5: Valid inequalities for the cut polytope.

section, we claimed that we can efficiently solve the cycle relaxation even though it is defined by an exponential number of inequalities.

The solution is to start with a small subset of the constraint set, solve the LP to optimality, and check if the optimum solution satisfies the constraints that have not been taken into account. If this is the case, then the computed solution is optimal for the complete problem. Otherwise, there are constraints that are violated. One or some of them are added to the current LP, the LP is reoptimized, and the process continues until all constraints are satisfied. So the core problem that has to be solved here is the *separation problem*: Given some x^* and a set of constraints $Ax \leq b$, verify that $Ax^* \leq b$ or deliver an inequality $a^T x \leq a_0$ of the system such that $a^T x^* > a_0$. We describe below how this can be done for special relaxations that are of interest here.

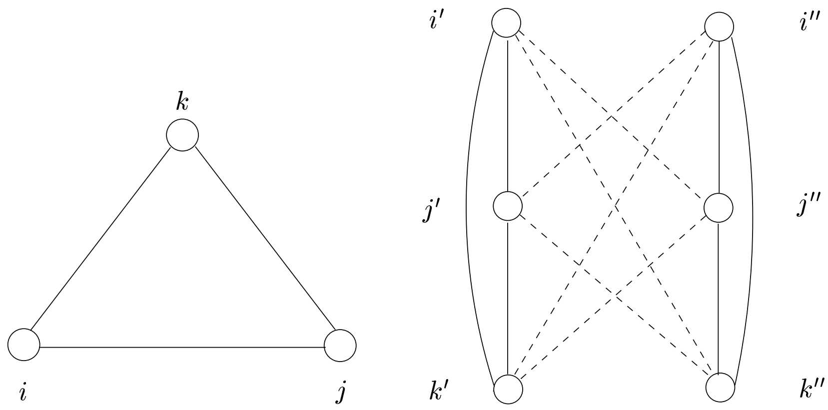
A special case of a celebrated result of Grötschel, Lovász, and Schrijver [18] says that we can optimize a linear objective function over a polytope P in polynomial time if and only if we can solve the separation problem for P in polynomial time.

This approach is termed *cutting-plane approach* due to the fact that the constraints added to the current LP "cut off" the current solution because it is infeasible for the original problem. It is important that the approach does not require an explicit list of the constraints defining the original problem. Required is "only" a method for identifying inequalities that are violated by the current solution.

We show that the cycle inequalities can be separated in polynomial time implying that the cycle relaxation of the max-cut problem can be solved in polynomial time. The algorithm we are going to describe is by Barahona and Mahjoub [9].

Let y, $0 \leq y_e \leq 1$ for all $e \in E$, be the point to be separated. We define a new graph $H = (V' \cup V'', E' \cup E'' \cup E''') = (W, F)$ which consists of two copies of G, say $G' = (V', E')$ and $G'' = (V'', E'')$, and the following additional edges E''': For each edge $uv \in E$ we create the two edges $u'v''$ and $u''v'$, where $u' \in V'$, $u'' \in V''$ denote the two copies of a node $u \in V$. The edges $u'v' \in E'$ and $u''v'' \in E''$ are assigned the weight y_{uv}, while the edges

$u'v''$, $u''v' \in E'''$ are assigned the weight $1 - y_{uv}$. In Fig. 1.6 we show a graph consisting of a triangle and the corresponding doubled graph H.



(a) A graph consisting of a triangle i, j, k.

(b) The corresponding graph H. The bold edges u', v', u'', v'' have weights x_{uv}, the dashed edges $u'v''$ and $u''v'$ have weights $1.0 - x_{uv}$

Figure 1.6: Separation of the cycle inequalities.

For each pair of nodes $u', u'' \in W$ we calculate a shortest (with respect to the weights just defined) path in H. Such a path contains an odd number of edges of E''' and corresponds to a closed walk in G containing u. Clearly, if the shortest of these (u', u'')-paths in H has length less than 1, then there exists a cycle $C \subseteq E$ and an edge set $Q \subseteq C$, $|Q|$ odd, such that y violates the corresponding odd cycle inequality. (C and Q are easily constructed from a shortest path.) If the shortest of these (u', u'')-paths has length at least 1, then y satisfies all these inequalities.

Since shortest path computations can be efficiently performed in polynomial time, we have described a polynomial time separation algorithm for cycle inequalities, and therefore, the cycle relaxation can be solved in polynomial time.

Gerards [17] has shown that also bicycle wheels can be separated in polynomial time, but no polynomial time separation algorithm is known for the separation of K_p-inequalities.

A cutting-plane procedure requires the solution of lots of linear programs, most of which arise by appending further inequalities to a linear program solved previously.

A linear program can be considered an easy problem. From a theoretical point of view there are polynomial time algorithms (ellipsoid method, interior point methods) for its solution. But also from a practical point of view there are effective algorithms (simplex algorithms, interior point algorithms) which are able to solve very large problem instances with several ten-thousands of constraints and millions of variables.

For in depth treatments of linear programming techniques, see the textbooks by Chvátal [12] and Padberg [26]. In the context of cutting-plane procedures, the preferred linear programming solver is the so-called *dual simplex method* that is not a polynomial time method but whose advantages are its practical efficiency and, in particular, the elegant and efficient re-optimization of a linear program after the addition of further inequalities.

The result of Grötschel, Lovász, and Schrijver is based on the ellipsoid method that solves linear programs in polynomial time yet does not perform well in practice for numerical reasons. By using the simplex method, we trade theoretical for practical efficiency.

1.5 Branch-and-Cut

We can now describe our approach for solving hard spin-glass problems. The basic idea of the algorithm is to implement the bounding part in branch-and-bound by a cutting-plane algorithm for linear programming relaxations of max-cut as described in the previous section. Efficient separation is crucial for success. Therefore, we first describe how we separate cycle inequalities.

In the basic algorithm for cycle inequality separation described in the previous section, we can use Dijkstra's well-known algorithm for the shortest path computations and this results in an implementation with $O(|V|^3)$ running time.

For practical purposes this is rather slow. Therefore, we have added faster heuristics for finding violated cycle inequalities in order to avoid calling the "exact separation routine" as much as possible. We describe them in the order we call them in the algorithm.

Suppose $y = (y_1 \dots y_{|E|})$ is the optimum solution of the last linear program. We have to check whether y is the characteristic vector of a cut, and if not, find odd cycle inequalities violated by y, if there are any.

For $0 \leq \varepsilon \leq \frac{1}{2}$ we define the graph $G_\varepsilon = (V, E_\varepsilon)$ as follows:

$$E_\varepsilon := \{e \in E \mid y_e \leq \varepsilon \text{ or } y_e \geq 1 - \varepsilon\}. \tag{1.27}$$

We try to 2-color the nodes of G_ε with red and green, say. First we pick an arbitrary node $v \in V$ and color it red. For all neighbors w of v in G_ε we do the following: If w is not colored, w receives the color of v if $y_{vw} \leq \varepsilon$, otherwise w receives the complementary color. If w is already colored, there are two cases. If w has the same color as v and $y_{vw} \leq \varepsilon$ or if v and w have complementary colors and $y_{vw} \geq 1 - \varepsilon$, we continue. Otherwise we have found a cycle C with an odd number of edges of value at least $1 - \varepsilon$. Let Q be the set of these edges. We check whether $y(Q) - y(C \setminus Q) > |Q| - 1$. If this is the case, a violated odd cycle inequality is found. When all neighbors of v have been considered, we pick a new, colored node, consider its neighbors etc. and proceed in breadth first search manner. In Fig. 1.7 we show an example where the graph G consists of a square. The edge labels correspond to the values y_{uv}. We first color node 1 red and its neigbors 2 and 4 green and red, respectively. We then consider the neighbors of node 2 and color node 3 red. When coloring the neighbors of node 3, we would have to assign the color 'green' to node 4. However, node 4 is already colored red and we end up with a contradiction. The corresponding violated cycle inequality reads $x_{12} + x_{23} - x_{34} + x_{14} \leq 2$.

If y is integral—which we check on the run—and not a cut, this procedure guarantees that a violated odd cycle inequality will be found. So, if for an integral y, the procedure does not produce a violated inequality, y is the incidence vector of a maximum weight cut in G.

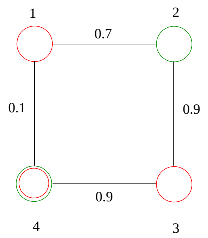


Figure 1.7: Example for the 2-coloring heuristic.

The breadth first search tree built up in this process allows us to generate the violated odd cycle inequalities efficiently. The worst-case running time of our implementation depends on the structure of G_ε and is between $O(|V|)$ and $O(|E| \log |V|)$. Empirically it is $O(|V|)$ and extremely fast.

If we solve spin-glass problems on a grid, then the unchorded 4-cycles of the graph correspond exactly to the grid squares. We scan through all of these and check each of the eight associated cycle inequalities for violation. This can be done in $O(|V|)$ time.

In three dimensions, we also enumerate the four different possibilities for chord-less cycles along the elementary cubes in the graph. This can also be done in $O(|V|)$ time.

If we have an external magnetic field, then all 3-cycles in the graph contain the corresponding field node. By scanning through all grid edges vw we check the four possible 3-cycle inequalities (1.16) that can be derived from the triangle with the external field. This algorithm has $O(|V|)$ running time.

In the last heuristic we calculate a maximum weight spanning tree $T_{\max}$ of G with edge weights $|y_e - \frac{1}{2}|$. For any non-tree edge $e = ij$, we consider its *fundamental cycle* C consisting of the union of e and the unique path from i to j in $T_{\max}$ and set $Q := \{e \in C \mid y_e > \frac{1}{2}\}$. We check whether $|Q|$ is odd and the corresponding odd cycle inequality is violated by y. Using Kruskal's algorithm, this heuristic runs in time $O(|V| \log |V|)$ on the average, and $O(|V|^2)$ in the worst case.

The above described heuristics are called in the same order as we have outlined them above. However, the following heuristic is only called if the previous one did not generate any or too few cuts.

The "exact" separation routine is called if all heuristics together found less than a certain number of cutting-planes. This kind of parameterization keeps the program flexible by allowing us to test various cutting-plane generation strategies. In practical spin-glass computations on a regular grid, our heuristics for generating violated cycle inequalities work very well. They are both fast and effective. When we determine ground-states of spin-glasses in two or three dimensions, there is usually no need to call the exact separation routine.

If we want to determine a maximum cut in an arbitrary graph and all of the above does not produce a cutting-plane, we apply the polynomial separation algorithm for bicycle-wheel separation and simple heuristics for K_p-inequality separation. However, these two separation algorithms are useful only for rather dense graphs. Two or three dimensional grid graphs, even if we have an external magnetic field, do not contain $(2k+1)$-wheels and do not have complete subgraphs larger than a triangle. Nevertheless, in Section 1.8 we will see how these separation algorithms may turn to be useful also for these kinds of graphs.

To keep the number of constraints small we eliminate inequalities in the following way: Whenever the objective function value has decreased by more than some specified amount compared to the previous solution value, all inequalities $a^T x \leq a_0$ with $a^T x^\star < a_0$ for the current optimum solution $x^\star$, are eliminated, otherwise no elimination is performed.

As a by-product of the described separation strategy for cycle inequalities, we obtain cuts in G that arise from (fractional) LP-solutions, e.g., the coloring heuristic and the spanning tree heuristic produce cuts in G, i.e., feasible solutions that can be used as starting solutions for improvement heuristics. Whenever we find a feasible solution with a higher objective function value than the incumbent solution $\bar{x}$ that serves as the best known upper bound for the optimum cut value at any point during the computation, we can update $\bar{x}$. We refer to this technique as "exploiting the LP". Thus our branch-and-cut algorithm produces a sequence of decreasing upper bounds for the optimum cut value, obtained by solving LP-relaxations, and at the same time, a sequence of feasible solutions with increasing objective function value. The branch-and-cut process terminates as soon as these bounds coincide.

As an example, we show in figure 1.8 the evolution of upper and lower bounds computed by our branch-and-cut program for a randomly generated two-dimensional ± 1 spin-glass of size 50×50 with periodic boundaries. The instance took 128 seconds on a 1400 MHz Athlon computer.

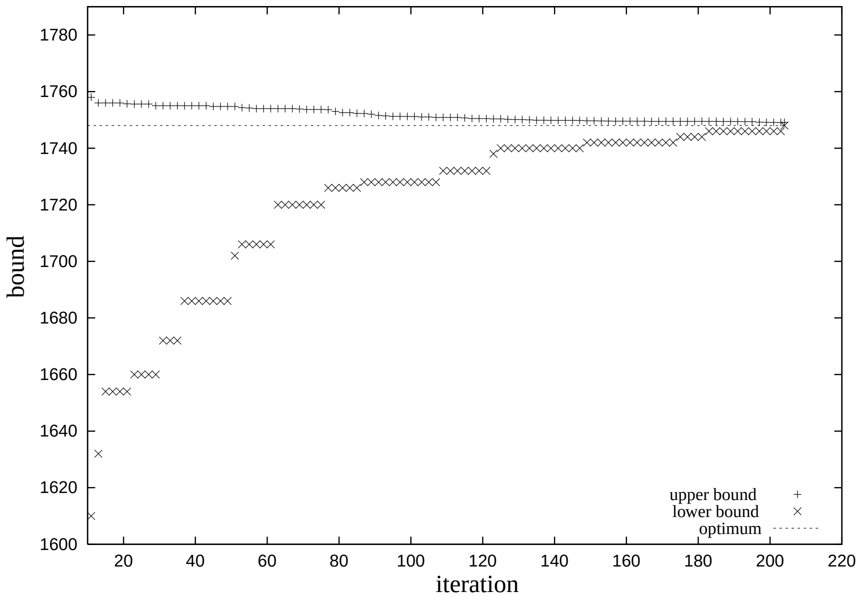


Figure 1.8: Evolution of upper and lower bounds on the max-cut value for a 50×50 grid. For clarity, the data is only shown for every second iteration.

Our branch-and-cut software basically results from integrating the above separation and LP-exploiting procedures into the ABACUS framework [23] for branch-and-cut algorithms that provides the branch-and-bound frame and provisions for dealing with the LP-relaxations.

1.6 Results of Exact Ground-State Computations

For short-range Ising spin-glasses like the 3d grid, the nature of the spin-glass phase is not thoroughly understood, and different models have been described in the literature. For long-range spin-glasses the so-called RSB solution of Parisi [29] is assumed to hold. In the RSB picture of the spin-glass phase there are infinitely many states with an energy difference of order $O(1)$, independent of their size. Therefore, the infinite system has many low lying energy excitations exist that involve an infinite number of spins. The surface of these excitations is space filling, i.e., the fractal dimension d_s of the surface equals the space dimension d. For short-range spin-glasses, the picture is less clear. It is possible that the RSB solution correctly describes the spin-glass phase in finite dimensional spin-glasses. However, Fisher and Huse [15] proposed the "droplet-theory" which implies that the energy cost of flipping a cluster of length scale l is of order $O(l^\theta)$ with θ being a positive constant in three dimensions. Consequently, in this picture there are no excitations of finite energy involving infinitely many spins. The fractal dimension of the surface of these excitations d_s is less than the space dimension d. Also "intermediate" scenarios have been proposed, e.g., [24], [28]. In the so-called TNT scenario we have two exponents θ and θ', which describe the growth of the energy of an excitation of scale L:

(i) θ (> 0) such that L^θ is the typical change in energy when the boundary conditions are changed, for example from periodic to anti-periodic, in a system of size L, and

(ii) $\theta' \sim 0$, which characterizes the energy of clusters excited within the system for a *fixed* set of boundary conditions.

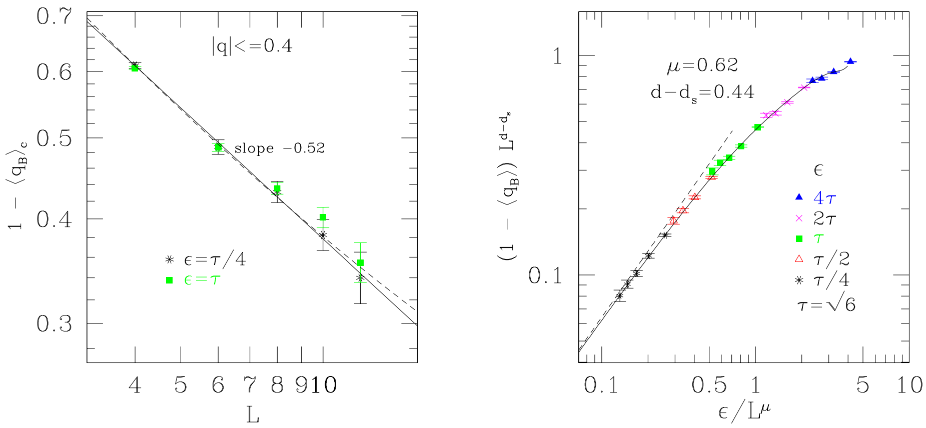
Many numerical studies, mostly done with heuristic algorithms, tried to clarify the nature of the spin-glass phase, but yet none of them could undoubtedly settle this issue.

In the following we describe some results from our work together with M. Palassini and A. P. Young in which we study the nature of the spin glass state. For more details, see [27]. We examine 3d Gaussian distributed spin-glasses with free boundary conditions by following the "bulk perturbation" approach of [28] that works as follows. For a specified instance, we determine the (exact) ground-state with spin configuration $S_i^{(0)}$. We choose a small parameter ε and add the amount $-\frac{\varepsilon}{N_b} S_i^0 S_j^0$ to all couplings J_{ij} (N_b is the number of bonds) and compute the ground-state of the perturbed system with spin configuration $\tilde{S}_i^{(0)}$. This perturbation possibly induces a change in the ground-state, since the energy of the former ground-state increases by ε, whereas the energy of any other state increases by a lesser amount $\frac{\varepsilon}{N_b} \sum_{\langle i,j \rangle} S_i^0 S_j^0 S_i S_j$, which can easily be verified. We study systems of size $L \leq 12$ with different perturbations ε and high statistics.

As we are interested in how the solution changes from the unperturbed to the perturbed system, we analyze different correlation functions and show here the results only for the absolute value of the so-called *box overlap* defined as

$$q_B = \frac{1}{L_B^d} \sum_i S_i^{(0)} \tilde{S}_i^{(0)} \tag{1.28}$$

where the sum runs over the sites contained in a central cubic box of fixed size $L_B = 2$. As the box overlap is measured away from the boundaries, it should have smaller corrections to scaling and should be less sensitive to boundary conditions than, e.g., the usual *spin overlap* q in which the sum in 1.28 extends over the whole lattice. We analyze the average $\langle \cdots \rangle$ of the box overlap as well as restricted averages $\langle \cdots \rangle_c$ where the average is taken only over samples for which the unperturbed and perturbed ground-states are very different, i.e., where the spin overlap satisfies $q \leq q_{\max}$ with $q_{\max}$ chosen appropriately.



(a) Logarithmic plot of the average box-overlap, restricted to samples such that $q \leq 0.4$. The lower continuous line is a power-law fit for $\varepsilon/\tau = 4$. The dashed line is the fit with $\langle 1 - q_B \rangle_c = a + b/L + c/L^2$.

(b) Scaling plot of the box-overlap according to Eq. (1.30). The continuous line is a polynomial fit of order $n = 6$, which gives χ^2/d.o.f = 0.63, and a goodness-of-fit parameter $Q = 0.85$. The dashed line is the linear term of the polynomial fit, corresponding to the asymptotic behavior for $L \to \infty$.

Figure 1.9: Box overlaps.

Fig. 1.9(a) shows the *restricted* average $\langle 1 - q_B \rangle_c$, with $q_{\max} = 0.4$, as a function of L for two values of ε. The data are clearly decreasing with L and close to a straight line on the logarithmic plot, consistent with the droplet or the TNT scenarios. When a large-scale cluster of spins is flipped, for large L the probability that its surface goes across the central box is proportional to the ratio of its surface area, $\sim L^{d_s}$, to the volume, L^d. Therefore $1 - q_B \sim L^{-(d-d_s)}$. The exponent $d - d_s$ can be read off from the log-log plot Fig. 1.9(a) as the slope of the straight line. We obtain the estimate

$$d - d_s = 0.48 \pm 0.03. \tag{1.29}$$

The box overlap is expected to scale as [27]

$$\langle 1 - q_B \rangle = L^{-(d-d_s)} F_{q_B}(\varepsilon/L^{\mu}), \tag{1.30}$$

where $F_{q_B}(x) \sim x$ for small x and $\mu \equiv \theta' + d - d_s$. With θ' we denote the energy exponent that might be different from the exponent θ introduced in the droplet theory. We observe that our data scale well according to (1.30), see Fig. 1.9(b), and we obtain the best data collapse for $\mu = 0.62 \pm 0.04$. We find $\theta' = 0.15 \pm 0.7$ which is consistent with the droplet scaling picture.

However, the data are also fitted well by the RSB picture, if we allow large corrections to scaling. Under the RSB assumption we estimate $\lim_{L\to\infty}\langle 1 - q_B\rangle_c = 0.25 \pm 0.10$ (not derived here). In this case, the good scaling behavior we observed would only be a finite-size artifact, and would disappear at larger sizes. We get comparable results when analyzing other correlation functions, e.g., the link overlap. Therefore, by current standards, it is not possible to undoubtedly clarify the nature of the spin-glass state by our work [27]. For being able to do this, larger system sizes will be needed.

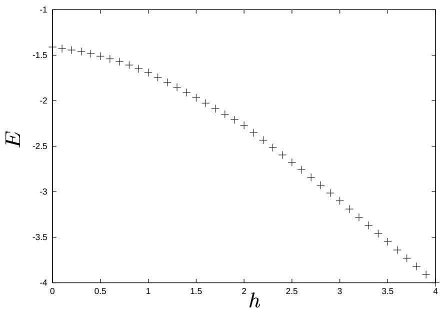
1.7 Advantages of Branch-and-Cut

Usually, for being able to draw physical conclusions, we have to average over the disorder which means here that we calculate the ground-state of many (hundreds, thousands or more) realizations of a spin-glass with the same characteristics (e.g., of the same size) and study the average, higher moments or even the whole probability distribution of the physical quantities we are interested in. Obviously, for a good statistics one needs sufficient computer resources.

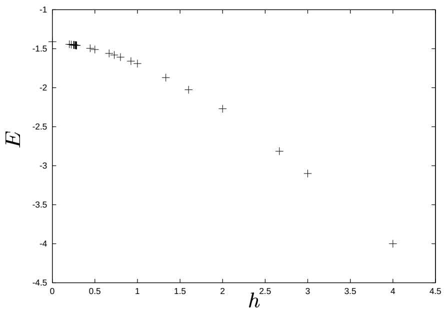
Physical projects like the one described above, often are of this flavor: First we take an instance and compute the ground-state. Then we slightly change the problem (e.g., perturb some couplings), compute the new ground-state and study the relations between the solutions. With methods like Monte Carlo simulations, we would have to compute both solutions from scratch. However, when doing branch-and-cut we know that the formerly optimal solution is still feasible for the new problem and can be used as a starting solution. The linear programs generated during the run of the algorithm consist only of valid inequalities for the cut polytope. The last solved linear programs might still be a "tight" relaxation for the new problem and can serve as starting relaxations. By using this information from the previous computation, we can save a considerable amount of running time.

The spin-glass susceptibility is another quantity for which our branch-and-cut approach is especially suited. Suppose we have found a ground-state and we wish to see how the spin configuration changes when an external magnetic field of continuously increasing strength is applied. The obvious way is to increase the field strength in fixed increments and compute a new ground-state starting with the optimum solution of the previous field strength. If this previous optimum solution was found without branching, we can do even better: Linear programming techniques that are beyond the scope of this introduction can be used to determine the maximum field for which the current spin configuration, which is the optimum integral solution of some linear programming relaxation, remains optimum. Up to this field strength, the energy drops linearly and the magnetization stays constant. Beyond this point, a new optimization "warm starts" on the previous optimum solution. This technique is currently under investigation [21]. It has the clear advantage over the obvious fixed increment approach that the exact break points in the piecewise linear energy curve are determined and much fewer reoptimizations are necessary.

In Fig. 1.10 we show the energy per spin as a function of the strength h of the external field for a randomly chosen two-dimensional ± 1 instance of size 20×20. Fig. 1.10(a) shows the data generated by increasing the field in fixed increments Δh = 0.1. Fig. 1.10(b) shows the corresponding plot for the same instance when the data is generated by the sensitivity method described above. Here the field strengths at which the solutions change are determined exactly.



(a) Energy per spin as a function of the field strength h for a 20×20 ± 1 spin-glass. The field is increased in fixed increments $\Delta h = 0.1$.



(b) Energy per spin as a function of the field strength h for a 20×20 ± 1 spin-glass. The field strengths at which the solutions change are determined exactly.

Figure 1.10: Energy E per spin as a function of the external field strength h for a randomly chosen $2d$ instance of size 20×20.

1.8 Challenges for the Years to Come

One approach to reduce the running time of the algorithm is to reduce the number of subproblems to be solved. Some work in progress [21] tries to exploit the many results on the facial structure of the cut polytope $P_{\mathrm{CUT}}^{K_p}$ for the complete graph with p nodes compiled by Deza and Laurent in [13] without completing the graph with 0-weight edges, because this would render the solution practically impossible. The idea is as follows:

We are given a point $\bar{x}_n \in P_{\mathrm{CYCLE}}^G$ that does not belong to P_{CUT}^G, G being an arbitrary graph with n nodes. We want to find an inequality valid for P_{CUT}^G (possibly facet defining) that is not satisfied by $\bar{x}_n$. To do so, we want to use the algorithmic and the structural results that are available for $P_{\mathrm{CUT}}^{K_p}$, the cut polytope for a complete graph.

First, by a sequence of operations on $\bar{x}_n$ that amount to switching along certain cuts and contracting node pairs corresponding to the end-nodes of integral edges, such a point is transformed to $\bar{x}_{n'} \in \mathbb{R}^{E'}$, where n' is usually much smaller than n. The point $\bar{x}_{n'}$ is always guaranteed to be outside $P_{\mathrm{CUT}}^{G'}$ but inside $P_{\mathrm{CYCLE}}^{G'}$. It can be seen as a fractional solution of a cutting-plane algorithm applied to a max-cut instance on a smaller and denser graph $G' = (V', E')$ where $|V'| = n'$.

At this point all the machinery available for the max-cut problem on complete graphs can be used for each complete subgraph K_p of G'. Therefore, some separation procedures for the cut polytope on complete graphs are applied to the restriction of $\bar{x}_{n'}$ to the edges of these components that (hopefully) generate an inequality $a_{n'}x_{n'} \leq \alpha$, valid for $P_{\mathrm{CUT}}^{G'}$ and violated by $\bar{x}_{n'}$ by an amount β.

Finally, a sequence of "lifting" procedures is applied to $a_{n'}x_{n'} \leq \alpha$ that transforms it to an inequality $a_n x_n \leq \beta$ valid for P_{CUT}^{G} and violated by $\bar{x}_n$ by the same amount β. As a by-product, one of these lifting procedures provides a way to generate facet defining inequalities for P_{CUT}^{G}. Namely, under certain conditions, this procedure, applied to a facet defining inequality for $P_{\mathrm{CUT}}^{G'}$, produces not only a valid, but also a facet defining inequality for P_{CUT}^{G}.

These separation and lifting procedures enrich the description by linear inequalities of the cut polytope on arbitrary graphs and, at the same time, constitute an algorithmic tool for exactly solving the max-cut problem on these graphs.

Some preliminary experiments indicate that the above sketched procedure can indeed reduce the number of subproblems considerably. However, currently it is still too slow to give a significant overall performance gain. However, we are confident that optimizing our procedure will result in an improvement of the running time in the future. Another clear advantage of such a procedure is that it helps in solving problems without enumeration, which is a prerequisite for applying the sensitivity method discussed in the previous section.

In practical computations, around 90% of the total running time is spent in solving the linear programs by the simplex algorithm. Therefore, a topic of current research is to study the performance of branch-and-cut by replacing the simplex algorithm with fast approximate linear program solvers. The rationale for using an approximate solver is that especially in the beginning of the optimization process the current relaxation is not a "tight" relaxation of the cut polytope anyway. Additionally, methods like interior point solvers have a nice feature: They tend to return a solution that it "near" an optimal face, and not an optimal vertex. Intuitively this means that in the next round of separation the generated cutting-planes will cut "deeper" than if only a vertex is cut off. Therefore we expect less rounds of separation.

Recently, Barahona et al. suggested to use the so-called volume algorithm [5], [8] inside a branch-and-cut framework. The volume algorithm is a subgradient method for generating an approximate solution of a linear program that can be modified to converge to its optimal solution, see [4]. During the run of the algorithm, the current relaxation is solved exactly always before a possible branching step takes place in order to correctly check whether a node can be fathomed. It turns out in [8] that the running times of their volume based code are much better for two dimensional ± 1 spin-glass problems than their simplex based code. However, the performance for the other problem classes (e.g., three dimensional instances) is much worse with the volume algorithm. Thus the volume algorithm seems to improve the performance only in limited cases.

In ongoing studies we replace the simplex method with an interior point solver resulting in a considerably smaller number of linear programs to be solved. Despite the smaller numbers of linear programs, the overall running time is worse than with the traditional simplex as the time for solving each LP increases. Up to now, we also find worse performance when using so-called bundle-methods [31] that are also approximate solvers for which there exists a proof of convergence. Thus the question is still open whether we can gain performance by replacing the simplex algorithm by other methods.

Acknowledgement. This work is partially supported by the Future and Emerging Technologies Programme of the European Union under contract number IST-1999-14186 (ALCOM-FT).

Bibliography

[1] K. Aardal, S. P. M. van Hoesel, A. Koster, C. Mannino, and A. Sassano, *Models and Solution Techniques for Frequency Assignment Problems*, *ZIB-Report 01-40*, December 2001.

[2] F. Barahona *On the computational complexity of Ising spin glass models* J. Phys. A: Math. Gen. **15** 3241–3253 (1982).

[3] F. Barahona *The Maxcut Problem on Graphs not Contractable to K_5* Operations Research Letters 2, 107–111 (1983).

[4] L. Bahiense, N. Maculan, and C. Sagastizábal, *The Volume Algorithm Revisited: Relation with Bundle Methods*
www.optimization-online.org/DB_HTML/2001/11/400.html

[5] F. Barahona and R. Anbil, *The Volume Algorithm: Producing Primal Solutions with a Subgradient Algorithm*, *Mathematical Programming* **87**, 385–399 (2000).

[6] F. Barahona, M. Grötschel, M. Jünger, and G. Reinelt, *An Application of Combinatorial Optimization to Statistical Physics and Circuit Layout Design*, *Operations Research* **36**, 493–513 (1988).

[7] F. Barahona, M. Jünger, and G. Reinelt, *Experiments in Quadratic 0-1 Programming*, *Mathematical Programming* **44**, 127–137 (1989).

[8] F. Barahona and L. Ladányi,
www.optimization-online.org/DB_HTML/2001/12/420.html

[9] F. Barahona and A. R. Mahjoub, *On the Cut Polytope*, *Mathematical Programming* **36**, 157–173 (1986).

[10] M. Bellare, O. Goldreich, M. Sudan *Free bits, PCPs and non-approximability - towards tight results* Proc. of 36th Ann. IEEE Symp. on Fundations of Computer Science IEEE Computer Society 422-431 (1995).

[11] E. Boros and P. L. Hammer, *The Max-Cut problem and Quadratic 0/1 Optimization*, Polyhedral Aspects, Relaxations and Bounds, *Annals of Operations Research* **33**, 151–180 (1991).

[12] V. Chvátal, *Linear Programming*, (W. H. Freeman and Company, New York, 1983).

[13] M. Deza and M. Laurent, *Geometry of Cuts and Metrics*. Algorithms and Combinatorics Vol. 15 (Springer-Verlag, Berlin, 1997).

[14] M. Elf, M. Jünger, and G. Rinaldi, *Minimizing Breaks by Maximizing Cuts*, Report No. 2001.409, Angewandte Mathematik und Informatik, Universität zu Köln, to appear in: *Operations Research Letters*, 2003.

[15] D. S. Fisher and D. A. Huse, J. Phys. A. **20** , L997 (1987); D. A. Huse and D. S. Fisher, J. Phys. A. **20**, L1005 (1987); D. S. Fisher and D. A. Huse, Phys. Rev. B **38**, 386 (1988).

[16] A. Galluccio, M. Loebl, and J. Vondrak, *Optimization via Enumeration: A New Algorithm for the Max Cut Problem*, *Mathematical Programming* **90**, 273–290 (2001).

[17] B. Gerards, *Testing the Odd Bicycle Wheel Inequalities for the Bipartite Subgraph Polytope*. *Mathematics of Operations Research* **10**, 359–360 (1985).

[18] M. Grötschel, L. Lovász, and A. Schrijver, *The Ellipsoid Method and its Consequences in Combinatorial Optimization*, *Combinatorica* **4**, 169–197 (1981).

[19] A. K. Hartmann, H. Rieger, *Optimization Algorithms in Physics* (Wiley-VCH 002)

[20] M. X. Goemans and D. P. Williamson *.878-approximation algorithms for MAX CUT and MAX 2SAT.* ACM Symposium on Theory of Computing (STOC) (1994).

[21] M. Jünger, G. Reinelt, and G. Rinaldi, *Lifting and Separation Procedures for the Cut Polytope*, Technical Report, Universität zu Köln, in preparation.

[22] M. Jünger and G. Rinaldi, *Relaxations of the Max Cut Problem and Computation of Spin-Glass Ground-States*, in: P. Kischka et al. (eds.), *Operations Research Proceedings 1997*, Springer, 74–83 (1998).

[23] M. Jünger and S. Thienel, *The ABACUS System for Branch-and-Cut-and-Price Algorithms* in Integer Programming and Combinatorial Optimization, *Software Practice and Experience* **30**, 1325–1352 (2000).

[24] F. Krzakala and O. C. Martin, Phys. Rev. Lett. **85**, 3013 (2000).

[25] A. H. Land and A. G. Doig, *An Automatic Method for Solving Discrete Programming Problems*, *Econometrica* **28**, 497–520 (1960).

[26] M. Padberg, *Linear Optimization and Extensions*, (Springer-Verlag, Berlin, 1995).

[27] M. Palassini, F. Liers, M. Jünger, and A. P. Young, *Low Energy Excitations in Spin-Glasses from Exact Ground-States*, Physical Review B 68, 064413 (2003).

[28] M. Palassini and A. P. Young, Phys. Rev. Lett. **85**, 3017 (2000).

[29] G. Parisi, Phys. Rev. Lett. **43**, 1754 (1979); J. Phys. A **13**, 1101, 1887, L115 (1980); Phys. Rev. Lett. **50**, 1946 (1983).

[30] L. Saul and M. Kardar, Phys. Rev. **E 48**, R3221 (1993); Nucl. Phys. B **432**, 641 (1994).

[31] H. Schramm and J. Zowe A Version of the Bundle Idea for Minimizing a Nonsmooth Function: Conceptual Idea, Convergence Analysis, Numerical Results SIAM J. Opt. 2 121–152, (1992).