Contributions to Determining Exact Ground-States of Ising Spin-Glasses and to their Physics

Inaugural-Dissertation
zur
Erlangung des Doktorgrades
der Mathematisch-Naturwissenschaftlichen Fakultät
der Universität zu Köln

vorgelegt von
Frauke Liers
aus Ingelheim am Rhein

Köln 2004

Berichterstatter: **Prof. Dr. Michael Jünger**
Prof. Dr. Dietrich Stauffer

Tag der mündlichen Prüfung: **07. Juli 2004**

Zusammenfassung

Seit Jahrzehnten wird intensiv daran gearbeitet, sogenannte Spingläser (z.B. die Legierungen CuMn oder AuFe) besser zu verstehen. Sie lassen sich bisher theoretisch nur unzureichend beschreiben. Man ist auf die Interpretation numerischer Ergebnisse angewiesen, um in der Literatur vorgeschlagene Theorien testen zu können. Wir betrachten Spingläser im Isingmodell, in dem die Spins (magnetische Dipole) genau zwei Einstellungsmöglichkeiten besitzen.

Uns interessieren die Zustände tiefster Temperaturen, bei denen die Spins ungeordnet 'einfrieren'. Das Bestimmen eines Zustandes minimaler Energie, eines Grundzustandes, läßt sich auf die Berechnung eines maximalen Schnittes in einem Graphen überführen. Das Maximum Schnitt Problem ist ein prominentes $\mathcal{NP}$-schweres Problem aus der kombinatorischen Optimierung. Maximale Schnitte können mit einem Branch and Cut Algorithmus exakt bestimmt werden.

Die vorliegende Arbeit ist interdisziplinär angelegt zwischen kombinatorischer Optimierung und theoretischer Physik. Im ersten Teil wird das Maximum Schnitt Problem und ein Branch and Cut Algorithmus zur Lösung von Instanzen vernünftiger Grösse vorgestellt. Es werden verschiedene Ansätze vorgestellt, wie dieser Algorithmus für Instanzen, die von Ising Spingläsern stammen, verbessert werden kann.

Im zweiten Teil der Arbeit studieren wir die Physik von Spingläsern. Wir stellen zuerst den Stand der Forschung dar. Danach geben wir Ergebnisse für sogenannte Bethe Spingläser an. Zuletzt studieren wir die Natur von Spingläsern im kurzreichweitigen dreidimensionalen Gitter. Erstere Resultate sind in einer Kooperation mit Dr. M. Palassini und PD Dr. A.K. Hartmann entstanden, letztere in Zusammenarbeit mit Dr. M. Palassini und Prof. A. Peter Young.

Danksagung

An dieser Stelle möchte ich meinen herzlichsten Dank an all diejenigen aussprechen, die mich bei dieser Arbeit unterstützt haben.

Zuallererst gilt mein Dank Herrn Prof. Michael Jünger, der mir sowohl wissenschaftlich als auch persönlich ein Vorbild war. Er hat mich an das spannende interdisziplinäre Thema "Spingläser" herangeführt und hat mir die Möglichkeit gegeben, sowohl zu Optimierern als auch zu Physikern Kontakte herzustellen. Seiner Unterstützung konnte ich mir stets sicher sein. Bei ihm und meinen Arbeitskolleginnen und -kollegen Michael Belling, Dr. Christoph Buchheim, Matthias Elf, Joachim Kupke, Thomas Lange, Ursula Neugebauer, Merijam Percan und Martin Wolff bedanke ich mich für die Schaffung einer kreativen Arbeitsatmosphäre. Dank gilt Prof. Dietrich Stauffer für sein Interesse an meiner Arbeit.

Mein herzlichster Dank gilt Dr. Matteo Palassini, der mir ein hervorragender Lehrer für die Physik von Spingläsern war. Sehr herzlich bedanken möchte ich mich bei ihm, PD Dr. Alexander K. Hartmann und bei Prof. A. Peter Young für die fruchtbare Zusammenarbeit, durch die ich viel gelernt habe und die mir viel Freude bereitet hat. Ergebnisse dieser Kooperationen sind im zweiten Teil dieser Arbeit zusammengefasst.

Bei Prof. Giovanni Rinaldi bedanke ich mich dafür, dass er mir die Möglichkeit gegeben hat, fünf Monate in Rom arbeiten zu können. Dank gilt auch seinen Mitarbeiterinnen und Mitarbeitern, die mitgeholfen haben, diese Zeit zu einer sehr interessanten und lehrreichen zu machen. Bedanken möchte ich mich bei ihm, Prof. Petra Mutzel und Prof. Gerhard Reinelt für all die interessanten Diskussionen über das Maximum Schnitt Problem. Mein herzlicher Dank gilt Prof. Miguel F. Anjos für die gute Zusammenarbeit während seiner Zeit als Post-Doc in Köln.

Prof. Olivier C. Martin verdanke ich einen zweimonatigen Forschungsaufenthalt in Orsay, in dem ich mein Wissen in der Physik von Spingläsern vertiefen konnte. Bei ihm und Dr. Helmut Katzgraber bedanke ich mich für die vielen interessanten Diskussionen. Dank gilt Dr. Katzgraber für das kritische Durchlesen von Kapitel 5.

Bei Thomas Lange, den Mitarbeiterinnen und Mitarbeitern des Kölner Rechenzentrums RRZK und den Administratoren des SCALE Clusters des Lehrstuhls Prof. Ewald Speckenmeyer möchte ich mich für das Bereitstellen von Rechenleistung und vielfältigen Support bedanken.

Diese Arbeit wäre natürlich nicht möglich gewesen ohne die Unterstützung meiner Eltern, meiner Schwester und meiner Freunde. Bei ihnen bedanke ich mich für den angenehmen Ausgleich, den sie zu meiner Arbeit bildeten. Mein Dank gilt vor allem Klaus Bergmann.

Abstract

In the last decades, much research has focused on a better understanding of so-called spin glasses (e.g., the alloys CuMn and AuFe.) Spin glasses are not yet fully understood. In order to be able to test the different theories that have been proposed for the nature of spin glasses we have to analyze numerically generated data. We consider spin glasses in the Ising model, where the spins (magnetic dipoles) have exactly two possibilities for aligning themselves.

We are interested in the low-temperature states of the system, in which the spins are 'frozen' and disordered. Determining a state of minimum energy, a ground state, amounts to calculating a maximum cut in a graph. The max-cut problem is a prominent $\mathcal{NP}$-hard problem from combinatorial optimization. Maximum cuts can be determined exactly with a branch–and–cut algorithm.

This thesis consists of two parts. In the first part we introduce the max-cut problem and a branch–and–cut algorithm for solving reasonably sized problems. We present several approaches for improving its performance for Ising spin-glass instances.

In the second part of this work, we study the physics of spin glasses. We first discuss what is known in the literature. Then we present results for Bethe spin glasses. Finally we study the nature of short-range three-dimensional spin glasses. Results of the former were obtained in collaboration with M. Palassini and A.K. Hartmann, results of the latter in cooperation with M. Palassini and A.Peter Young.

Acknowledgments

I would like to express my deepest thanks to all the people who supported me with this work.

First of all, I wish to thank Michael Jünger from whom I learned very much both scientifically and personally. He drew my attention to the exciting interdisciplinary field of spin glasses and gave me the possibility to get in contact not only with mathematicians but also with physicists. I could always be sure of his support. I am grateful to him and to my colleagues Michael Belling, Christoph Buchheim, Matthias Elf, Joachim Kupke, Thomas Lange, Ursula Neugebauer, Merijam Percan and Martin Wolff for establishing a highly creative working atmosphere. I also thank Dietrich Stauffer for his interest in my work.

My deepest thanks go to Matteo Palassini who was an excellent teacher in the physics of spin glasses. I am indebted to him, Alexander K. Hartmann and A. Peter Young for the fruitful collaboration in which I learned very much. It was a great pleasure for me. A summary of the results obtained in these collaborations can be found in the second part of this thesis.

I am grateful to Giovanni Rinaldi for having given me the possibility to continue my research in Rome. I wish to thank him and his staff for making these five months very instructive and pleasant. I am grateful to him, Petra Mutzel and Gerhard Reinelt for all the stimulating discussions on the maximum cut problem. Many thanks also to Miguel F. Anjos for the splendid collaboration while he was a postdoc in Cologne.

I wish to thank Olivier C. Martin for making a two-months research stay in Orsay possible, in which I could improve my knowledge in spin-glass physics. I am grateful to him and Helmut Katzgraber for many fruitful discussions. I wish to thank Helmut for critical reading of chapter 5.

I am indebted to Thomas Lange, the staff of the Regional Centre of Computing (RRZK) in Cologne and the administrators of the SCALE linux cluster from Ewald Speckenmeyer's group for the allocation of computer time and various support.

Of course, this work would not have been possible without the support of my parents, my sister and my friends. They formed a most enjoyable balance to my work. In the first place, I wish to thank Klaus Bergmann.

Contents

Introduction

Interactions between physicists and computer scientists are steadily growing, and improvements in one field stimulate new developments in the other. With methods coming from physics, e.g., the simulated annealing or the cavity method, new algorithms can be designed for solving several problems occurring in computer science or in applied mathematics. On the other hand, several problems arising in statistical physics can be mapped on combinatorial optimization problems. With the help of solution algorithms we can gain a deeper insight into the characteristics of the problems.

Traditional representatives for problems of this kind are *spin glasses*. 'Real' spin glasses are, for example, the alloys CuMn or AuFe, where manganese (Mn) or iron (Fe) is brought as impurities into copper (Cu) or gold (Au), respectively. In the 1970s it became obvious that these materials show surprising behaviour at low temperatures in laboratory experiments, e.g., when brought into an oscillating magnetic field. It was soon clear that the type of magnetism present in spin glasses was different to what was known before. Since then, many new theoretical approaches have been designed to understand their physics. Still there are some challenging open problems.

In the 'classical' spin glasses AuFe or CuMn, the spins (magnetic dipoles) of the impurities produce a magnetic polarization of the host metal conduction electrons which is ferromagnetic at some distances and antiferromagnetic at others. This magnetic polarization produces local magnetic fields. A different impurity spin tries to align itself according to the local fields. As the impurities are randomly scattered in the host, some interactions are ferromagnetic and some are antiferromagnetic.

What makes a spin glass different to paramagnets or ferromagnets? For constituting a spin glass, two basic ingredients are necessary: *randomness* (e.g., in the position of the impurities) and *competing interactions*. Competing interactions means that no spin configuration is uniquely favoured by all interactions, which is called *frustration*. At low temperatures the spins freeze. Spin glasses inherit a certain *ordering* that is however different from the spatial ordering present in ferro- or antiferromagnets.

Randomness has become an important ingredient in modern physics. Furthermore, spin glass has become a fundamental form of magnetism. Some of the new concepts and ideas, once introduced for spin glasses, are now successfully applied to other

problems as well. At present, there are still some unsolved problems making spin glasses an active and important field of research.

It turned out that some real spin glasses can be treated in the so-called Ising model, in which the spins can only point in two directions, 'up' or 'down'. The high temperature of a spin glass is paramagnetic. However, at low temperatures the spins are frozen and disordered. We are mainly concerned with *ground states* of spin glasses, i.e., states that attain the global minimum of the energy associated with the system. As the ground states also influence the low-lying excited states, they are important. Additionally, knowledge about ground states and the application of a 'stimulus' (i.e., a slight increase of the ground-state energy) can give new insights into spin-glass physics.

As we do not have a closed-form function whose evaluation yields a ground-state spin configuration, we have to use a numerical algorithm. A naive approach would be to enumerate all configurations and to take one with minimal energy as a ground state. However, the number of possible configurations grows exponentially fast and already becomes computationally too demanding for small systems. Therefore this approach is not feasible in practical computations.

However, a close connection between Ising spin glasses and combinatorial optimization exists. The problem of determining a ground state of an Ising spin glass is equivalent to the prominent *max-cut problem* from combinatorial optimization.

In combinatorial optimization we are concerned with problems of the following form. Let $N = \{1, \ldots, n\}$ and consider a finite collection of subsets, say $\{S_1, S_2, \ldots, S_m\}$. For each subset we are given an objective function value, $f(S_k)$, and the problem is either to maximize or minimize $f(S_k)$. Combinatorial optimization problems are, for example, the shortest path problem, the maximum flow problem, the well-known travelling salesman problem and many others.

In the first part of this thesis we will be concerned with the max-cut problem from combinatorial optimization. We are given a weighted graph $G = (V, E)$. The cut $\delta(W)$ associated with a node set $W \subseteq V$ is defined as the set of edges having exactly one endpoint in W. The weight of a cut is the sum of the weights of the cut edges. The max-cut problem is to find a cut of G with maximum weight. On general graphs this is a 'hard' both in theory and in practice. In fact, the max-cut problem was one of the first seven problems for which a proof of their $\mathcal{NP}$-hardness could be given. It is widely believed that this means that we will not be able to design a solution algorithm whose running time is bounded by a polynomial in the size of the input. Therefore, in practice we have to study algorithms with exponential running time. Those negative results of course raise interest in designing and implementing algorithms that can solve in practice 'medium sized' problems, however if not in *polynomial* time, then at least within *reasonable computation time*. In the eighties so-called branch–and–cut techniques were introduced for the solution

of hard combinatorial optimization problems. This technique is successfully applied to many hard problems.

This thesis consists of two parts. The first part is devoted to the study of the max-cut problem. Here we pay special attention to max-cut problems that are defined on regular grid graphs in two and three dimensions. We first summarize the state of the art in the research on the max-cut problem in Chapter 1. We establish the connection between max-cut and the problem of determining ground states of Ising spin glasses. Then we introduce the branch–and–cut approach for solving hard max-cut instances. Subsequently, we summarize what is known about the structure of the cut polytope, which is the convex hull of the cut vectors. During the run of the branch–and–cut algorithm, we have to generate and optimize over progressively better approximations (*relaxations*) of the cut polytope. In Chapter 2 we first experimentally show that the cycle relaxation gives a good approximation of the cut polytope in practice. Then we show how we can generate and solve the cycle relaxation for sparse Ising spin glass instances. We introduce a new facet, the *4-neighbour* facet for the cut polytope in Section 2.4. In the subsequent section we generate tighter approximations of the cut polytope by using a lift-and-project approach proposed by Jünger, Reinelt and Rinaldi. Finally, we show how good cuts can be generated within the branch–and–cut framework.

In Chapter 3 we devise a branch-cut&price algorithm for solving spin glass instances defined on the one-dimensional Ising chain. The model consists of a fully connected graph where all spins are connected with each other. However, the coupling strengths fall off with a positive power of the spin distance. We can additionally improve the quality of the relaxations of the cut polytope. Both the pricing and the better relaxations yield a considerable speedup.

In the second part of the thesis we study the physics of spin glasess. First we start with the historic developments and give an introduction into spin-glass physics in Chapter 5. Then we argue that a heuristic algorithm should fail with higher probability in case the energy landscape of an instance is 'complicated'. There are obvious advantages of an exact method for determining ground states over heuristic methods that are not able to guarantee that true ground states are determined. With heuristic methods, we always introduce a certain bias in the measured data, which is positive, e.g., when determining the ground-state energy. Two spin configurations with almost the same energy might be very different in configuration. Therefore it is not clear how accurate the heuristic data is and whether useful insight into the structure of the ground-state spin configurations can be gained.

In Chapter 6 we study the ground-state properties of so-called Bethe lattices by analyzing the exact data generated by branch–and–cut. For the spin-glass/ferromagnet phase transition we determine the transition point and study the behavior of the branch–and–cut algorithm in the vicinity of the phase transition. Results are obtained in collaboration with Matteo Palassini and Alexander K. Hartmann. In Chap-

ter 7 we study the nature of the low-lying excitations in the three-dimensional spin glass which summarizes the results obtained in cooperation with Matteo Palassini and A. Peter Young. Finally, we give a summary of this thesis and mention prospectives for further research in the Conclusions.

Part I

Branch–And–Cut For Max-Cut

Chapter 1

The Max-Cut Problem

The max-cut problem has been a topic of intensive research. It is a 'classical' combinatorial optimization problem and one of the first that could be proven to be $\mathcal{NP}$*-hard* on arbitrary graphs. Most people believe this means that the running time of any of its solution algorithms depends exponentially on the size of the input, in the worst-case. Another reason for the interest in the max-cut problem comes from the fact that it is equivalent to quadratic 0/1 programming. Furthermore, it was the first problem for which approximation guarantees could be given with methods from positive semidefinite optimization. Finally, exact ground states of Ising spin glasses can be computed by calculating maximum cuts. In this introductory chapter we summarize the most important facts known about the max-cut problem. After defining the problem we establish the connection between the determination of maximum cuts and the calculation of ground states of Ising spin glasses. Then we explain the basic concepts of a branch–and–cut algorithm that is used to calculate maximum cuts. Finally, we summarize in Section 1.3 what is known about the polyhedral description of the cut polytope.

1.1 The Max-Cut Problem and Ground States of Spin Glasses

In the max-cut problem we are given a graph $G = (V, E)$ with edge weights $c_{ij} \in \mathbb{R}$ for all edges $e \in E$. Let $W \subset V$ be a (possibly empty) subset of nodes. The *cut* $\delta(W)$ is defined as the set of edges having exactly one endpoint in W. In formulas, for $W \subset V$ the cut is defined as

$$\delta(W) = \{(i, j) \in E \mid i \in W, j \in V \setminus W\}. \tag{1.1}$$

The weight of $\delta(W)$ is sum of the weights of the edges in the cut, $\sum_{e\in\delta(W)} c_e$. The *max-cut problem* is to find a cut of G with maximum weight. For a detailed book on the max-cut problem that covers many theoretical aspects, see [30].

Despite the $\mathcal{NP}$-hardness of the general max-cut problem, there are some classes of graphs for which it is *polynomially solvable*, i.e., for which it can be solved in a number of elementary steps that is polynomially bounded by the number of bits needed to store the input data. The max-cut problem is polynomially solvable for graphs that are not contractable to K_5 (the complete graph on five nodes) [8]. This class of graphs include planar graphs. It is polynomially solvable for weakly bipartite graphs and graphs with non-positive edge weights. It is interesting to notice that max-cut is already $\mathcal{NP}$-hard for almost planar graphs [8], i.e., graphs where only one node has to be removed to obtain a planar graph.

Goemans and Williamson [41] 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 $\mathcal{P} \neq \mathcal{NP}$ there is no polynomial algorithm that provably delivers a solution of at least 98% of the optimum value of a maximum cut [14].

Next we describe the connection between the determination of ground states of Ising spin glasses and the max-cut problem. For an introduction into the field of spin glasses and the Ising model used here, we refer to Chapter 5 and the references therein. Let a spin glass consist of n spins. Spins i and j might be coupled with coupling strength J_{ij}. Usuall, the couplings are either Gaussian distributed following the probability distribution $P(J)$ with

$$P(J) = \frac{1}{\sqrt{2\pi}} \exp(-J^2/2). \tag{1.2}$$

or have value $\{\pm J\}$, with 50% negative values. We study an Ising model in which the spin variable S_i for spin i is one dimensional and can take only the two values $+1$ or -1. An external magnetic field of strength h might be present. The Hamiltonian of a system with spin configuration $\omega = (S_1, \ldots, S_n)$ is

$$H(\omega) = -\sum_{(ij)} J_{ij} S_i S_j - h \sum_{i=1}^{n} S_i, \tag{1.3}$$

where the sum $\sum_{(ij)}$ runs over the coupled spins. We identify the spins with the node set $V = \{1, \ldots, n\}$ of a graph $G = (V, E)$. Two nodes i and j are connected by an edge $e \in E$ if and only if spin i and j are coupled by a nonzero coupling strength J_{ij}. For modelling the external field, we introduce a new node "0" for the field having spin S_0. Node 0 is connected via an edge $(0, i)$ to all other spins $i \in V$.

We let the graph $G_0 = (V_0, E_0)$ consist of the nodes and edges of G together with the field node 0 and the field edges $(0, i)$. By setting the field couplings J_{0i} as $J_{0i} = h$, we can write (1.3) as

$$H(\omega) = -\sum_{(ij)\in E_0} J_{ij} S_i S_j. \tag{1.4}$$

A spin configuration ω corresponds to a partition of the nodes $V = V_0^+ \cup V_0^-$, where $V_0^+ := \{i \in V_0 \mid S_i = +1\}$ and $V_0^- = \{i \in V_0 \mid S_i = -1\}$. We split the sum in the right hand side of (1.4) as

$$H(\omega) = -\sum_{(i,j)\in E_0,\ i,\ j \text{ both in } V_0^+ \text{ or in } V_0^-} J_{ij} \ + \sum_{(i,j)\in E_0,\ i \in V_0^+, j \in V_0^-} J_{ij}$$

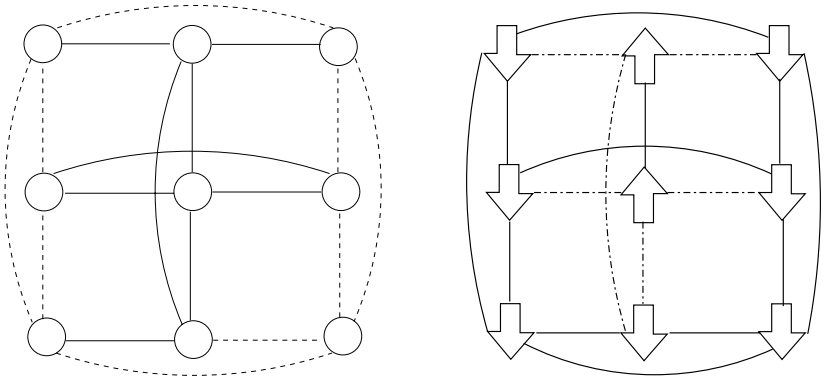
We add to both sides of the equation the sum of all couplings in the graph which is a constant and end up with

$$H(\omega) + \sum_{(ij)\in E_0} J_{ij} = 2 \sum_{(ij)\in \delta(V^+)} J_{ij}.$$

Therefore, we have expressed the energy function in terms of cuts in G. We set the weight of edge $(i, j) \in E$ as $c_{ij} := -J_{ij}$. Hence, minimizing the Hamiltonian is equivalent to maximizing the weight of the cut in the graph G_0 over all possible sets $V^+ \subseteq V$. We conclude that determining ground states of Ising spin glasses can be determined by calculating maximum cuts in the graph associated with the spin glass system.

As an example, we show in Figure 1.1 an instance on a 3×3 grid with periodic boundary conditions, $\pm J$ interactions and no external field. Figure 1.1(a) shows the instance of 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. Figure 1.1(b) shows an optimum solution. The dash-dotted lines correspond to the cut edges.

Therefore, determining ground states of Ising spin glasses is $\mathcal{NP}$-hard. However, polynomially solvable cases exist. For example, the two-dimensional Ising spin glass on a lattice with nearest-neighbor interactions, free boundary conditions and no magnetic field amounts to solving a max-cut problem in a planar graph which is polynomially solvable. Fast programs exist in practice [103]. The two-dimensional Ising spin glass with periodic boundary conditions, no external magnetic field and $\pm J$ interactions [109] is a polynomial problem. More generally, it remains polynomial if the genus of the graph is bounded by a constant and the sizes of the integral edge



(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.

weights are bounded in absolute value by a polynomial in the size of the graph [39]. For (unbounded) Gaussian distributed couplings the question is still open. As soon as an external field is present, the problem becomes $\mathcal{NP}$-hard for all kinds of spin interactions [7]. Furthermore, the Ising spin glass in three dimensional grids is $\mathcal{NP}$-hard [7]. In this thesis we will mainly be concerned with hard instances of the max-cut problem. In the following section we explain the framework of a branch–and–cut algorithm that can be used for determining optimum solutions of hard instances.

1.2 How to Calculate Maximum Cuts: a Framework

A branch–and–cut–algorithm for max-cut was started by Michael Jünger, Gerhard Reinelt, and Giovanni Rinaldi. Over the years, there have been several coworkers, Caterina De Simone, Martin Diehl, and Petra Mutzel. At present it is implemented using the C++ library ABACUS [2] that provides a branch–and–cut framework. The running time of the branch–and–cut algorithm for max-cut depends exponentially on the size of the input, in the worst case. As explained above, we cannot expect to do better than that for the hard instances that are of interest here. However, we will see later that with a branch–and–cut approach medium sized problems can be solved within reasonable time in practice. For a recent survey we refer to [76].

The framework of the branch–and–cut algorithm is as follows. For an instance, we always maintain an upper (ub) and a lower bound (lb) for the optimum solution

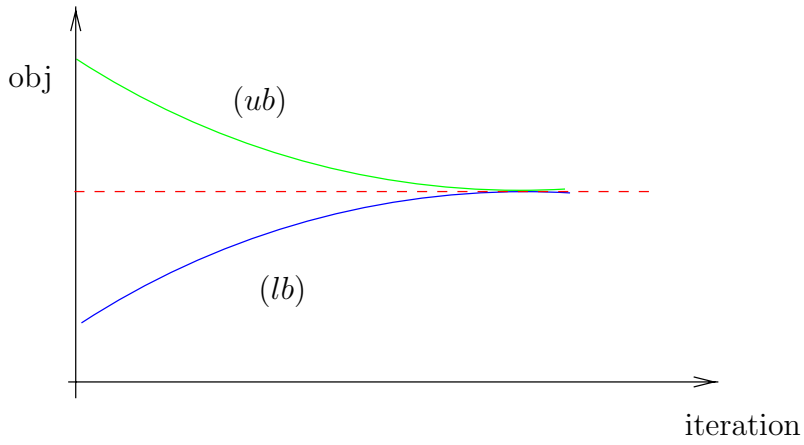


Figure 1.2: Iterative improvement of upper and lower bounds in a branch–and–cut algorithm.

value of the maximum cut. Iteratively we improve upper and lower bounds until they coincide at the optimum solution value or are tight enough for proving optimality, see Figure 1.2. The existence of the upper bound marks the difference between an approximate and an exact solution method, and we explain the determination of the upper bounds (ub) in more detail.

Let $\delta(W)$ be the cut associated with node set $W \subseteq V$. The incidence vector $\chi^{\delta(W)} \in \mathbb{R}^m$ is defined by

$$\chi_e^{\delta(W)} = \begin{cases} 1 & \text{if } e \in \delta(W), \\ 0 & \text{otherwise.} \end{cases}$$

Let the cut polytope $P_C(G)$ be defined as the convex hull of all incidence vectors of cuts in G, i.e.,

$$P_C(G) = \text{conv}\{\chi^{\delta(W)} \mid \delta(W) \text{ is a cut in } G\}. \tag{1.5}$$

The smallest interesting example consists of a graph that is a triangle, see Figure 1.3.

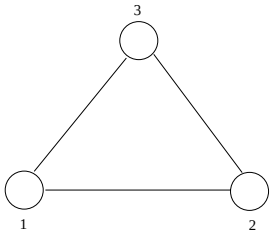


Figure 1.3: A graph consisting of a triangle.

The set of characteristic cut vectors for the graph in Figure 1.3 is

$$\left\{ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \right\}, \tag{1.6}$$

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

In Figure 1.4[1] we show the cut polytope $P_C(K_3)$, i.e., the convex hull of the cut vectors (1.6) for the triangle graph of Figure 1.3.

Figure 1.4: The cut polytope $P_{\text{cyc}}(K_3)$ in the cube $[0,1]^3$.

Coming back to the general case, as the cut vectors are 0-1 vectors, the vertices of $P_C(G)$ are exactly the incidence vectors of cuts in G. Hence, we can solve the maximum cut problem by solving the optimization problem

$$\begin{aligned} (MC) \quad \max \quad & c^{\mathrm{T}} x \\ & x \in P_C(G) \end{aligned} \tag{1.7}$$

Unfortunately, we don't know how to input '$x \in P_C(G)$' efficiently into a computer program, when $P_C(G)$ is defined as the convex hull of a set of points. However, from theorems by Minkowski and Weyl we know that there exists a matrix A and a vector b with

$$P_C(G) = \{x \mid Ax \leq b\}. \tag{1.8}$$

In theory, by applying the Fourier-Motzkin algorithm we could transform the representation (1.5) to representation (1.8). However, for practical problems the number of needed inequalities is too large to be generated explicitely. Additionally, Karp and Papadimitrou [59] showed that no computationally tractable complete linear

[1] Thanks to Constantin Hellweg and Ramin Sahamie for generating this figure.

description of an NP-complete combinatorial optimization problem can be found, unless $\mathcal{P} = \text{co-}\mathcal{NP}$, which seems to be unlikely.

Despite these negative results, we can make use of *partial linear descriptions* of a polytope within the branch–and–cut approach. We informally summarize its basic concepts. We need a partial description P of $P_C(G)$ with the properties that the latter is contained in P and that the inequality system $A_P x \leq b_P$ describing P is known and can be generated 'fast', i.e., in polynomial time. By optimizing over the superset P instead over $P_C(G)$ we obtain an upper bound (*ub*) on the value of the maximum cut. Optimizing over P amounts to solving a *linear program (lp)* which is in general of the form

$$\begin{aligned} \max \quad & c^{\mathrm{T}} x \\ A_P x \quad & \leq b_P \\ x \quad & \geq 0 \end{aligned}$$

Fast algorithms for solving linear programs exist, e.g., the simplex method.

Within the branch–and–cut approach, we start with some partial description P of $P_C(G)$. Iteratively we improve P. A lower bound (*lb*) on the optimum value of the maximum cut can be obtained using any heuristic generating a cut in a graph. In case upper and lower bounds coincide or the upper bound solution vector represents a cut, we can stop and return an optimum solution. However, it is possible that we neither can generate an optimum solution nor a better description $P \supseteq P_C(G)$. In this case we *branch*. In a branching step we select a variable x_{ij} for $(i, j) \in E$ that is neither zero nor one in the upper bound solution vector and generate two sub problems in one of which x_{ij} is set to 0, and in the other to 1. Through subsequent branching steps, a tree of sub problems (the *branch–and–cut tree*) emerges. We call a sub problem a *node* of the tree. The outline of the branch–and–cut framework is summarized in Algorithm 1.

Algorithm: *branch–and–cut-framework*

1 start with some $P \supseteq P_C(G)$
2 solve $(ub) = cx^\star = \max\{cx \mid x \in P\}$
3 (lb): value of a cut found heuristically
4 **if** $(ub) = (lb)$ *or* $x^\star$ *is a cut* **then**
stop
5 **else**
find a better description P of $P_C(G)$
go back to 2
6 **if** *no better description can be found* **then**
branch: select a variable x_{ij} with $x^\star_{ij} \notin \{0; 1\}$
generate two sub problems in one of which x_{ij} is set to 0
and in the other to 1.

Algorithm 1: Framework of a branch–and–cut algorithm for max-cut.

In the following section we describe how we can generate partial descriptions $P \supseteq P_C(G)$.

1.3 Known Facets of the Cut Polytope

We start with a definition on valid inequalities, faces and facets of a polytope.

Definition 1.1 (Validity, Faces, Facets). *(i) Given a polytope $P = \{x \mid Ax \leq b\}$, an inequality $ax \leq a_0$ is called a* valid inequality*, if $a\bar{x} \leq a_0$ for all feasible solutions $\bar{x} \in P$.*

(ii) If $ax \leq a_0$ is valid and the intersection of the affine subspace $H = \{x \mid ax \leq 0\}$ with the polytope P is both not empty and not equal to P, $H \cup P$ is called a face *of P.*

(iii) Let the dimension of P be d. The faces of maximum dimension of P, namely $d-1$, are called facets *of P. If $P \cup \{x \mid ax = a_0\}$ is a facet of P, the inequality $ax \leq a_0$ is called* facet defining inequality *for P.*

Hence, for showing that $ax \leq a_0$ is a facet of P, it is enough to determine d affinely independent incidence vectors $x_1, \ldots, x_d$ at which the inequality is *tight*, i.e., $ax_i = a_0$ for $i = 1, \ldots, d$.

In the following we explain how we can determine partial descriptions of the cut polytope in step 5 of the branch–and–cut Algorithm 1. These partial descriptions are called *relaxations*. To be more specific, let the cut polytope of a graph G be contained in some polytope P. Then the problem

$$(MC_R) \quad \begin{aligned} \max \quad & c^{\mathrm{T}}x \\ & x \in P \end{aligned} \tag{1.9}$$

is called a *relaxation* of the max-cut problem (1.7). We sometimes also call P itself a relaxation.

The basic idea in determining progressingly tighter relaxations of $P_C(G)$ is as follows. We start by generating some 'easy' relaxation P (e.g., the unit hypercube $[0 \dots 1]^{|E|}$.) Let the optimum solution of (MC_R) be $c^{\mathrm{T}}x^\star$. We test whether $x^\star$ satisfies inequalities that are known to be valid for $P_C(G)$. In case we can generate a valid inequality that is violated by $x^\star$, we can add it to the description of P. By adding this inequality to P we 'cut off' $x^\star$ from it, and these inequalities are sometimes called *cutting planes*. The resulting polytope will be a 'tighter' relaxation of $P_C(G)$ than P was and hence improve the upper bound. We also say we *separate* $x^\star$ from the cut polytope. The corresponding problem is called the *separation problem*. As we usually don't know how to solve the separation problem for valid inequalities of any kind, we study it for each *class* of inequalities separately. Before we introduce classes of inequalities valid for the cut polytope, we formally define the separation problem in Definition 1.2.

Definition 1.2 (Separation Problem). *Given a class of valid inequalities for $P_C(G)$ and a vector $x^\star \in \mathbb{R}^m$, either prove that $x^\star$ satisfies all inequalities of this class, or return an inequality violated by $x^\star$.*

An algorithm that solves the separation problem is called *exact separation algorithm*. Unfortunately, such an algorithm is often not known for a class of valid inequalities or it can be shown that solving the separation problem is an $\mathcal{NP}$-hard problem. In this case we have to use a *heuristic separation algorithm*. Heuristic separation algorithms may find violated inequalities, but maybe not all.

Next we introduce classes of inequalities that are valid or facet defining for the cut polytope. Given a graph $G = (V, E)$, an incidence vector χ of a cut obviously has to satisfy

$$0 \leq x_e \leq 1 \; \forall e \in E \tag{1.10}$$

Barahona and Mahjoub proved the following lemma.

Lemma 1.3 (Barahona and Mahjoub [13]). *The 'trivial' inequality (1.10) defines a facet of the cut polytope $P_C(G)$ if and only if e does not belong to a triangle.*

A proof of this lemma can be found in [13].

A class that is more interesting than the trivial inequalities are the *cycle inequalities*. An edge set $C = \{(v_0, v_1), (v_1, v_2), \dots, (v_{k-1}, v_0)\} \subseteq E$ is called a *cycle* (of length

k) in the graph $G = (V, E)$. Let $C \subseteq E$ be a cycle and e be an edge that does not belong to C. We say e is a *chord* of C if it joins two nodes of C. A cycle C is called *chordless* if it does not contain a chord. The cycle inequalities are drawn from an easy observation:

Observation 1.4. *A cut and a cycle can only have an even number of common edges.*

Barahona and Mahjoub proved the following results on the cycle inequalities.

Lemma 1.5 (Barahona and Mahjoub [13]). *Let $C \subseteq E$ be a cycle in $G = (V, E)$ and $F \subseteq C$ a subset of cycle edges of odd cardinality. Then the* cycle inequality

$$\sum_{e \in F} x_e - \sum_{e \in C \setminus F} x_e \leq |F| - 1 \tag{1.11}$$

is valid for the cut polytope $P_C(G)$. In case C is chordless, (1.11) is facet definining for $P_C(G)$.

As the cycle inequalities play an important role in the computation of exact ground states we give a proof of Lemma 1.5.

Proof. The validity of (1.11) is easy to see. Let an arbitrary cut $\chi \in \mathbb{R}^m$ and an arbitrary cycle inequality be given. In case there exists an edge $e \in F$ with $\chi_e = 0$, (1.11) is obviously satisfied. Otherwise it is $\chi_e = 1$ for all $e \in F$. As $|F|$ odd, we know from Observation 1.4 that for at least one other edge in $C \setminus F$ it has to be $\chi_e = 1$, too. Thus, (1.11) is satisfied. We postpone the facet defining property until Section 2.5 where we will prove it as an application for the lift-project approach.

Let C have a chord f. f partitions C into two paths P_1, P_2 such that $C_i := P_i \dot{\cup} \{f\}$ are cycles for $i = 1, 2$. It is not hard to see that any cycle inequality defined on C can be written as the sum of two appropriately chosen cycle inequalities, one defined on C_1, the other on C_2. Hence, cycle inequalities defined on cycles with a chord are not facet defining.

□

The *triangle inequalities*

$$\begin{aligned} x_{ij} + x_{ik} + x_{jk} &\leq 2 \\ x_{ij} - x_{ik} - x_{jk} &\leq 0 \\ -x_{ij} + x_{ik} - x_{jk} &\leq 0 \\ -x_{ij} - x_{ik} + x_{jk} &\leq 0 \end{aligned} \tag{1.12}$$

for a triangle i, j, k in G are a special case of the cycle inequalities for $|C| = 3$ and define facets of $P_C(G)$. (This is easy to see as the inequalities are tight at the affinely independent incidence vectors ${}^T(x_{ij}, x_{ik}, x_{jk}, \ldots) =$

$\{^T(0,1,1,\ldots),^T(1,0,1,\ldots),^T(1,1,0)\}$.) The cycle polytope of K_p, where K_p is the complete graph on p nodes, consists of all $4\binom{p}{3}$ triangle inequalities.

Together with Lemma 1.5 and the observation that 0/1-vectors satisfying the cycle inequalities are incidence vectors of cuts, we get an integer linear programming formulation of the max-cut problem.

$$\begin{aligned}(MC)\quad \max\{c^T x \mid\ & x(F) - x(C\setminus F) \le |F| - 1 \text{ for each } F \subseteq C,\ |F| \text{ odd},\\ & \text{for each cycle } C \text{ in } G,\\ & 0 \le x_e \le 1 \text{ for each } e \in E,\\ & x_e \text{ integer for each } e \in E\}\end{aligned} \tag{1.13}$$

For making use of the cycle inequalities within branch–and–cut we have to solve the *separation problem for the cycle inequalities*. We formulate it as follows.

Definition 1.6 (Separation Problem for the Cycle Inequalities). *Given $x^\star \in \mathbb{R}^m$ with $0 \le x^\star_e \le 1$ for all edges $e \in E$, decide whether $x^\star$ satisfies all cycle inequalities. If not, return an inequality (1.11) violated by $x^\star$.*

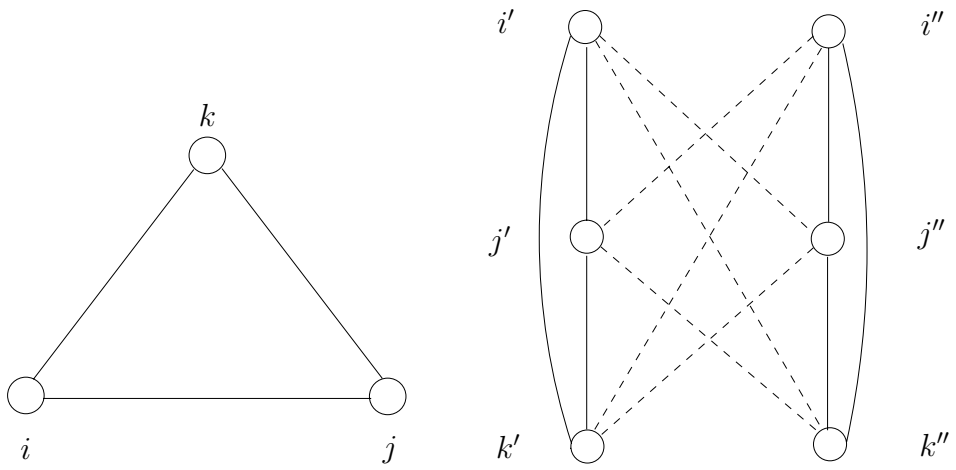
Barahona and Mahjoub also proved in [13] that this separation problem can be solved in polynomial time. To this end, we make use of the fact that we can rewrite a cycle inequality (1.11) as

$$\sum_{e\in C\setminus F} x_e + \sum_{e\in F}(1-x_e) \ge 1. \tag{1.14}$$

We generate a graph H that consists of two copies of the graph G, $G' = (V', E')$ and $G'' = (V'', E'')$, together with some additional edges. $u' \in V'$, $u'' \in V''$ denote the two copies of a node $u \in V$. Let $H = (V' \cup V'', E' \cup E'' \cup E''') = (V^H, E^H)$. In addition to the edges $(u', v') \in E'$ and $(u'', v'') \in E''$ present in the two copies G' and G'', there are extra edges in E^H. For each edge $(u, v) \in E$, the two edges (u', v'') and (u'', v') are in E^H. The weight of $(u', v') \in E'$ and $(u'', v'') \in E''$ is chosen as $x^\star_{uv}$, while the weight of (u', v''), $(u'', v') \in E'''$ is set to $1 - x^\star_{uv}$. In Figure 1.5 we show a graph consisting of a triangle and the corresponding graph H.

The separation procedure works as follows. For each pair of nodes $u', u'' \in W$ we calculate a shortest path in H. We choose the edge weights as defined above. According to the definition of H, such a path contains an odd number of edges of E'''. It corresponds to a closed walk in G that contains u. Thus, the minimum of over all nodes u of G of these shortest paths gives the minimum value of $\sum_{e\in C\setminus F} x_e + \sum_{e\in F}(1-x_e)$, the left hand side of (1.14). If the shortest of these (u', u'')-paths in H has length at least 1, all cycle inequalities are satisfied. Otherwise, a shortest path of length less than 1 corresponds to a violated cycle inequality.

Shortest paths can efficiently be computed in polynomial time. Thus, the separation problem for the cycle inequalities is polynomially solvable. Furthermore, a special



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

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

Figure 1.5: Separation of the cycle inequalities.

case of the famous result of Grötschel, Lovász and Schrijver [43] says that we can optimize over a relaxation within polynomial time if and only if we can solve the corresponding separation problem in polynomial time. Let the *cycle polytope* $P_{\text{cyc}}(G)$ consist of the vectors in $\mathbb{R}^n$ satisfying all cycle inequalities,

$$\begin{aligned} P_{\text{cyc}}(G) = \{x \in \mathbb{R}^E \mid x(F) - x(C \setminus F) \leq |F| - 1 \text{ for each } F \subseteq C, |F| \text{ odd }, \\ \text{for each cycle } C \text{ in } G, \\ 0 \leq x_e \leq 1 \text{ for each } e \in E\}. \end{aligned} \tag{1.15}$$

We deduce that we can solve within polynomial time the problem

$$(MC_{\text{cyc}}) \quad \max \quad c^{\mathrm{T}}x \tag{1.16}$$

$$x \in P_{\text{cyc}} \tag{1.17}$$

It is interesting to ask how 'tight' the cycle relaxation is for the cut polytope. If theory is concerned, one answer is given in the next lemma. In practice, the cycle relaxation is a 'tight' relaxation for Ising spin-glass instances, see Section 2.2.

Lemma 1.7 ([8]). *The cycle polytope $P_{\text{cyc}}(G)$ of a graph $G = (V, E)$ is equal to the cut polytope $P_C(G)$ if and only if G does not have any K_5-minor.*

Seymour proved the above result for the cut cone. Barahona and Mahjoub proved Lemma 1.7. As we can optimize over the cycle polytope in polynomial time, we have as a corollary:

Lemma 1.8. *The maximum cut problem can be solved in polynomial time for the class of graphs with no K_5-minor.*

Kuratowski's theorem says that planar graphs are exactly the graphs that do not contain K_5- and $K_{3,3}$-minors, where $K_{3,3}$ is the complete bipartite graph consisting of three nodes in each shore. Thus, the maximum cut problem can be solved in polynomial time for planar graphs.

Despite the fact that cycle inequalities can be generated in polynomial time, it turns out in practice that the exact separation routine needs much CPU time. (In Chapter 2 we will explain how the cycle relaxation can be generated fast for Ising spin-glass instances.) Thus, it is favorable to include heuristic separation routines. Already in the first version of the max-cut algorithm for spin-glass instances, heuristics were used for generating violated cycle inequalities [10]. When heuristically separating cycle inequalities, we sometimes encounter the following problem: We are given a cycle C and $x^\star \in \mathbb{R}^m$ to be separated. We want to determine a cycle inequality (1.11) on C that is maximally violated by $x^\star$. Let the violation v be defined as

$$v = \sum_{e \in F} x_e^\star - \sum_{e \in C \setminus F} x_e^\star - |F| + 1.$$

We want to solve the problem

$$\max v = \max_{F \subseteq C, |F| \text{ odd}} \sum_{e \in F} (x_e^\star - 1) - \sum_{e \in C \setminus F} x_e^\star \tag{1.18}$$

We solve (1.18) algorithmically. An edge e in F contributes an amount $x_e^\star - 1$ to the violation v. An edge e in the set $C \setminus F$ contributes $x_e^\star$ to v. We can solve problem (1.18) in $\mathcal{O}(|C|)$ time by assigning an edge e to F in case $\tilde{c}_e := 2x_e^\star - 1 > 0$ and to $C \setminus F$ otherwise. After having assigned each cycle edge to either F or $C \setminus F$, we might end up with a set F having even cardinality. Then we have to either delete the edge from F with minimum $\tilde{c}_e$ or to insert to F the edge from $C \setminus F$ with maximum $\tilde{c}_e$, whichever is better. In function *best_ineq_on_given_cycle* we summarize the procedure. In Chapter 2.2.2 we will use this algorithm.

Algorithm: *best_ineq_on_given_cycle*
$F := \emptyset$
for *all edges* e *in* C **do**
  $\tilde{c}_e = 2x^\star_e - 1$
  if $\tilde{c}_e > 0$ **then**
    $F = F \cup \{e\}$
if $|F|$ *even* **then**
  determine $e_1 \in C \setminus F$ with maximum $\tilde{c}_e$
  determine $e_2 \in F$ with minimum $\tilde{c}_e$
  if $|\tilde{c}_{e_1}| < |\tilde{c}_{e_2}|$ **then**
    $F = F \cup \{e\}$
  else
    $F = F \setminus \{e\}$

Algorithm 2: *best_ineq_on_given_cycle* generates the best (with regard to violation) inequality on a given cycle.

Next we introduce the *bicycle-wheel* inequality. To this end, let a graph G consist of a cycle of length p and two nodes adjacent to each other and to every node of the cycle. Then G is called a *bicycle p-wheel*. Barahona and Mahjoub proved the following lemma.

Lemma 1.9 (Barahona and Mahjoub [13]). *Let (W, B) be a bicycle $(2k+1)$-wheel, $k \geq 1$, contained in G. Then the inequality*

$$x(B) \leq 2(2k+1) \tag{1.19}$$

defines a facet of the cut polytope $P_C(G)$.

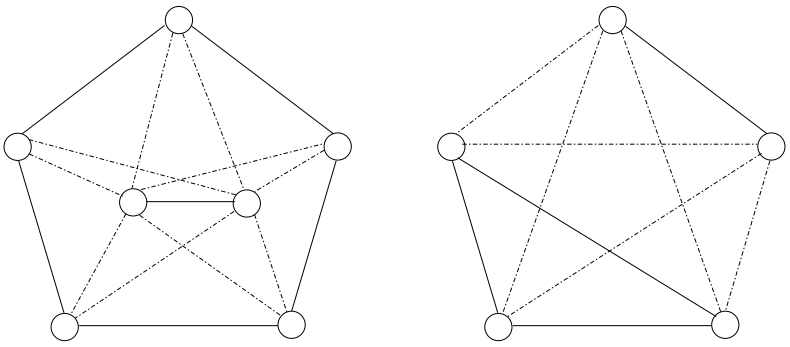
See Figure 1.6(a) for a bicycle 5-wheel together with a cut of maximum cardinality satisfying the inequality with equality. Gerards [40] has shown that the class of bicycle wheel inequalities can be separated in polynomial time by an algorithm similar to the cycle separation procedure with different choice of the edge weights.

Another well-known class of facets for the cut polytope are the *clique*-inequalities, with complete graphs K_p on p nodes as support.

Lemma 1.10 (Barahona and Mahjoub [13]). *Let $K_p = (W, E_p)$ be a complete subgraph of order p of G. Then the K_p-*inequality

$$x(E_p) \leq \left\lceil \frac{p}{2} \right\rceil \left\lfloor \frac{p}{2} \right\rfloor \tag{1.20}$$

is valid for $P_C(G)$. (1.20) defines a facet of the cut polytope $P_C(G)$ if and only if p is odd.



(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 1.6: Valid inequalities for the cut polytope.

The separation of K_p-inequalities is $\mathcal{NP}$-hard which is obvious as also the determination of a clique of maximum size in a graph is $\mathcal{NP}$-hard. In Section 2.5 we will use the bicycle wheel and the clique inequalities inside the branch–and–cut framework within a lift-project approach.

The so-called *parachute inequalities* were introduced by Deza and Laurent in [29]. Further results on them can be found in [30]. The parachute inequality is defined on an odd number of points denoted as $\{k, k-1, k-2, \ldots, 1, 0, 1', 2', \ldots, (k-1)', k'\}$. By defining the path P as $P = (k, k-1, \ldots, 1, 1', \ldots, (k-1)', k')$, we can formulate the parachute inequality as

$$(\mathrm{Par}_{2k+1})x = \sum_{i,j \in P} x_{ij} - \sum_{1 \le i \le k-1} (x_{0i} + x_{0i'} + x_{ki} + x_{k'i}) - x_{kk'} \le 0 \qquad (1.21)$$

In Figure 1.7 we show the support graph of the parachute inequality on seven points. Inequality $(\mathrm{Par}_7)x \le 0$ has coefficient $+1$ on the solid edges and -1 on the dashed edges.

We call path P the *parachute*, the edges between different non neighboring nodes in P the *support edges*, node '0' the *jumper*, edges between the jumper and the nodes along the parachute *jumper edges*. For k even, the parachute inequality is not even valid for the cut cone. Consider, e.g., the cut $\delta(\{1, 3, \ldots, k-1\} \cup \{2', 4', \ldots k'\})$. All $2k-1$ edges along the parachute are cut edges, but only $k-1$ jumper edges and $k-1$ support edges are cut edges. Thus, we have $2k - 1 - (k-1) - (k-1) \not\le 0$.

However, for odd k, inequality $(\mathrm{Par})_{2k+1}$ is valid and facet defining and we can formulate the following theorem.

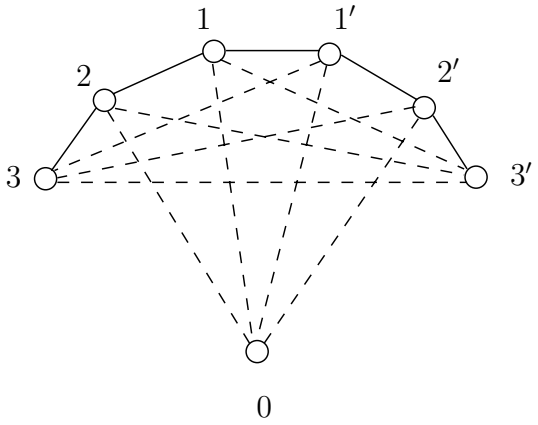


Figure 1.7: The parachute inequality on 7 nodes. Coefficients on solid edges are 1, on dashed edges -1 in $(\mathrm{Par}_7)x \leq 0$.

Theorem 1.11. *The parachute inequality (1.21) $(Par_{2k+1})x \leq 0$ is valid and facet defining for odd $k \in \mathbb{N}$, $k \geq 3$.*

Another interesting class of inequalities are defined on *circulants*.

Definition 1.12. *A circulant $C(n, r)$ is a graph consisting of n nodes $1, \ldots, n$ and the edges $(i, i+1), (i, i+r)$ for all nodes $i = 1, \ldots, n$ with the indices taken modulo n.*

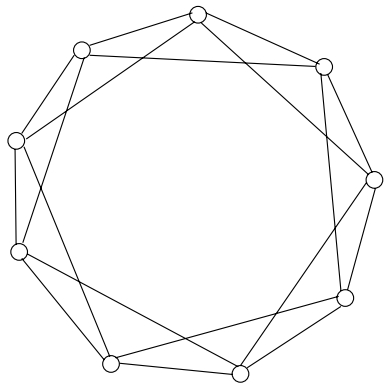


Figure 1.8: The circulant $C(9, 2)$.

In Figure 1.8 we show the circulant $C(9, 2)$.

Poljak and Turzik (1992) presented an $\mathcal{O}(r \log^2 n)$ algorithm for computing a maximum cut in a circulant graph $C(n, r)$. The authors introduced the class of circulant inequalities for the cut polytope and showed their validity and the conditions under which they are facet defining.

Theorem 1.13 ([118]). *Let $n = kr + 1$ with $k, r \geq 2$ be even integers. Then the* circulant inequality

$$\sum_{ij \in C(n,r)} x_{ij} \leq 2n - k - r \tag{1.22}$$

defines a facet of the cut polytope.

The authors note that already the separation of the circulant inequalities for circulants $C(n, 2)$ is $\mathcal{NP}$-hard. In practical computations we separate circulant inequalities using a heuristic. In Section 3 we use a heuristic separation of for spin glasses defined on the one-dimensional Ising chain model.

The last class of inequalities that we want to present here are the *hypermetric* inequalities [30].

Lemma 1.14. *Let $b = (b_1, \ldots, b_n)$ an integral vector that satisfies $\sum_{i=1}^{n} b_i = 1$. Then the inequality*

$$\sum_{1 \leq i < j \leq n} b_i b_j x_{ij} \leq 0 \tag{1.23}$$

is valid for $P_C(G)$.

Some hypermetric inequalities are facets, e.g., the triangle inequalities are special cases of the hypermetric inequalities. Separating the hypermetric inequalities is an $\mathcal{NP}$-hard problem and we separate them using heuristics.

All facet defining inequalities of the cut polytope could only be determined for small sizes and are compiled in the SMAPO library [114]. The complete description is known up to $P_C(K_7)$. Much is known about the polyhedral structure of the cut polytope of dense or complete graphs. Structures like bicycle-wheels, cliques etc. are not present in d-dimensional Ising spin-glass instances. However, in Section 2.5 we will use the facets known for the complete graph for sparse spin-glass instances within a lift-project approach.

In the last paragraph of this section we introduce the *switching map* for incidence vectors of cuts. See [30] for the details on the switching operation. This operation will be helpful in subsequent chapters. The set of cuts in a graph G is closed under taking symmetric differences. This means that the symmetric difference of the cuts $D \in \mathcal{D}$ and $D' \in \mathcal{D}$ in G, denoted by $D \triangle D'$ (i.e., the set of edges that belong to one of the two cuts but not to both) is a cut in G. We can formulate this property in algebraic terms as follows. Let χ^D and $\chi^{D'}$ be the characteristic vectors of D and D', respectively. Then the map

$$s^D : \mathbb{R}^E \longrightarrow \mathbb{R}^E \tag{1.24}$$

is called *switching along the cut* D and is defined by

$$s^D(\chi^{D'})|_e = \begin{cases} \chi_e^{D'} & \text{if } e \notin D, \\ 1 - \chi_e^{D'} & \text{if } e \in D. \end{cases} \tag{1.25}$$

With (1.25) we map the characteristic vector of a cut D' in G into the characteristic vector of another cut in G. Whenever we generate an inequality valid for the cut polytope we can switch it along an appropriate cut and obtain another valid inequality. In formulas, if $ax \leq a_0$ is valid for $P_C(G)$ and D a cut in G, then $\sum_{e \notin D} a_e x_e - \sum_{e \in D} x_e \leq a_0 - \sum_{e \in D} a_e$ is the corresponding valid inequality switched along the cut D. Dimensionality of the faces is preserved by switching, i.e., switching a facet of $P_C(G)$ yields another facet of $P_C(G)$.

One application of the switching map is as follows. Assume we have generated a valid inequality $ax \leq a_0$ violated by $x^\star$ by the amount $v > 0$, i.e. $ax^\star = a_0 + v$ within the branch–and–cut framework. We want to add the violated inequality to the current polytope P and obtain a tighter relaxation of $P_C(G)$ and a better upper bound. We assume that the bigger the violation v is, the better the improvement in the bound is. So we aim at determining an optimum switching that maximizes v. To this end, we have to solve the optimization problem

$$\max v = \max_{W \subseteq V} \sum_{e \in \delta(W)} a_e(1 - x_e^\star) + \sum_{e \notin \delta(W)} a_e x_e^\star. \tag{1.26}$$

We write (1.26) as

$$\begin{array}{l} \max \sum_{e \in E} a_e(1 - x_e^\star)x_e + \sum_{e \in E} a_e x_e^\star(1 - x_e) \\ \quad x \in P_C(G) \end{array} \tag{1.27}$$

which is a max-cut problem on the support of the inequality under consideration. (1.27) is equal to

$$\begin{array}{l} \max \sum_{e \in E} a_e(1 - 2x_e^\star)x_e \\ \quad x \in P_C(G) \end{array} \tag{1.28}$$

We know how to solve the max-cut problem (1.28) exactly. However, in practical computations it does not pay off to determine a maximum cut on the support of an inequality only for determining one maximally violated inequality. Hence, we restrict ourselves to determining a 'good' switching yielding fast a 'well' (maybe not optimally) violated inequality. We do this with a so-called GRASP heuristic (Greedy Randomized Adaptive Search Procedure.) GRASP heuristics are fast and usually give better results in practice than simple greedy algorithms, see e.g. [33]. The key idea in these heuristics is as follows: In a greedy algorithm we take in each step the locally best choice. In a grasp heuristic instead, we generate the k best choices and randomly take one of them. For solving our problem (1.28) we color the nodes, say

red and black. We start with all nodes colored red and cut value zero. The score of each node is the amount the cut changes when changing its color. As long as we can improve the cut, we determine the k nodes with maximum score. We choose randomly one of them, say node i, and change the color of i from red to black or vice versa. We summarize the procedure in Algorithm 3.

Algorithm: *good_switching*

color all nodes red
for *all nodes* $i \in V$ **do**
    determine score$(i) = \sum_{(i,u)\in E}(1 - 2x^{\star}_{iu})a_{iu}$
while *cut can be improved* **do**
    from the k nodes with best scores determine a node i randomly
    change color of i
    update scores

Algorithm 3: Generating a 'good' switching with a GRASP heuristic for improving the violation of an inequality.

Chapter 2

Branch–and–Cut for Sparse Max-Cut Instances

During the run of the branch–and–cut algorithm we generate and solve tighter and tighter relaxations of the max-cut problem until we can prove optimality of a known cut. It is important for the performance of branch–and–cut that the chosen relaxation can be generated 'fast' in practice and that it yields a 'tight' approximation of the cut polytope, four our class of instances. In this chapter we study what kind of relaxation gives the best results within a branch–and–cut framework and how it can be solved.

In the first section we experimentally compare different relaxations for the max-cut problem, finding that the cycle relaxation gives the best results for spin-glass instances. Then we explain how we can heuristically generate the cycle relaxation fast for regular grid graphs and give experimental results. Subsequently, we show that the overall performance of branch–and–cut is better when the cycle relaxation is solved by the traditional simplex method than by subgradient- or interior point methods. Next we tighten the cycle relaxation by adding inequalities beyond the cycles. Finally, we study the question how good cuts can be generated.

2.1 Choosing a Relaxation for Max-Cut

Within branch–and–cut, different choices for relaxing the integer programming formulation (1.13) of the max-cut problem are possible. We might want to use a *linear* relaxation based on the cycle polytope. This *cycle relaxation* can be determined efficiently both in theory and in practice, see the polynomial separation routine explained in Section 1.3. For spin-glass instances, the cycle relaxation is a 'tight' approximation of the cut polytope, and usually the optimum over the cycle polytope is only a few percents away from the optimum cut value. Another possible

relaxation based on *positive semidefinite optimization* is of *quadratic* nature. While Miguel Anjos was doing a Post Doc at our institute in Cologne, we studied the power of positive semidefinite optimization for spin glasses. Results presented in the corresponding section are joint work with him. The motivation for using positive semidefinite optimization lies in the fact that the optimum over the quadratic relaxation is known to be at most 14% away from the value of a maximum cut in case the edge weights are nonnegative. No quality guarantee is known for linear relaxations. So it is an interesting question whether we can strengthen the relaxation by using positive semidefinite optimization. In the following section, we compare the cycle relaxation with the quadratic relaxation. We find that the quadratic relaxation has only limited power for spin glasses making the linear our relaxation of choice.

First we introduce the quadratic relaxation. Let a weighted graph $G = (V, E)$ with weight $c_{ij} \in \mathbb{R}$ for edge $(i, j) \in E$ be given. For a node set $W \subset V$, let $v \in \{\pm 1\}^{|V|}$ with

$$v_i = \begin{cases} 1 & \text{if } i \in W, \\ -1 & \text{otherwise.} \end{cases}$$

Then the cut $\delta(W)$ corresponds to the edges $(i, j) \in E$ for which $v_i v_j = -1$. The *Laplace matrix* L of a graph G is defined as $L_{ij} = -c_{ij}$ for $(i, j) \in E$, $L_{ii} = \sum_{j:(i,j)\in E} c_{ij}$ for $i = 1, \ldots, n$, and $L_{ij} = 0$ otherwise. The max-cut problem can be formulated as the quadratic problem

$$\begin{aligned} (MC) \qquad & z_{\text{MC}} = \tfrac{1}{4} \max v^T L v \\ & v \in \{-1, 1\}^{|V|}. \end{aligned}$$

We notice that $v^T L v = L \bullet (vv^T)$, where we define for two matrices A, B $A \bullet B := \sum_{ij} a_{ij} b_{ij}$. It is not hard to see that the matrices of the form aa^T, with $a \in \{-1, 1\}^{|V|}$, are exactly the positive semidefinite matrices X (denoted as $X \succeq 0$) of rank one having entry 1 along the main diagonal. The basic positive semidefinite (SDP) relaxation (MC_{SDP}) of max-cut is obtained by dropping the rank-one condition:

$$\begin{aligned} (MC_{SDP}) \quad z_{SDP} = \quad & \frac{1}{4} \max L \bullet X \\ & X_{ii} = 1 \; \forall \; i = 1, \ldots, |V| \\ & X \succeq 0 \end{aligned}$$

In 1994 Goemans and Williamson [41] showed that it is possible to give a quality guarantee of the relaxation (MC_{SDP}). If all edge weights are nonnegative, the op-

timum solution z_{SDP} of (MC_{SDP}) is at most 14% away from the optimum value of the maximum cut. If negative edge weights are present, it is

$$z_{MC} - C^- \geq 0.878(z_{SDP} - C^-),$$

where $C^- = \sum_{(i,j)\in E, c_{ij}<0} c_{ij}$ is the sum of the negative edge weights in G. For a $\pm J$ Ising spin glass, the sum of the negative weights is approximately $-\frac{|E|}{2}$. We have

$$z_{SDP} \lessapprox 1.14 z_{MC} + 0.07|E|.$$

In the same paper [41] Goemans and Williamson also derived a randomized approximation algorithm based on positive semidefinite optimization with the same approximation guarantee. This means the algorithm yields a cut that is at most 14% away from the optimum solution. The algorithm can be derandomized. This result was the first to show that it is possible to derive an algorithm with an approximation guarantee for a hard problem like max-cut with the use of positive semidefinite optimization. Since then, much research has focused on the latter.

We experimentally determine the quality of the basic SDP relaxation (MC_{SDP}) by comparing it with optimizing over the cycle polytope (*cycle relaxation*). For estimating the power of the positive semidefinite relaxation for spin-glass instances, we also compute a relaxation of max-cut that is intermediate between the linear and the quadratic one. For an arbitrary instance, we add to the basic relaxation (MC_{SDP}) all cycle inequalities that are necessary for defining a point in the cycle polytope. We call the intermediate relaxation *SDP & cycles*. Positive semidefinite problems can be solved exactly through the *NEOS Server for Optimization* [96].

In Table 2.1 we give some results on the optimum value of the different relaxations for three-dimensional ± 1 distributed instances in the naming convention `s<dim><linearsize>_<random seed>`. In practice, the optimum solution of (MC_{SDP}) lies roughly 13% above the optimum of the cyle relaxation which is a considerable amount.

Instance	z_{cyc}	z_{SDP}	$z_{\text{SDP\&cyc}}$
s6_111	190.985	215.915	190.983
s8_111	470.736	532.048	470.672
s10_111	912.268	1030.237	912.265

Table 2.1: Comparison of the cycle relaxation bound z_{cyc}, the basic SDP bound z_{SDP} and the bound $z_{\text{SDP\&cyc}}$ derived through the intermediate relaxation SDP & cycles.

The bounds z_{cyc} and $z_{\text{SDP\&cyc}}$, reported in the first and third column respectively, are almost equal. In the SDP & cycles relaxation the set of feasible solutions consists of the intersection of the cycle polytope with the cone of positive semidefinite matrices.

From the numbers in Table 2.1 we conclude that performing this intersection has almost no effect on the quality of the cycle relaxation bound and does not strengthen it significantly.

In Table 2.2, we report the running times needed for determining the results in Table 2.1 in seconds. The cycle relaxation is computed on a 296 MHz SUN workstation. The NEOS server only reports the needed real time making a running time comparison difficult. Whereas the determination of the basic SDP relaxation is quite fast, the running times for solving the SDP & cycles relaxation are higher than solving the cycle relaxation. (We take into account that probably the NEOS machines are faster than the machine we used.)

Instance	CPU z_{cyc}	time z_{SDP}	time $z_{\text{SDP\&cyc}}$
t3pm6_111	33	25	124
t3pm8_111	620	2040	2542
t3pm10_111	3323	1177	5466

Table 2.2: Running times for determining the bounds in Table 2.1.

The running times for solving the quadratic relaxation SDP & cycles is not better than for solving the linear relaxation. Additionally the improvement in the bound introduced by positive semidefiniteness is only marginal. We conclude that we cannot improve the cycle relaxation by using positive semidefinite optimization. Hence, it is best to generate a linear relaxation of the max-cut problem inside the branch–and–cut framework. In the following section we show how this can be done fast for regular grid graphs.

2.2 Generating the Cycle Relaxation

For convenience, we repeat the formulation of the cycle inequalities from Chapter 1. Let C be a chordless cycle in the graph $G = (V, E)$ and $F \subseteq C$ an odd subset of edges in C. Then the cycle inequality

$$\sum_{e \in F} x_e - \sum_{e \in \subseteq C \setminus F} x_e \leq |F| - 1$$

is a facet of the cut polytope $P_C(G)$, see Lemma 1.5. A polynomial separation routine for cycle inequalities exist (see Chapter 1). However, in practical spin-glass computations calling the exact separation routine needs much CPU time. We are interested in fast heuristics generating promising chordless violated cycles in regular grids. We call a cycle *simple* if a node occurs only once in the cycle and start with two observations.

Lemma 2.1. *Let $d \in \mathbb{N}$ and G be a d-dimensional regular grid with $L_1 \times L_2 \times \cdots \times L_d$ sites and either free or periodic boundary conditions. Let each L_i be an even number. If C is a simple cycle in G, then C has even length.*

In a grid with $L_1 \times L_2 \times \cdots \times L_d$ sites and at least one L_i being odd, cycles of odd length exist, for example the *long torus cycles* consisting of the edges along an (odd) row in the grid. In Figure 2.1 we show a long torus cycle in a 5×2 grid.

Observation 2.2. *The only cycles of odd length occur in a finite grid with $L_1 \times L_2 \times \cdots \times L_d$ sites and periodic boundaries, if at least one of the L_i is odd.*

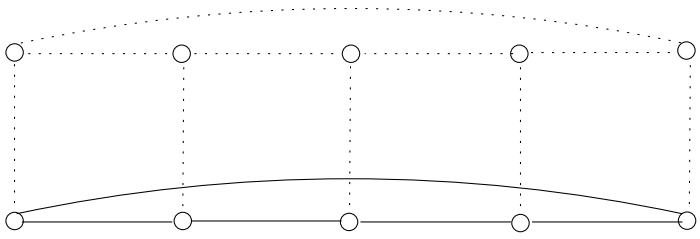


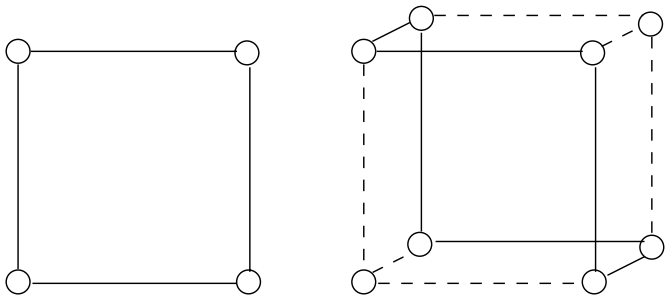
Figure 2.1: A long torus 5-cycle in a 5×2 grid. Solid edges are cycle edges, dashed edges are grid edges that do not belong to the cycle.

Proof of Lemma 2.1. Let $C = (u_1, u_2, \ldots, u_l)$ be a simple cycle in G. We first assume that G has free boundaries. Let us start at a node in C and proceed along the cycle edges. When proceeding along an edge $e = (u_i, u_{i+1}) \in C$, we move from the k-th site in a dimension j to the $(k+1)$-th (or $(k-1)$-th) site in this dimension. For closing the cycle, there must exist another edge in C that moves us from the $(k+1)$-th site in dimension j back to the k-th site (or from the $(k-1)$-th to the k-th site). By

repeatedly applying this argument, C has even length. If G has periodic boundary conditions in a dimension j, then for an edge $e = (u_i, u_{i+1}) \in C$ either the argument above applies. Otherwise, we find $L_j - 1$ cycle edges moving us from the $(i+1)$-th site in dimension j to the $(i+2)$-th site (or from the i-th site to $(i-1)$-th site), from the $(i+2)$-th site to the $(i+3)$-th site etc. until we reach again the i-th site in dimension j. Together with edge e we have to move along L_j edges which is even. By repeatedly applying the arguments, C has even length.

□

For the application to spin glasses, we restrict ourselves to chordless simple cycles occuring in two- and three-dimensional regular grids. It will come out in the following section that small chordless cycles play an important role in practice. In Figures 2.2 and 2.3 we display representatives of the chordless cycles of size less than or equal to ten that are present in two- and three-dimensional grids. We consider long torus cycles or extensions of it later. Solid lines represent cycle edges, dashed edges are grid edges included as guide for the eyes. The smallest possible chordless cycles of length four correspond to the grid plaquettes, see Figure 2.2(a). The chordless six-cycles are layouted along the elementary cubes in a three-dimensional grid, see Figure 2.2(b). In a two-dimensional grid we do not have chordless six-cycles. The chordless cycles of size eight are displayed in Figure 2.3(a). In Figure 2.3(b) we show chordless cycles consisting of ten edges. We show the cycles modulo symmetry.

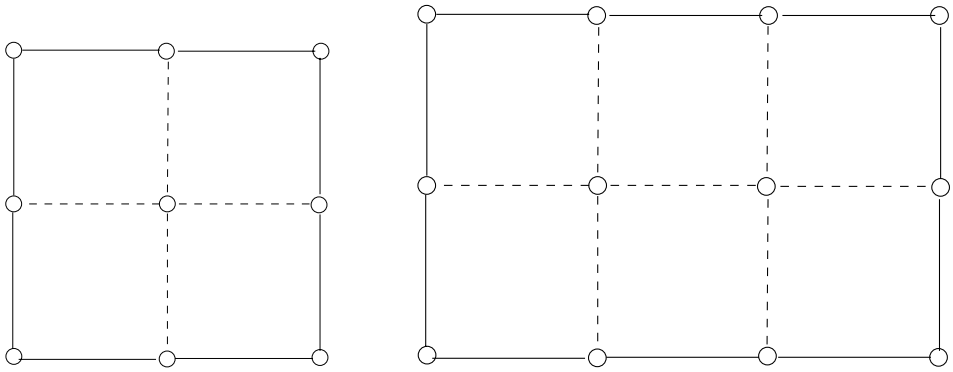


(a) The representative of a chordless four-cycle in a two-dimensional grid

(b) The representative of a chordless six-cycle in a three-dimensional grid

Figure 2.2: Chordless four- and six-cycles occurring in two- and three-dimensional grids.

Now let for a two- or three-dimensional grid G with $L_0 \times L_1$ or $L_0 \times L_1 \times L_2$ sites at least one L_i be odd. Let $L_i^\star$ be the smallest of those. Then the shortest chordless cycles of odd length are the long torus cycles consisting of $L_i^\star$ edges. There are also chordless cycles consisting of $L_i^\star + k$ edges, k even, that extend long torus cycles, see



(a) The representative of a chordless eight-cycle in the grid.

(b) The representative of a chordless 10-cycle in the grid

Figure 2.3: Chordless eight- and 10-cycles.

Figure 2.4, where we show a representative of a long torus cycle of size seven in a $L = 5 \times 2$ grid. We call those chordless cycles *extended long torus cycles*. In Section 2.2.2 we examine extended long torus cycle inequalities in more detail.

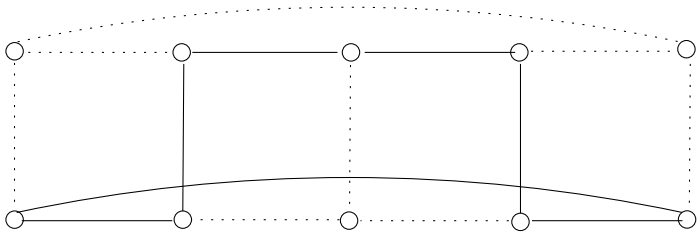


Figure 2.4: An extended long torus 7-cycle.

2.2.1 Practical Relevance of Small Chordless Cycles

We ask the question which chordless cycles are important in practice. To this end, we consider small grid sizes and determine the cycle polytope by only calling the exact cycle separation routine. We restrict the algorithm to generating chordless cycles only. For each sample, we let the branch–and–cut algorithm output the last linear program generated in the optimization process. (The instances don't have to branch. So with the analyzed linear programs optimality of a known solution can be proven.)

We consider 50 randomly generated two-dimensional grids of quadratic size $L \times L$ with $L = 11$, having $\pm J$ distribution and 50% negative couplings. Considering

a grid of odd length scale in all direction is advantageous because we can easily differentiate between long torus cycles and extensions of it and the small chordless cycles displayed in Figures 2.2 and 2.3.

$\|C\|$	occurrence in %
4	46.7 ± 0.4
8	3.5 ± 0.1
10	1.9 ± 0.1
11	2.2 ± 0.1
12	1.5 ± 0.1
13	8.8 ± 0.3
14	1.0 ± 0.1
15	8.0 ± 0.3
16	0.9 ± 0.3
17	5.6 ± 0.2
18	0.8 ± 0.09
19	3.4 ± 0.1
20	0.7 ± 0.08
21	2.3 ± 0.1
22	3.8 ± 0.1
23	1.5 ± 0.1
24	1.7 ± 0.1
25	0.7 ± 0.08
26	1.1 ± 0.1
27	0.56 ± 0.08
28	0.59 ± 0.08
29	0.39 ± 0.08
30	0.54 ± 0.08
31	0.24 ± 0.08
32	0.46 ± 0.08
33	0.63 ± 0.08
34	0.21 ± 0.08
35	0.28 ± 0.08
$36 \leq l \leq 58$	0.65 ± 0.08

Table 2.3: Occurrence of chordless cycles of different sizes $|C|$ in a two-dimensional grid of size $L = 11 \times 11$, $\pm J$ distribution and 50% negative interactions, optimized using the exact cycle separation.

On average, for an 11×11 grid the analyzed linear program consists of 187 ± 21 inequalities. In Table 2.3 we report the occurrence of cycles of different sizes $|C|$ in percent. The occuring cycles are short, cycles containing more than $2L$ edges are not relevant. The four-cycles occur most often ($46.728 \pm 0.4\%$). The second most often

occurring cycles are cycles of size 13, 15 and 17. From the last section we know that the support of these inequalities are extended long torus cycles.

We see the same qualitative behavior when studying grid size 15×15. The four-cycles occur most often ($46.949 \pm 0.5\%$). Also important are the inequalities of size $L+2$ and $L+4$ that are extended long torus cycle inequalities. Together they form $13.571 \pm 0.4\%$ of the occurring inequalities.

We now study 50 randomly chosen instances of three-dimensional grids with sizes $L = 3, 4, 5$ with $\pm J$ distribution. As the numbers for $L = 3, 4$ are analogous to the numbers for $L = 5$, we restrict ourselves to $L = 5$. For three-dimensional grids of size $L = 5$ a sample contains on average 310.58 ± 29 inequalities per lp. For a statistics of 50 instances with $\pm J$ distribution, the four-cycles on a plaquette occur most often ($49.314 \pm 0.4\%$). Long-torus cycles and extended long-torus cycles occur frequently ($19.57 \pm 0.2\%$ for cycles of size $5, 7, 9$). The generated violated cycles are short, no cycle contains more than 25 edges, i.e., all cycles contain less than $6.67 \pm 0.2\%$ of the grid edges.

In our tests each variable usually occurs at least once in an lp. Additionally, whenever negative edge weights induce a frustrated plaquette (i.e., an elementary square of four edges with an odd number of negative interactions) in the grid, the corresponding cycle inequalities occur in the cyclic description. This result is independent of the dimension, the grid size and the percentage of negative couplings. An intuitive explanation is as follows. Assume we are given a frustrated plaquette as in Figure 2.5 and we want to maximize $x_{12} + x_{23} + x_{13} - x_{34}$ under the constraint that the solution vector x is contained in some relaxation of the cut polytope. An optimal solution tries setting variables $x_{12}x_{23}, x_{13}$ to a value near one and x_{34} to a value near zero. Therefore, the corresponding probably violated cycle inequality $x_{12} + x_{13} + x_{24} - x_{34} \leq 2$ is added to the problem formulation. We conclude that negative weighted edges occur with a minus in the inequality. Even in non-frustrated cycles the negative weighted edges more likely occur with a minus in the corresponding cycle inequality than with a plus. In Table 2.2.1 we report the percentages of minus-edges and plus-edges in non frustrated and frustrated cycles, for 50×50 grids.

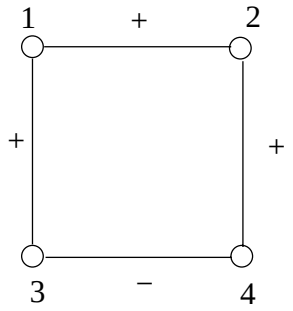


Figure 2.5: A frustrated plaquette.

	# of -1	# of $+1$
non frustrated cycles	89.97 ± 0.03	10.03 ± 0.004
frustrated cycles	99.57 ± 0.02	0.430 ± 0.001

Table 2.4: Occurrence of -1 and $+1$ as coefficients in the cycle inequalities, for non frustrated cycles and for frustrated cycles in $L = 50 \times 50$ grids.

From these numbers, we can deduce a strategy for heuristically generating violated cycle inequalities in the separation procedure. We should first test cycle inequalities for violation that are defined on chordless cycles of small length and then proceed to bigger cycles.

Algorithm: *generate_cycle_relaxation*

1 generate frustrated violated four-cycles defined on the plaquettes
2 generate non frustrated violated four-cycles
3 generate frustrated violated six-cycles defined on the elementary cubes
4 generate non frustrated violated six-cycles defined on the elementary cubes
5 generate violated long-torus and extended long-torus cycles
6 generate more general violated cycle inequalities
7 **if** *no inequality can be generated in steps 1 to 6* **then**
separate cycle inequalities exactly

Algorithm 4: *generate_cycle_relaxation* for fast generation of the cycle relaxation in regular grids.

Steps 1. to step 4. can be done by enumeration. In the subsequent section 2.2.2 we present a heuristic for the long-torus and extended long-torus cycles that have to be generated in step 5. Step 6 can be done by either calling the *forest* routine or the *random_cycles* routine explained in Section 2.2.3. The exact cycle separation is done as explained after Definition 1.6.

2.2.2 Generating Extended Long Torus Cycle Inequalities

For step 5 of the cycle separation in Algorithm 4 of the last section, we give a heuristic that generates 'promising' extended long torus cycle inequalities. We restrict ourselves to generating (extended) long torus cycle inequalities of size $L, L+2$ and $L+4$, as the numbers in the tables above report that extended long torus cycle inequalities of bigger sizes do not occur frequently. The separation routine we describe generates extended long torus cycles along a row in a grid with periodic boundaries. Taking the grid symmetry into account, we analogously generate extended long torus cycle inequalities column- and inter-layerwise.

Assume we are given $x^\star$ to be separated. For a row i, we first generate the long torus cycle C_L along row i. C_L has length L. Let the nodes on C_L be $j, j+1, j+$

$2, \ldots, j+L-1, j$. We determine the best (with regard to violation) cycle inequality defined on C_L by calling Procedure 2 *best_ineq_on_given_cycle*. We try improving its violation by extending C_L to an extended long torus cycle C_{L+2} of length $L+2$ as can be seen in Figure 2.6. In order to do this, we choose three neighboring nodes, say $k, k+1, k+2$ from row $i-1$. (Indices are always taken modulo the number of rows and the number of grid sites in a row, respectively). $k, k+1, k+2$, together with $|C_L|-1$ appropriately chosen nodes from C_L form a chordless extended long torus cycle of length $L+2$. The nodes $k, k+1, k+2$ are chosen such that the resulting cycle inequality has the highest violation. Let the extended long torus cycle C_{L+2} be of the form $j, j+1, \ldots, j+r, k, k+1, k+2, j+r+2, \ldots, j+L-1, j$. We now test whether we can improve the violation of the best inequality defined on C_{L+2} by extending the path $k, k+1, k+2$ to the left and to the right to a path $\ldots k-1, k, k+1, k+2, k+3, \ldots$. We keep on extending the nodes to the right (left) as long as less than $L-1$ nodes are affected in row $i-1$ and as long as doing this improves the violation of an inequality defined on C_{L+2}.

This procedure is indicated by the arrows in Figure 2.6. Assume we end up with an extended long torus cycle C_{L+2} that contains the nodes $k-t, \ldots, k, k+1, k+2, \ldots, k+s$ from row $i-1$. We try improving the violation of the best cycle inequality on C_{L+2} further by expanding C_{L+2} to a cycle of size $L+4$ by letting expand the cycle along three rows as indicated in Figure 2.7. We do this analogously to the generation of a cycle of size $L+2$. As we want to generate chordless cycles, we only have to consider replacing the nodes $k-t \ldots, k+s$ by the corresponding nodes in row $i-2$. We summarize the heurisic *expand_long_torus_cycles* below.

Algorithm: *expand_long_torus_cycles*
for *each row i in the grid* **do**
    generate long torus cycle C along row i
    generate best extended long torus cycle inequality of size $L+2$
    generate best extended long torus cycle inequality of size $L+4$
    if *best cycle inequality is violated* **then**
        add inequality to the problem formulation

Algorithm 5: *Expand_long_torus_cycles* generates extended long torus cycles in a regular grid.

In the following we give experimental results for three-dimensional grids of $\pm J$ or Gaussian distribution. We report the random seeds of the instances, the numbers of linear programs solved and the CPU time with (denoted by (1)) and without (denoted by (2)) calling the heuristic for generating extended long torus cycles. For (2), we generate the cycle polytope as in Algorithm 4. For (1) we do it analogously but skip the generation of the extended long torus cycles. We report numbers for 5^3 grids and $\pm J$ distributed instances and 6^3 grids for Gaussian distributed instances.

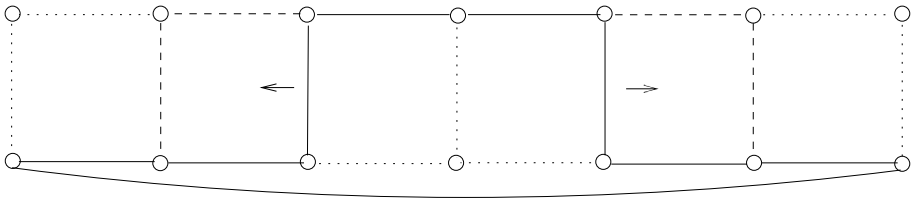


Figure 2.6: Generating an extended long torus cycle of size $L + 2$, having started from a long torus cycle.

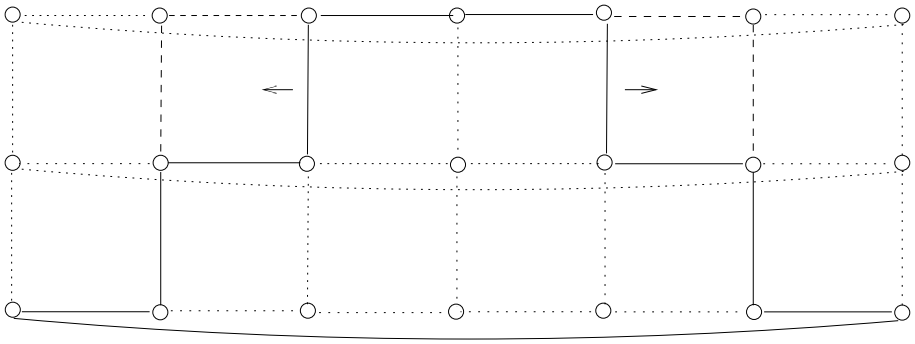


Figure 2.7: Generating an extended long torus cycle of size $L + 4$, having started from an extended long torus cycle of size $L + 2$.

The number of linear programs usually reduces if the *extended_long_torus_cycles* routine is called. The runnning time reduces slightly with the new heuristic.

2.2.3 Generating More General Violated Cycle Inequalities

In spin-glass computations it is not sufficient to generate small cycles. Being able to generate some long violated cycles seems to be important in order to improve the upper bound. We do this in step 6 of the cycle separation Algorithm 4. We generate longer violated cycles in the *forest heuristic* and the exact cycle separation only called on a random fraction of the nodes.

As the forest heuristic is already introduced elsewhere [10], we only briefly summarize it. Let $G = (V, E)$ be given and $x^\star$ the point to be separated. We determine a maximum weight spanning tree $T = (V_T, E_T)$ with $V_T = V, E_T \subseteq E$, where the edge weights are chosen as $|x^\star_e - \frac{1}{2}|$. By inserting a non-tree edge $e = (i, j)$ to T, exactly one cycle evolves. This *fundamental cycle* consists of the union of e and the unique path from i to j in the tree. We set $F = \{e \in C \mid x^\star_e > \frac{1}{2}\}$. In case $|F|$ is odd and the corresponding cycle inequality is violated by $x^\star$, we add it to the constraint buffer.

Another possibility of generating general violated cycle inequalities exist. As explained in Chapter 1.3, the exact cycle separation routine works as follows: We create two copies of the original graph G that contain for each node $i \in V$ the two

instance	lps(1)	lps(2)	cpu(1)	cpu(2)
1000	152	123	42.81	37.35
1001	55	53	19.01	16.50
1002	151	160	47.03	47.00
1003	42	57	13.73	18.02
1004	73	94	22.00	26.86
1005	34	33	10.91	11.36
1006	31	27	11.73	11.03
1007	207	220	62.96	66.76
1008	43	40	14.75	13.95
1009	125	122	35.33	33.90

Table 2.5: Performance of the algorithm without (denoted by (1)) and with (denoted by (2)) calling the heuristic *extended_long_torus_cycles* for grids of size $6 \times 6 \times 6$ and Gaussian distributed couplings.

instance	lps(1)	lps(2)	cpu(1)	cpu(2)
1000	87	65	13.77	10.88
1001	189	147	32.97	24.13
1002	43	117	10.04	21.29
1003	25	10	5.70	3.02
1004	19	20	4.40	4.73
1005	133	120	21.88	19.16
1006	17	16	4.07	4.13
1007	25	12	4.33	3.12
1008	15	13	3.72	3.42
1009	52	15	9.76	4.17

Table 2.6: Performance of the algorithm without (denoted by (1)) and with (denoted by (2)) calling heuristic *extended_long_torus_cycles* for grids of size $5 \times 5 \times 5$ and $\pm J$ couplings.

copies i' and i''. For each pair of nodes i' and i'' we solve a shortest path from i' to i'' in the doubled graph. The exact separation needs much CPU time in practice. Nevertheless, we can improve the cycle bound by choosing pairs i', i'' randomly. We only determine the shortest paths between the chosen pairs. We do this for a fraction $\frac{1}{k}$ of the nodes.

2.2.4 Experimental Results

Finally, we give experimental results for the performance of branch–and–cut when the cycle relaxation is either generated heuristically or exactly. The heuristic cycle bound well approximates the exact bound. For two-dimensional instances up to size 100×100 branch–and–cut with the heuristic cycle separation virtually never has to branch. This means that the cycle polytope is a very tight relaxation for the cut polytope in two dimensions. Furthermore, the heuristic cycle bound is a good approximation of the exact cycle bound. This is intuitive as we generate the violated small chordless cycles first and general violated cycles subsequently as suggested by the statistical tests we reported in Table 2.3. In Table 2.7 we show the CPU time for two-dimensional Ising spin-glass instances with Gaussian distribution. We compare the branch–and–cut algorithm with the exact cycle separation routine only (denoted as (1)) with the algorithm in which the heuristics are included. The runs are performed on a 440MHz Ultra Sparc. It is immediate that the running times are considerably faster with the heuristic than with the exact cycle bound. We have analogous characteristics for three-dimensional instances. (However, the cycle bound is not as good as in two-dimensions. This is intuitive: The cut polytope coincides with the cycle polytope for planar graphs. The more edges we insert that destroy planarity, the worse the cycle bound is.)

seed	CPU(1)	CPU(2)
1000	3944.37	24.85
1001	5419.40	71.37
1002	4464.46	43.98
1003	2493.35	22.87
1004	4541.33	27.10

Table 2.7: Performance of branch–and–cut when the exact cycle separation is used (denoted as (1)) or when the heuristic cycle separation is included (denoted as (2)) for two-dimensional instances of size 40×40 with Gaussian distribution.

As a conclusion of the previous sections we note that a good performance is archieved if the cycle relaxation is generated heuristically by Algorithm 4 and solved by the simplex algorithm.

2.3 Solving the Cycle Relaxation

In the section above we studied the question how the cycle relaxation can be generated. In this section we study the question how to solve it. In practice, the simplex algorithm is very fast. We let the primal simplex algorithm solve the very first linear program in the branch–and–cut algorithm. In subsequent iterations, we add constraints to the linear program and resolve it. After having added constraints to an lp the former optimal basis is not primal feasible any more, but remains dual feasible as the optimality conditions are still satisfied. Starting from this basis we can apply the dual simplex algorithm which is much faster than starting from scratch with the primal simplex algorithm.

Recently, Barahona et al. suggested to replace the traditional simplex solver inside a branch–and–cut framework by the so-called *volume algorithm* [9], [12], an approximate solver for linear programs. In [12] the running times of a simplex based branch–and–cut code are compared with a volume based branch–and–cut code for Ising spin glass and Steiner tree problems. The published running times for two-dimensional $\pm J$ spin-glass instances and two-dimensional Gaussian instances with a magnetic field are much better with the volume algorithm than with IBM's OSL dual simplex algorithm. These results motivated us to study the behavior of the volume algorithm in more detail. In the following we introduce the volume algorithm and show computational results. It turns out that the volume algorithm is rather slow when used as a standalone lp solver. Furthermore, replacing the simplex algorithm by the volume algorithm inside branch–and–cut does not seem to improve its performance in most cases.

The volume algorithm as introduced in [9] is a subgradient method for generating an approximate solution of the linear program (P)

$$
\begin{aligned}
& \min c^T x \\
(P) \qquad & Ax \geq b \\
& 0 \leq x \leq 1.
\end{aligned}
$$

No convergence proof is known for its original version. However, later it could be shown that a modified version converges [5].

Let the Lagrangean function for the dual multipliers π be

$$
\begin{aligned}
& L(\pi) = \min_x (c - \pi A)x + \pi b \\
& 0 \leq x \leq 1
\end{aligned}
\tag{2.1}
$$

The volume algorithm aims at maximizing $L(\pi)$. For all $\pi \geq 0$, $L(\pi)$ is a relaxation of the problem (P), thus gives a lower bound on the optimum solution of the latter.

The maximum $L(\pi^\star)$ of the Lagrangean is equal to the optimum solution value of (P), if for $\pi^\star$ and the corresponding $x^\star$, $\pi^\star(b - Ax^\star) = 0$ is satisfied.

Algorithm: *volume algorithm [9]*
$t = 1$
choose $\bar{\pi} \geq 0$, solve (2.1) with $\pi = \bar{\pi}$
let solution be $\bar{x}, \bar{z} = L(\bar{\pi})$, set $x_0 = \bar{x}$
while $t \leq t_{\max}$ **do**
  heuristically choose an appropriate stepsize s
  compute $v^t = b - A\bar{x}, \pi' = \bar{\pi} + sv^t$
  set $\pi_i^t := \max(0, \pi_i')$
  solve (2.1) with $\pi = \pi^t$; let the solution be $x^t, z^t = L(\pi^t)$
  update $\bar{x} = \alpha x^t + (1 - \alpha)\bar{x}$, with $0 \leq \alpha \leq 1$ chosen such that $A\bar{x} - b$ is minimal
  if $z^t > \bar{z}$ **then**
    $\pi \rightarrow \pi^t, \bar{z} \rightarrow z^t$
  $t = t + 1$

Algorithm 6: The volume algorithm.

The optimum solution values of the primal problem (P) correspond to the volumes below the active faces of the corresponding dual problem. Within the algorithm, $\bar{x}$ is chosen in order to heuristically approximate these volumes. This is where the name 'volume algorithm' comes from. For the details, see [9].

We first study the performance of the volume algorithm as a standalone approximate solver for linear programs. Then we compare the behavior of branch–and–cut for max-cut both with the simplex and with the volume algorithm. We want to point out that the authors themselves have not claimed that the volume algorithm performs well as a standalone lp solver. However, studying its power standalone helps understanding its characteristic behavior.

For the tests, we use IBM's programs. We use the programs from the *COIN-OR* open source project [27]. Among others, COIN-OR contains the volume algorithm and `Bcp`, a branch–and–cut framework that has an interface both to OSL and the volume algorithm. With COIN comes also a program `bcps2` solving max-cut instances using `Bcp` with OSL or `Bcp` with volume. Special functions are included for solving two-dimensional Ising spin-glass instances.

We compare the volume algorithm with an interior point solver that is run without performing a crossover to a basic solution. Comparable to the volume algorithm, such an algorithm determines an approximate solution 'near' an optimal face. We stop the volume algorithm (as default in `bcps2`) if either the default convergence tolerance is reached or if a maximum of 2000 steps is performed. As interior point solver, we choose CPLEX's barrier algorithm (version 7.1) `baropt`. We test convergence

tolerances of $10^{-6}\%$ (as default for `baropt`) and 2% (default for volume) for both the volume and the `baropt` algorithm. The runs were performed on a 1400 Mhz AMD Athlon.

We optimize a linear function over the cycle polytope $P_{\text{cyc}}(G)$ of a graph G, i.e., we solve

$$(MC_{\text{cyc}}) \qquad \begin{array}{l} \max c^T x \\ x \in P_{\text{cyc}}(G) \end{array}$$

In Table 2.8 we show running times in seconds for instances of (MC_{cyc}) with the underlying graph G being complete. The edge weights are chosen from either a Gaussian distribution or from $\{\pm J\}$, with 50% negative weights. The format of the name of an instance is `c<nnodes>_<random seed>`.

instance	`baropt` 2%	vol 2%	`baropt` $10^{-6}\%$	vol $10^{-6}\%$
`c30_555`	0.52	3	0.63	3
`c50_555`	5.08	22	6.00	99
`c70_555`	21.12	98	29.04	$675^\star$ (0.4%)
`c90_555`	74.39	213	109.72	$5636^\star$ (0.04%)

Table 2.8: Running time in seconds for different lp solvers when optimizing over the cycle polytope of a complete graph.

Instances marked with a * stopped with having performed the maximum number of iterations. In parenthesis we then give the quality of the approximation in percent. For a convergence guarantee of 2%, the interior point solver `baropt` is roughly a factor of four faster than the volume algorithm. From the last column in Table 2.8 we see that for graphs having more than 60 nodes, the volume algorithm cannot approximate the optimum solution of (MC_{cyc}) better than within $\sim 10^{-2}\%$, and the algorithm stops after considerable running time having performed the maximum number of steps without having reached the required guarantee. In contrast, `baropt` solves the linear programs fast within a guarantee of $10^{-6}\%$. The results for Gaussian distributed instances are comparable to the numbers in Table 2.8.

Next, we study the performance for linear programs appearing in spin glass ground state computations. We study instances defined on two-dimensional grids with periodic boundary conditions. Our code `mc` is based on the ABACUS [2] branch–and–cut framework. We let `mc` for max-cut output the 10th linear program it generates. (The choice of the 10th lp is arbitrary.) We solve the lp from scratch with the volume algorithm, `baropt`, CPLEX's primal simplex solver `primopt` and its dual simplex solver `tranopt`.

The linear programs have around 2700 rows. In Figure 2.8, we show on a logarithmic scale the running time as a function of the grid size for the different solvers. We show results for $\pm J$ distributed instances. (As before, the results for Gaussian instances are comparable and skipped.) Compared to the solvers included in CPLEX, we find long running times for the volume algorithm which are roughly a factor of 20 slower than with CPLEX's solvers. The volume algorithm typically approximates the optimal solution within 2%; usually not better. This quality is not very powerful compared to the default convergence tolerance of 10^{-6}% in `baropt`.

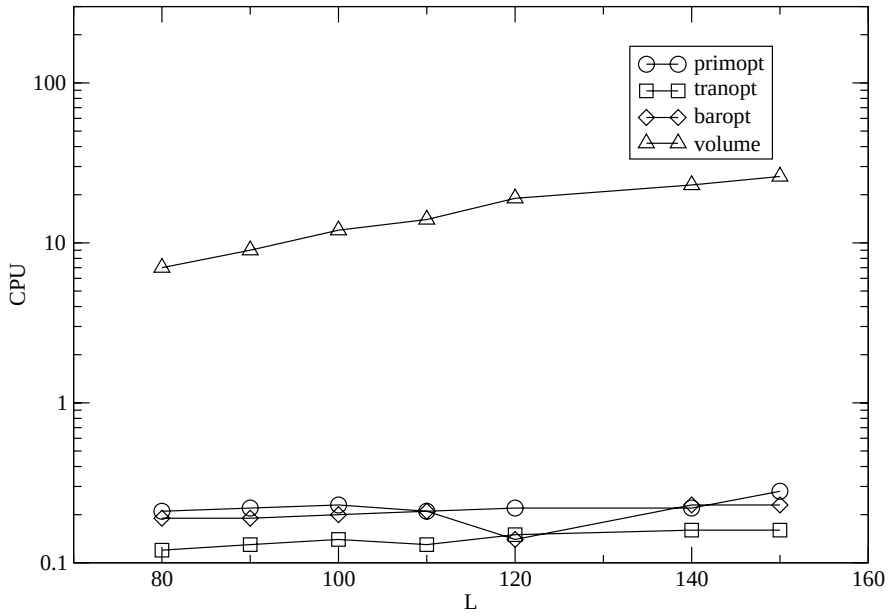


Figure 2.8: Running times of different lp solvers for linear programs arising in spin-glass ground-state computations.

As a conclusion, we notice that it does not pay off to use the volume algorithm as a standalone solver for solving linear programs. Both the running times are much higher and the approximation guarantee is worse than with CPLEX's lp solvers.

In the following we study the performance of different lp solvers within branch–and–cut. The max-cut solver contained in the COIN-OR project is based on the branch–and–cut library `Bcp` that optimizes over the cycle polytope. It can be compiled to either use OSL or to mainly use the volume algorithm for solving the linear programs. In the latter case, the arising lps are solved by the volume algorithm. However, the lp is solved exactly by the dual simplex algorithm before branching takes place. So a correct upper bound on the max-cut value is stored and used for

possible fathoming of sub problems. IBM's max-cut code contains some heuristics for solving spin-glasses. The four-cycles are enumerated and the forest heuristic is included, see Section 2.2. There are some additional primal randomized heuristics flipping up to four spins simultaneously.

For two-dimensional Gaussian Ising spin-glass instances, the running times for `bcps2` with volume and `bcps2` with OSL are roughly comparable. However, for $\pm J$ distributed instances the running times are much better for `bcps2` with volume than for `bcps2` with OSL's simplex algorithm, see Table 2.9. Runs were performed on a 1400 MHz AMD Athlon. Because of the long running time, we only did small sizes for `bcps2` with OSL.

instance	`bcps2` & volume	`bcps2` & OSL
`t30pm_555`	3.16	> 1800.69
`t30pm_666`	4.11	284.73
`t40pm_555`	7.50	> 4897.88
`t40pm_666`	24.50	not done
`t50pm_555`	75.51	not done
`t50pm_666`	65.39	not done

Table 2.9: Comparison of the performance of `bcps2` with OSL and with `bcps2` with volume for two-dimensional $\pm J$ distributed spin-glass instances.

The different behavior between Gaussian distributed and $\pm J$ instances is surprising. For understanding it, we first notice that the volume algorithm probably does not generate an (optimal) vertex as the simplex algorithm does, but a solution 'near' a (maybe optimal) face of the current polytope. Intuitively it is clear that in the subsequent round of separation the generated cutting planes will cut 'deeper' than when just a vertex is cut off. (Therefore, both for Gaussian and $\pm J$ distributed instances we need fewer rounds of separation if the volume instead of the simplex algorithm is used inside branch–and–cut. However, for the Gaussian case, the smaller number of iterations does not result in a better overall performance.)

An important reason for the better overall performance of the volume algorithm for $\pm J$ instances is due to the better *primal solutions*, i.e., the best known cuts, that can be generated. We explain this in the following. We will show in Section 2.6 that good cuts can be generated by appropriate rounding the fractional solutions generated by the lp solvers. As will also be explained in Section 2.6, for $2d \pm J$ instances the knowledge of a good cut helps reducing the running time, whereas for Gaussian distributed instances the knowledge of a good (or optimum) cut does not have a considerable effect on the performance.

Usually, the development of the primal solutions inside the branch–and–cut framework with the volume algorithm is much better than their development inside branch–and–cut with the simplex algorithm. For backing up this claim, we define the

gap g as the difference of the optimal cut z_{MC} and the best known cut z_0 generated in the first round of iteration, divided by z_{MC}:

$$g = \frac{z_{MC} - z_0}{z_{MC}}. \tag{2.2}$$

In Table 2.10 we show numbers for the gap in percent for two-dimensional $\pm J$ instances. We denote the gap as g_V for z_0 generated from the volume solution and as g_{OSL} for z_0 generated from the simplex algorithm.

instance	$g_V\%$	$g_{\text{OSL}}\%$
t30pm_555	6.00	34.07
t40pm_555	2.90	36.25
t50pm_555	0.81	31.26
t60pm_555	2.94	44.08

Table 2.10: The gap in percent defined as in (2.2) for some randomly chosen two-dimensional $\pm J$ instances.

We find that the cut generated with branch–and–cut and volume is within 6% of the optimum, whereas the cut generated with branch–and–cut and simplex is more than 30% away from the optimal cut value. We find comparable numbers for the Gaussian case. Therefore, having a better primal solution at hand early in the optimization process is a main reason for the good performance of the volume algorithm inside branch–and–cut for two-dimensional $\pm J$ spin-glass instances.

We find the same phenomenon (fewer rounds of separation, better cuts) if we use CPLEX's barrier solver in our b&c framework ABACUS. However, `mc` needs longer running time when the dual simplex method is replaced by the barrier solver and so we use the dual simplex algorithm.

It is also interesting to compare the performance of `mc` and `bcps2` for two-dimensional Ising spin-glass instances. For each system size, we run two different samples on a 1400 Mhz AMD Athlon. For two-dimensional $\pm J$ instances, the running times are roughly comparable, as can be seen in Figure 2.9. For Gaussian distributed instances, `mc` performs better than `bcps2`, as shown in Figure 2.10. In Table 2.11 we show the running times in seconds for the eight $\pm J$ instances that were used in [12]. For these instances `bcps2` performs better than `mc`. However, for the two randomly chosen instances of size 80×80 from Figure 2.9 above, the running times for `bcps2` and `mc` are comparable. We conclude that we have roughly comparable running times for `bcp2` and `mc` for two-dimensional spin-glass instances.

`bcps2` does not contain special separation routines for three-dimensional grids, but we can treat a three-dimensional instance as a general sparse graph. For eight tested instances of size 3^3 to 6^3 `bcps2` with simplex is even a factor of two faster than `bcps2` with volume. We show the running times for randomly chosen instances for `bcps2`

instance	bcps2 & volume	mc
L_70_1	226	242
L_70_2	45	109
L_70_3	186	332
L_70_4	545	1858
L_70_5	496	1216
L_70_6	319	4397
L_70_7	148	321
L_70_8	255	1052

Table 2.11: Comparison of running times for bcps2 with the volume algorithm and mc for the eight instances mentioned in [12].

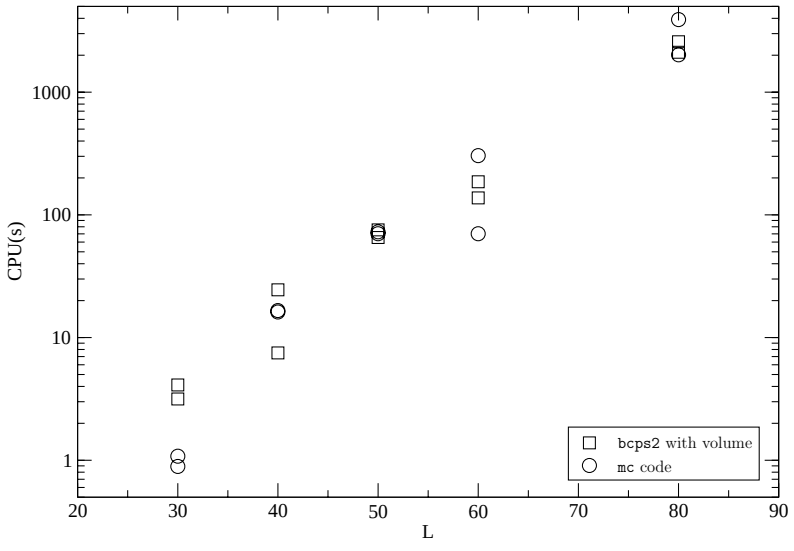


Figure 2.9: Comparison of the running times for two-dimensional $\pm J$-distributed weights of size $L \times L$ for our mc-code and the bcps2-code with the volume algorithm.

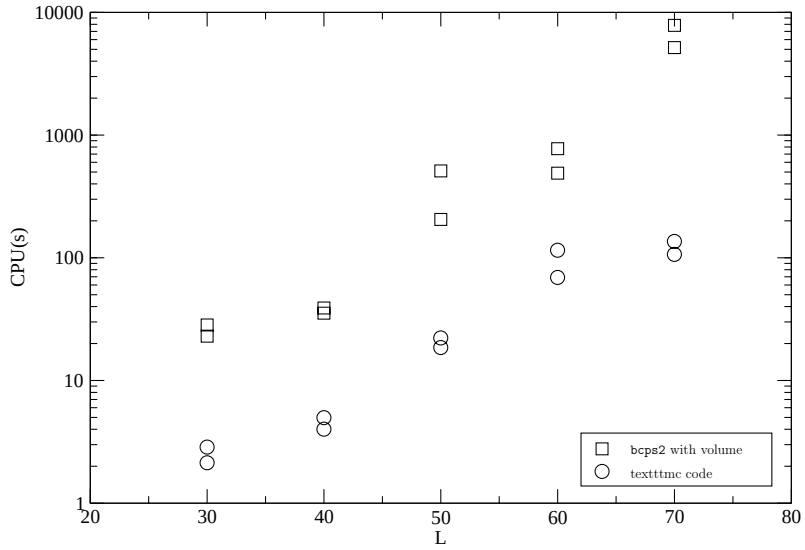


Figure 2.10: Comparison of the running times for two-dimensional Gaussian distributed weights of size $L \times L$ for our `mc`-code and the `bcps2` code with the volume algorithm.

and `mc` in three dimensions in Table 2.12. The naming convention is `t_<linear dimension><g/pm>_<random seed>`. The `mc` code is faster than the `bcps2` code (both with simplex and volume). We notice that the `mc` code contains special separation routines for three-dimensional spin-glasses, so we should not take this comparison too seriously.

instance	`bcps2` & volume	`mc`
`t5g_555`	21.26	0.63
`t5g_666`	13.51	0.40
`t6g_555`	138.35	2.70
`t6g_666`	980.54	65.71
`t5pm_555`	101.41	3.75
`t5pm_666`	52.39	1.23
`t6pm_555`	320.15	15.68
`t6pm_666`	2066.29	79.72

Table 2.12: Comparison of running times for `bcps2` and `mc` for three-dimensional Gaussian and $\pm J$-distributed instances.

As a conclusion, we notice that the volume algorithm inside branch–and–cut only performs well in special cases, e.g., the two-dimensional $\pm J$ spin glass. The performance of the volume algorithm inside a branch–and–cut framework is problem specific and it is not clear beforehand how it performs for other combinatorial optimization problems or even for different distributions of the weights.

2.4 The 4-Neighbour Graph Facet

In [118] the authors presented the circulant inequality (1.22) and mentioned that it would be interesting to know facet defining inequalities on the 'true' circulants that we call *k-neighbour graph*. Let the edge set E^k of a k-neighbour graph $H^k = (V^k, E^k)$ for all nodes i consist of $(i, i+1), \ldots, (i, i+k)$. Indices are always taken modulo $|V^k|$. In this section we introduce a new facet of the cut polytope that is defined on 4-neighbour graphs $H^4 = (V^4, E^4)$.

Lemma 2.3. *Let $r \geq 4$ be an even number and $n = 3r + 1$ or $n = 3r - 1$. Let $H^4 = (V^4, E^4)$ with $|V^4| = n$ be a 4-neighbour subgraph of a graph $G = (V, E)$. Then the inequality*

$$\sum_{e \in E^4} x_e \leq 3n - r - 1 \tag{2.3}$$

is valid and facet defining for $P_C(G)$.

Proof. We first show validity of (2.3). To this end, we show that (2.3) is satisfied for a cut of maximum cardinality in a 4-neighbour subgraph. Let us first consider an infinitely long 4-neighbour chain in which for each node i the edges $(i, i+1), \ldots, (i, i+4)$ exist, see Figure 2.11.

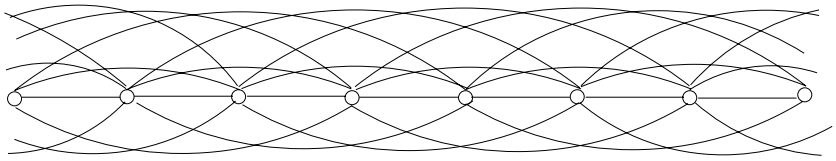


Figure 2.11: A part of an infinitely long 4-neighbour chain.

We determine a cut of maximum cardinality on finite substrings of length n, however $n \to \infty$. We first study 'regular' solutions, i.e., solutions in which l nodes are in W, the following l nodes are not in W, the subsequent are in W, etc. A solution with $l > 5$ can easily be improved, and so we consider $l \leq 5$. Solutions with $l = 1, 2$ do not cut $\sim 2n$ edges. For $l = 3$, $\sim \frac{4}{3}n$ edges are not in the cut, and for $l = 4$, $\sim \frac{3}{2}n$ edges are not in the cut. For $l = 5$, $\sim 2n$ edges are not cut. Therefore, an optimum regular solution (with regular as defined above) is obtained for $l = 3$. In Figure 2.12 we show such a solution on a part of the chain. White nodes are in W, black nodes are in its complement. We only show the edges that are not in the cut $\delta(W)$.

What about solutions that do not have this regular structure? We show in the following that they cannot be better than the regular solution with node packages of length $l = 3$. Suppose we have a substring with an assignment of colors to the nodes that is not regular, together with a copy of it on on the string on which the solution is regular with $l = 3$. We compare the solutions on the two strings.

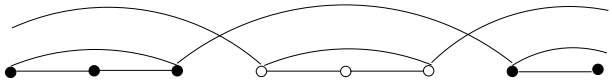


Figure 2.12: Optimum solution on the 4-neighbour chain. White nodes are in W, black nodes are in its complement. Displayed are only the edges that are not in the cut $\delta(W)$.

In order to do this, we assume we have a node package of length $p, p \leq 5$ within the not regular solution. We consider these p nodes plus additionally at least the four nearest neighbours at each end of the node package. (Nodes that are further away are not connected to a node in the package.) As we only have to study $p \leq 5$, we consider 13 nodes. Without loss of generality let the color of the p nodes be white. As the node package has exactly p nodes, the color of the nearest neighbours at both ends of the package is black. The color of the second, third, fourth nearest neighbours at both ends is not yet defined. We show this situation for $p = 5$ in Figure 2.13.

Figure 2.13: Considering a node package of $p = 5$ nodes together with four nodes at each end of the package. The color of the nodes with a question mark is not yet assigned.

It is enough to show for all values of p that no matter how we assign colors to the yet uncolored nodes, the resulting cut on the 13 node string does not cut more edges than the regular $l = 3$ solution.

For the the regular $l = 3$ solution on the 13 nodes, three qualitatively different possibilities for assigning colors exist. We show them in Figure 2.14.

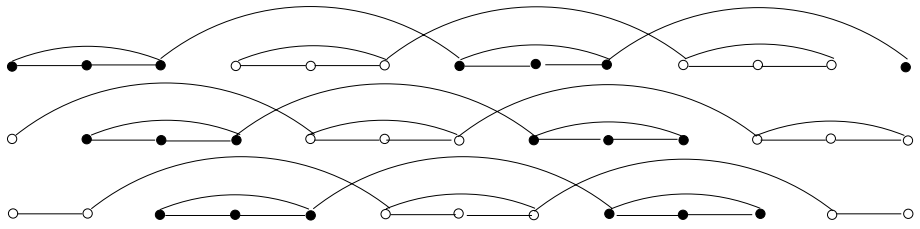


Figure 2.14: Three qualitatively different regular $l = 3$ solutions exist on a 13 node substring. Edges are only shown if they are not cut.

Two of the solutions in Figure 2.14 cut 27 edges, the other cuts 28 of them. Therefore, we show for the not regular solution that there exists no different assignment of colors to the yet uncolored nodes around the node package of p nodes such that more than 27 edges are in the cut. In fact, we checked this is true for all values $p \leq 5$ by brute

force enumeration of all possible assignments of colors to the uncolored nodes. We conclude that the regular $l = 3$ solution is optimum on the chain.

We now consider a 4-neighbour graph of $n = 3r$ nodes with $r \geq 6$ an even number. By analogous arguments as above we conclude that an optimum solution is obtained by the regular $l = 3$ solution.

For $n = 3r + 1$ ($n = 3r - 1$ respectively) it is not possible to always assign every three nodes a different color. However, by applying similar arguments as the ones above we obtain a solution with maximum cardinality by the 'regular' solution on $r - 1$ packages of three nodes each and one package consisting of four (two) nodes instead of three.

The corresponding cuts of maximum cardinality satisfy inequality (2.3) with equality and we have shown validity of (2.3). (Our considerations are for $n \geq 16$ nodes. However, $n = 11, 13$ can be treated separately.)

For inequality (2.3) being facet defining, we consider the case that H^4 is a 4-neighbour subgraph with $n = 3r - 1$ nodes. The case $n = 3r + 1$ is analogous. There exist n different cuts $\delta(W)$ of maximum cardinality. Each of them consists of a series of three nodes that are in W, three nodes that are not in W, etc., and one node package consists of only two nodes. We show that these incidence vectors $\chi^1, \ldots \chi^n$ are affinely independent. So we have to show that the equality system

$$\sum_{i=1}^{n} \alpha_i = 0 \tag{2.4}$$

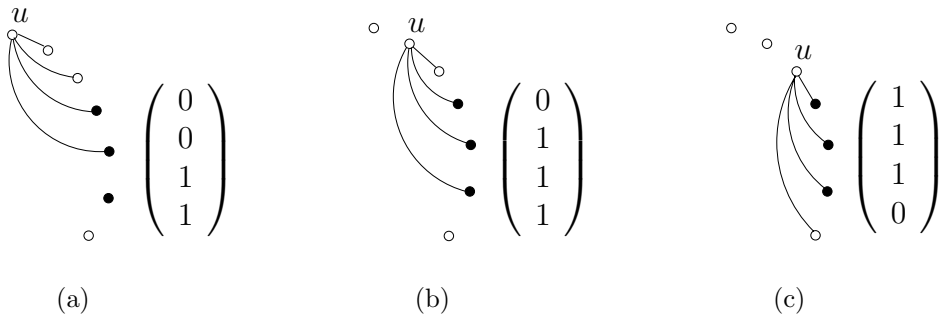
$$\sum_{i=1}^{n} \alpha_i \chi^i = 0 \tag{2.5}$$

has the unique solution $\alpha_1, \ldots, \alpha_n = 0$.

How do $\chi^1, \ldots \chi^n$ look like? Let us consider a node u that is connected to the subsequent four nodes in H^4. Five qualitatively different possibilities for u exist that we display in Figures 2.15 and 2.16. Either u is the first or second or third node of the three nodes in a package, and the following package also consists of three nodes. The other two possibilities occur when u is the second or the third node in a package and the subsequent package is the one package of two nodes. Next to the figures, we show the corresponding 4-tuples ordered as ${}^{\mathrm{T}}(\chi_{u,u+1}, \chi_{u,u+2}, \chi_{u,u+3}, \chi_{u,u+4})$ that appear in the incidence vectors.

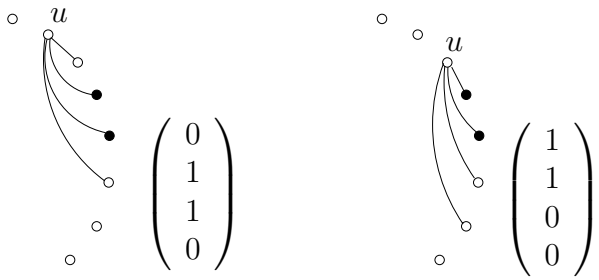
An incidence vector χ_i consists of the 4-tuples from Figure 2.15

$$\begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} \tag{2.6}$$



(a) (b) (c)

Figure 2.15: Three of the five building blocks of the incidence vectors χ_i.



(a) (b)

Figure 2.16: Two of the five building blocks of the incidence vectors χ_i.

repeatedly appearing one after the other, with the number of repetitions depending on the size of n. Once in each χ_i the tupels

$$\begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \tag{2.7}$$

occur, see Figure 2.16(b). We observe that each 4-tupel has 1 as the third entry except ${}^{\mathrm{T}}(1,1,0,0)$. The latter is of special interest. For one incidence vector it appears as the first 4-tupel that builds the 0/1 vector. For another vector it appears as the second, the third, etc., the n-th tupel.

We consider every third row of the coefficient matrix of the equality system (2.5). Exactly one zero appears in each of these rows, and all other entries are one. The zero appears exactly once as the first, second, third, etc., n-th entry. Thus, by extracting every third row in (2.4) and possibly reordering the extracted rows, we get an equality system

$$\begin{pmatrix} 0 & 1 & \dots & 1 \\ 1 & 0 & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \dots & 0 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{2.8}$$

The equality system (2.8) together with the equality $\sum_{i=1}^{n} \alpha_i = 0$ has the unique solution $\alpha_1, \dots, \alpha_n = 0$. Thus, the incidence vectors $\chi_1, \dots, \chi_n$ are affinely independent which completes the proof. □

We notice that for a 4-neighbour subgraph with $3r$, $r \geq 4$ even, nodes the analogous inequality $\sum_{e \in E^4} x_e \leq 3n - r$ is valid but not facet defining for $P_C(G)$. Furthermore, for a 3-neighbour subgraph $H^3 = (V^3, E^3)$ the corresponding inequality would be $\sum_{e \in E^3} x_e \leq 2n$. This inequality is also valid but is the sum of n appropriately chosen cycle inequalities and therefore not facet defining. We notice that it is possible to extend the above idea to k-neighbour graphs with $k = 5$ and that we can devise an analoguous facet for the 5-neighbor graph. Separating the 4-/5-neighbor inequalities is NP-hard. For H^6, the regular $l = 5$-solution is not the optimum with respect to cardinality. The smallest counterexample consists of a 5-neighbor graph on 51 nodes. The regular $l = 5$ solution cuts 186 edges, whereas the maximum cardinality cut cuts 192 edges.

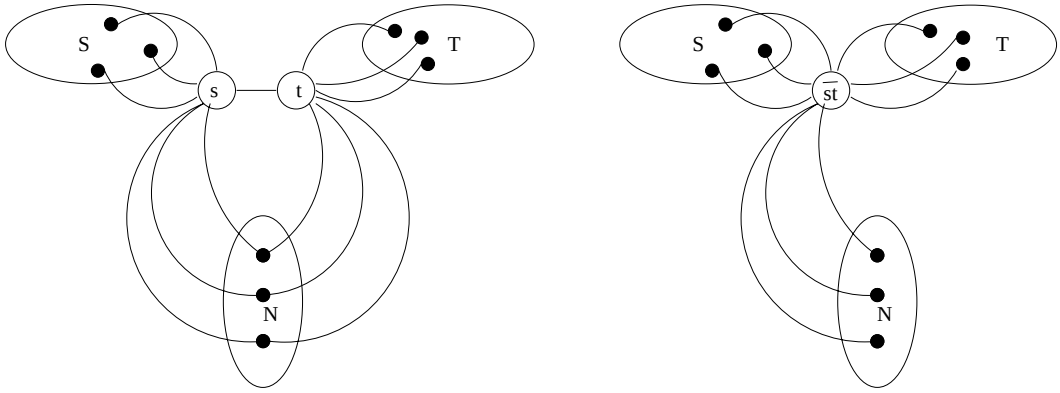
In Chapter 3 we study spin-glasses in the one-dimensional Ising chain model. For this model, k-neighbour subgraphs are important structures for the algorithm and we will use a heuristic separation of the 4-neighbour facets there.

2.5 Separating Inequalities Outside the Template Paradigm

2.5.1 Projecting and Lifting

The cut polytope is well-studied for dense or complete graphs. For sparse graphs, taking the cycle polytope as relaxation of the cut polytope gives already good results. The cycle polytope can be generated fast. It is an interesting question how to strengthen the cycle relaxation by adding further inequalities beyond the cycle inequalities.

Jünger, Reinelt and Rinaldi [55] give a lift–and–project procedure as follows. For an edge (s,t) in a graph $G = (V,E)$ we denote by S the set of nodes that are neighbors of s but not of t, by T the nodes that are neighbors of t but not of s and by N the set of nodes that are neighbors of both s and t, see Figure 2.17(a) for an example. By shrinking an edge (s,t) in a graph $G = (V,E)$ we end up with a graph $\overline{G} = (\overline{V}, \overline{E})$. We do this as follows. We combine the nodes $s \in V$ and $t \in V$ to a supernode $\overline{st} \in \overline{V}$. An edge $(u,v) \in E$ with $u \in \{s,t\}$ and $v \in S$ or $v \in T$ becomes $(\overline{st}, v) \in \overline{E}$, multiple edges with one endpoint in N are deleted as shown in Figure 2.17(b). Nodes and edges unaffected by this procedure remain unchanged in $\overline{G}$. A point $x^\star \in \mathbb{R}^{|E|}$ to be separated is projected to $\bar{x} \in \mathbb{R}^{|\overline{E}|}$.



(a) Original graph. (b) Shrunken graph.

Figure 2.17: Shrinking an edge (s,t). S and T are the sets of nodes that are only neighbors of s or t, respectively. N is the set of common neighbors.

Assume we have generated a valid inequality $ax \leq a_0$ for the cut polytope $P_C(\overline{G})$ of the shrunken graph that is violated by $\overline{x} \in \mathbb{R}^{|\overline{E}|}$. We *lift* $ax \leq a_0$ to an inequality $a'x' \leq a_0$ valid for the cut polytope of the original graph G by applying the *lifting procedure* introduced in the following definition.

Definition 2.4 (Lifting and unshrinking procedure). *Wlog. we denote the sets S, T such that*

$$\sum_{v \in T} |a_{vt}| \leq \sum_{v \in S} |a_{sv}|.$$

We lift *an inequality $ax \leq a_0$ defined on the shrunken graph $\overline{G} = (\overline{V}, \overline{E})$ to an inequality $a'x' \leq a_0$ defined on the lifted graph $G = (V, E)$ by setting*

- $a'_{sv} = a_{v\overline{st}} \; \forall \; v \in S$
- $a'_{vt} = a_{\overline{st}v} \; \forall \; v \in T$
- $a'_{st} = -\sum_{v \in T} |a_{v\overline{st}}|$
- $a'_{sn} = a_{\overline{st}n} \; \forall \; n \in N$
- $a'_{nt} = 0 \; \forall \; n \in N$

The next theorem states that by applying the lifting procedure 2.4 validity is maintained.

Theorem 2.5 (Jünger, Reinelt and Rinaldi [55]). *Let $ax \leq a_0$ be valid for $P_C(\overline{G})$. Then the lifted inequality $a'x' \leq a_0$ is valid for $P_C(G)$.*

Proof. Let us suppose there exists $W \subseteq V$ with $a'(\delta(W)) > a_0$. i.e., the inequality is not valid. Then edge (s, t) has to be in the cut $\delta(W)$. Let wlog $s \in W, t \notin W$ and set $W_t = W \cup \{t\}$. It is

$$a'(\delta(W_t)) = a(\delta(W_t)) \leq a_0,$$

as $ax \leq a_0$ is valid for $P_C(\overline{G})$. On the other hand, we have $a'(\delta(W_t)) = a'(\delta(W \cup \{t\}))$, which is

$$\begin{aligned} a'(\delta(W \cup \{t\})) &= a'(\delta(W)) - a'_{st} - \sum_{v \in T \cap W} a'_{vt} + \sum_{v \in T \cap V \setminus W} a'_{vt} \\ &= a'(\delta(W)) + \sum_{v \in T} |a'_{vt}| - \sum_{v \in T \cap W} a'_{vt} + \sum_{v \in T \cap V \setminus W} a'_{vt} \\ &\geq a'(\delta(W')) > a_0 \end{aligned}$$

and we have a contradiction. Thus, the lifting procedure from Definition 2.4 preserves validity. □

Suppose we generate an inequality valid for $P_C(\overline{G})$ that is violated by $\bar{x} \in \mathbb{R}^{|\overline{E}|}$. We want to make sure that the lifted inequality is also violated by the corresponding point $x^\star \in \mathbb{R}^{|E|}$. As formulated in the next Observation 2.6, this is achieved if we only shrink edges (s, t) with $x^\star_{st} = 0$ or $x^\star_{st} = 1$.

Observation 2.6. *Let a graph $G = (V, E)$ and a point $x^\star \in \mathbb{R}^{|E|}$ be given. Let $\overline{G}$ be obtained by shrinking an edge $(s, t) \in E$ with the property $x^\star_{st} = 0$ or $x^\star_{st} = 1$. Project $x^\star \in \mathbb{R}^{|E|}$ to $\overline{x} \in \mathbb{R}^{\overline{E}}$. Let $\overline{x}$ violate an inequality $ax \leq a_0$ by $v > 0$, i.e., $a\bar{x} = a_0 + v$. Then $x^\star$ violates the lifted inequality $a'x \leq a_0$ also by v.*

Proof. If $x^\star_{st} = 1$ we switch the vector $x^\star$ as

$$x^\star_i = \begin{cases} x^\star_i & e \notin \delta(s) \\ 1 - x^\star_e & e \in \delta(s) \end{cases}$$

and only have to consider the case $x^\star_{st} = 0$. It is $a'x^\star = ax^\star + a'_{st}x^\star = ax^\star = a_0 + v$. □

Under certain circumstances, the lifted inequality is facet defining.

Theorem 2.7 (Jünger, Reinelt and Rinaldi [55]). *Let $ax \leq a_0$ be valid for $P_C(\overline{G})$. Let the inequality $a'x \leq a_0$ be obtained through the lifting procedure defined in Definition 2.4. $a'x \leq a_0$ is facet defining if the following conditions are satisfied:*

(i) $ax \leq a_0$ is facet defining for $P_C(\overline{G})$ and

(ii) there exists a node set $W \subset \overline{V}$ with $\overline{st} \in W$ and $a(\delta(W)) = a_0$. Furthermore, it is

(a) $a_{\overline{st}v} \geq 0 \; \forall \; v \in T \cap W$
(b) $a_{\overline{st}v} \leq 0 \; \forall \; v \in T \cap \overline{V} \setminus W$
(c) $\sum_{w \in W} a_{vw} = \sum_{w \in \overline{V} \setminus W} a_{vw} \; \forall \; v \in N$.

Proof. Let $ax \leq a_0$ be a facet for $P_C(\overline{G})$ of dimension $|\overline{E}| - 1$. Then there exist $|\overline{E}|$ affinely independent incidence vectors of cuts satisfying $ax \leq a_0$ with equality. Let the incidence vectors be $\chi^{\delta(W_i)}$ with $W_i \subseteq \overline{V}$ and wlog $\overline{st} \in W_i$. By setting $W'_i = W_i \setminus \{\overline{st}\} \cup \{s\}$ the $|\overline{E}|$ incidence vectors $\chi^{\delta(W'_i)}$ are affinely independent. Furthermore, it is

$$\begin{aligned} a'(\delta(W')) &= a(\delta(W)) + a'_{st} + \sum_{v \in T \cap W} a_{\overline{st}v} + \sum_{v \in N \cap W} a_{\overline{st}v} - \sum_{v \in T \cap \overline{V} \setminus W} a_{\overline{st}v} \\ &= a(\delta(W)) + a'_{st} + \sum_{v \in T} |a_{\overline{st}v}| \\ &= a_0 \end{aligned}$$

and by zero-lifting the inequality on the edges $(t, n), n \in N$, $a'x \leq a_0$ is a facet. □

As an application of the above Lemma 2.7, we show the yet missing part of the proof 1.5 that cycle inequalities are facet defininig.

Cycle Inequalities are Facets. Let C be a cycle consisting of n edges and let $x(F) - x(C \setminus F) \leq |F| - 1$ be an arbitrary cycle inequality defined on C. Let C consist of the edges $c_1, c_2, \ldots, c_n$. We shrink $c_1, \ldots c_{n-3}$ by applying the shrinking procedure as in Figure 2.17. After $n-3$ shrinking steps the cycle is a triangle $\tilde{C} = \tilde{c}_1, \tilde{c}_2, \tilde{c}_3$. Consider a triangle facet (1.12) on $\tilde{C}$. We unshrink the graph again by undoing the shrinking steps in reverse order. Simultaneously we lift the triangle inequality by applying procedure 2.5. The resulting inequality on C in the unshrunken graph is again a cycle inequality. We show that a lifting step preserves the facet property. To this end we have to show the existence of the node set W with the property (iia), (iib) and (iic). For each node set W (iic) is satisfied as for all nodes $n \in N$ it is $a_{nu} = 0$ for $u \in \overline{V}$. We observe that exactly one coefficient $a_{\overline{stv}}$ for $v \in T$ has value 1 or -1; all other coefficients are zero. In both cases it is not hard to generate a node set W that satisfies (2.9) and (2.10) such that the cycle inequality is tight at $\delta(W)$. Therefore, the cycle inequality that is obtained by lifting a triangle inequality is a facet in the original graph. We can obtain all cycle inequalities on a cycle by switching it along an appropriately chosen cut. Switching preserves the dimension of a face. We conclude that cycle inequalities are facets. □

We can make use of the project–and–lift procedure within our separation procedure. We start with a weighted graph $G = (V, E)$ and $x^\star \in \mathbb{R}^{|E|}$ to be separated. As long as there is an edge $e = (u, v)$ in G with $x_e = 0$ or $x_e = 1$, we switch x along the cut $\delta(v)$ in case $x_e = 1$ and shrink edge e as explained above. After having applied all possible shrinking steps, we end up with a graph $\overline{G} = (\overline{V}, \overline{E})$ and a point $\overline{x} \in \mathbb{R}^{|E|}$. We generate inequalities valid for $P_C(\overline{G})$ that are violated by $\overline{x}$. By simultaneously unshrinking the graph and lifting the generated inequalities we reconstruct the original graph again and add the generated inequalities to our problem description.

Suppose we have applied the shrinking procedure and the resulting graph $\overline{G}$ is not complete, i.e., there exist nodes $i, j \in \overline{V}$ such that $(i, j) \notin \overline{E}$. We want to add the artificial edge $e = (i, j)$ with weight zero to the edges in $\overline{G}$ and do the separation procedure for the completed graph $\overline{G}^e$. Then we want to remove the artificial edges again and lift the violated inequalities. Several questions arise:

1. How can we extend $\overline{x} \in \mathbb{R}^{|\overline{E}|}$ to a point $\overline{x}^e \in \mathbb{R}^{|\overline{E}|+1}$ to be separated?
2. In case a generated inequality valid for the completed graph $\overline{G}^e$ has nonzero coefficient a_e, how do we get rid of the coefficient a_e when we delete the artificial edges again?

Jünger, Reinelt and Rinaldi answered both questions [55]. We first answer 2. Let e be an artificial edge and

$$\sum_{i \neq e} a_i x_i + a_e x_e \leq a_0 \tag{2.9}$$

with $|a_e| \neq 0$ be an inequality valid for $P_C(\overline{G}^e)$. If we know another valid inequality with

$$\sum_{i \neq e} b_i x_i - a_e x_e \leq b_0 \tag{2.10}$$

then summing up (2.9) and (2.10) yields $\sum_{i \neq e}(a_i + b_i)x_i \leq a_0 + b_0$, and the coefficient of the artificial edge cancels. We choose the inequality (2.10) as follows.

Let $i, j \in \overline{V}$ with $(i, j) \notin \overline{E}$ and suppose we want to add the artificial edge (i, j) to the graph. It will become clear in the following that we can safely do this if we are given a cycle C in $\overline{G}$ with $i, j \in C$ and the additional requirement that there is a cycle inequality $\sum_{e \in F} x_e - \sum_{e \in C \setminus F} x_e \leq |F| - 1$ defined on C that is tight at $\overline{x}$.

Inserting edge $e = (i, j)$ makes C a chordal cycle consisting of two cycles $C_1 = u_1, \ldots, u_k, i, j$ and $C_2 = v_1, \ldots, v_r, i, j$, see Figure 2.18. Either C_1 or C_2 contains an odd number of edges from F, without loss of generality C_1.

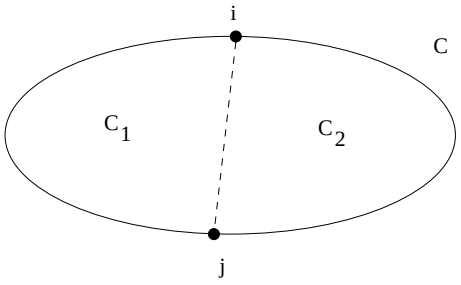


Figure 2.18: Schematic picture of the cycles C_1 and C_2.

We denote $F_i = \{e \in E \mid e \in C_i, e \in F\}$. We consider the cycle inequalities on C_1 and C_2

$$\sum_{e \in F_1} x_e - \sum_{e \in C_1 \setminus F_1} x_e - x_{ij} \leq |F_1| \tag{2.11}$$

$$\sum_{e \in F_2} x_e + x_{ij} - \sum_{e \in C_2 \setminus F_2} x_e \leq |F_2| + 1 \tag{2.12}$$

In case we encounter the artificial edge (i, j) with a positive (negative) coefficient in an inequality, we can add an appropriate positive multiple of inequality (2.11) ((2.12), respectively) to it and the coefficient of the artificial edge cancels. Thus we have answered question 2.

For answering 1., we extend $\overline{x}$ to $\overline{x}^e$ such that the two cycle inequalities (2.11) and (2.12) are both tight at $\overline{x}^e$. By this choice, we archieve the following. Suppose $\overline{x}^e$ violates the inequality (2.9) by $v > 0$. After having added an appropriate multiple of either (2.11) or (2.12), $\overline{x}$ also violates the projected inequality by v.

For the details, we denote the two paths P_1, P_2 as $P_1 = u_1, \ldots, u_k$ and $P_2 = v_1, \ldots, v_r$. Inserting $\overline{x}$ in the tight cycle inequality on C and rewriting it yields

$$\sum_{e \in F}(1 - \overline{x}_e) + \sum_{e \in C \setminus F} \overline{x}_e = 1. \tag{2.13}$$

Thus, if e is part of the odd set, it contributes $1 - \overline{x}_e$ to the left hand side of (2.13), otherwise it contributes $\overline{x}$. We define

$$x_1 = \sum_{e \in P_1, e \in F_1} (1 - \overline{x}_e) + \sum_{e \in P_1, e \in C_1 \setminus F_1} \overline{x}_e$$
$$x_2 = \sum_{e \in P_2, e \in F_2} (1 - \overline{x}_e) + \sum_{e \in P_2, e \in C_2 \setminus F_2} \overline{x}_e$$

It is $x_1 + x_2 = 1$, and x_i denotes the contribution of the path P_i to the left hand side of the cycle inequality.

We extend $\overline{x}$ to $\overline{x}^e$ by setting $\overline{x}_e = x_2$. Then both cycle inequalities (2.11) and (2.12) are tight at $\overline{x}^e$. (For C_1, (i, j) is in the odd set and thus contributes $1 - x_2$ to the inequality.) In the following section we explain how we do the lift–and–project procedure in practice.

2.5.2 Separation Procedure with Lift–and–Project

We setup the separation procedure as follows. In order to keep the program flexible, we add in each round of separation at most s inequalities to the problem formulation. (For three-dimensional spin-glass instances $s = 300$ is a good choice.) We start by generating the cycle polytope. If we find less than s violated cycle inequalities we call the lift-project procedure. We shrink and complete the sparse input graph. In the shrunken and completed graph $\overline{G}^c$ we expect to find violated bicycle-wheel inequalities. We separate them with the polynomial separation routine explained in Chapter 1 and add the generated violated inequalities to the problem formulation. It turns out in practice that the shrunken and completed graph is dense but by far not complete. In order to be able to use the knowledge on the cut polytope for complete graphs, we determine big complete subgraphs (cliques) in $\overline{G}^c$ and later lift the inequalities generated on these cliques. As the problem of determining a maximum clique in a graph is an $\mathcal{NP}$-hard problem we use a heuristic. It is advantageous to generate the cliques such that each node in the shrunken graph is part of at least one clique.

Suppose we have identified a clique $K_k = (V_k, E_k)$ on k nodes in the shrunken and completed graph $\overline{G}^c$. Let $x^c \in \mathbb{R}^{\binom{k}{2}}$ the point to be separated that is projected on the

clique edges. For our range of instance sizes, the cliques usually are of a size $k \leq 20$. How can we apply a separation procedure on these small cliques? Facets of the cut polytope for complete graphs K_k with $k \leq 9$ are compiled in the SMAPO-library [114] modulo permutation of the nodes and modulo switching, i.e., each inequality mentioned there occurs in the description once for each node permutation and for each possible switching along a cut. Additionally, for each inequality the number of roots, i.e., the number of cut vectors that is contained in it, is computed. $P_C(K_5)$ is described by 56 facets, where 40 facets are at a vertex. The facets are either triangle (1.12) or clique inequalities (1.20). In addition to the triangle facets and the clique facets on the six K_5-subgraphs contained in K_6, $P_C(K_6)$ contains a hypermetric inequality. $P_C(K_6)$ contains 368 facets, and 210 facets are at one vertex. The facial description of the cut polytope is completely known up to $k \leq 7$, [42] and contained in SMAPO. $P_C(K_7)$ is described by 11 classes of facets. In total these are 116764 facets For K_8, the description of $P_C(K_8)$ contained in SMAPO is conjectured to be complete. It consists of 147 classes of inequalities yielding 217093472 facets in total. 49604520 facets are at a vertex. We see that the problem becomes highly degenerate. The description of $P_C(K_9)$ in SMAPO is possibly complete and contains at least 164506 classes yielding 12246651158320 facets in total. The number of facets that are needed to describe the cut polytope $P_C(K_k)$ grows very fast with k.

We setup the separation procedure on a subgraph K_k as follows. For $k = 5$ we just check whether the clique inequality (or a switched version of it) is violated. We assume that an inequality with many roots is stronger than an inequality with a small number of roots, so for $6 \leq k \leq 9$ we first sort the inequalities in $P_C(K_k)$ with regard to the number of roots. As long as there is space in the constraint buffer, we do the following steps. For an inequality contained in the SMAPO-description we have to determine a node permutation and a switching optimizing the violation of the resulting inequality. For $k = 6$, the number of facets is small enough that testing all node permutations is fast. For $k > 6$ instead, it is described in [25] that determining the best node permutation with regard to the violation reduces to solving a quadratic assignment problem (QAP). The QAP is a hard problem, both in theory and in practice. We use the GRASP heuristic presented in [25] for determining a good node permutation that we summarize now. For an inequality, we sort both its coefficients a_i and the entries of vector $\overline{x}^c$ in decreasing order. From the d largest products $a_{s_1t_1}\overline{x}^c_{i_1j_1}, \ldots, a_{s_dt_d}\overline{x}^c_{i_dj_d}$ in sorted order we choose randomly one of them, say $a_{s_lt_l}\overline{x}^c_{i_lj_l}$, and initialize a list L of assignments as $L = \{(s_l, i_l), (t_l, j_l)\}$. We find the assignments for the other nodes step by step. In each step we setup a heap. For all possible assignments (e, f) we insert (e, f) into the heap with a score that reflects the contribution to the left hand side of the inequality in case e would be assigned to f. The score of (e, f) is thus $\sum_{(q,r)\in L} \overline{x}^c_{fr}a_{qe} + \overline{x}^c_{rf}a_{eq}$. We randomly choose an assignment from the p entries with highest score in the heap and insert it into the list of assignments L. After having constructed the assignment in L we can

apply a local search procedure that tests whether the assignment can be improved by permuting any two assigned pairs.

Determining a switching that maximises the violation of the inequality reduces to a max-cut problem on K_k, see Chapter 1. For the determined node permutation we determine a good switched inequality heuristically by applying Algorithm 3 from Chapter 1. In case the resulting inequality is violated by $\overline{x}^c$ we store it. In case the size of the clique is bigger than 9, we first heuristically separate hypermetric inequalities on K_k. Then we check whether a (switched) clique inequality on k nodes is violated on K_k. Subsequently we make use of the inequalities compiled in SMAPO, where we take as input complete subgraphs K_l with $l \leq 9$. We summarize the procedure in the algorithms below.

Algorithm: *lift-project_separation*
shrink edges e with $x^\star_e = 0$ or $x^\star_e = 1$
complete the graph by adding artificial edges
end up with $\overline{G}^c$
separate bicycle-wheel inequalities in $\overline{G}^c$
construct-cliques in $\overline{G}^c$
foreach *clique* K_k **do**
    separate(K_k)
    zero-lift inequalities on K_k to inequalities on $\overline{G}^c$
delete artificial edges again
unshrink graph and lift inequalities

Algorithm 7: Separation procedure based on lift-project.

With $K_{1,\ldots,l}$ we denote the clique consisting of the nodes $1, \ldots, l$. The function *separate(K_k)* then is formulated in Algorithm 8.

Algorithm: *separate(K_k)*
if $k > 9$ **then**
    heuristically separate hypermetric inequalities
    separate clique inequality on K_k
    $SMAPO(K_{1,\ldots,9})$
    $SMAPO(K_{k-8,\ldots,k})$
else
    $SMAPO(K_k)$

Algorithm 8: Separation routine for small complete graphs K_k.

The algorithm *SMAPO* using the SMAPO library is outlined below.

Finally, we summarize the GRASP heuristic *construct-cliques* for generating cliques. We sort the nodes with regard to their degress in descending order. We start with a

Algorithm: *SMAPO*(K_k)
if $k > 6$ **then**
  sort inequalities in $P_C(K_k)$ wrt. number of roots
  foreach *inequality i in sorted order* **do**
    determine good node permutation with GRASP heuristic
    for this permutation determine good switching with Algorithm 3
    if *ineq. violated* **then**
      add inequality to constraint buffer
if $k = 6$ **then**
  sort inequalities in $P_C(K_k)$ wrt. number of roots
  foreach *inequality i in sorted order* **do**
    foreach *node permutation* **do**
      determine good switching with Algorithm 3
      if *ineq. violated* **then**
        add inequality to constraint buffer
if $k = 5$ **then**
  check clique inequality on K_5 for violation

Algorithm 9: Separation routine using the inequalities contained in the SMAPO library.

triangle that has the maxium sum of node degrees among all triangles. Node by node we increase the size of the clique. In each step we generate the extension candidates, i.e., the non-clique nodes that are connected to all nodes in the clique. We choose randomly a node from the five candidates with highest degrees and add it to the clique. If we cannot extend the size of the clique any further we check whether a one-node-exchange of a node in the clique with a node not in it yields a clique of bigger size. We stop if we cannot increase the size of the clique any further.

2.5.3 Computational Results

We are mainly interested in the results for three-dimensional spin-glass instances. We call the nodes in the shrunken graph *active nodes* and the edges *active edges*. We show results for Ising spin-glass instances of size $5^3, 6^3$ with $\pm J$ distribution and of size 7^3 with Gaussian distribution. We choose periodic boundaries. We consider 10 randomly chosen instances per size. First we study the shrunken graph. During the run of the algorithm, the generated shrunken graphs get denser. This is intuitive as we expect to find more 'already decided' edges on which the lp solution vectors attain value zero or one. Furthermore, we expect to find more tight cycle inequalities in the graph completion at a later stage of the algorithm than at the beginning. For the $\pm J$ instances of size 5^3 on average we have 93 ± 1 active edges and a graph density of $\sim 30\%$. (161 ± 1 active edges and a density of $\sim 27\%$ for 7^3 Gaussian instances.) For both classes of instances, roughly 80% of the present edges are artificial and are added in the graph completion process.

In the shrunken and completed graphs cliques are generated. In practice, both for $\pm J$ and Gaussian instances the average clique size increases only slightly during the run of the algorithm, but the number of found cliques grows. This is intuitive as the shrunken graphs get denser. However, the effect is not strong enough to result in cliques of significantively bigger size. On average, for $\pm J$ distributed instances the cliques have size 10.8 ± 0.3. On average, 29 ± 1 cliques are generated. (For the Gaussian instances of size 7^3 we generate 52.5 ± 0.1 cliques of average size 11.41 ± 0.03. In case we only shrink the graph and skip the graph completion procedure the graph does not contain big enough cliques for being able to separate enough interesting inequalities.

After shrinking and completing graph we separate the bicycle-wheel inequalities. In Table 2.13 we report the average number of violated inequalities generated in each round of separation.

We skip the numbers for separating $P_C(K_6)$ and $P_C(K_7)$ with the SMAPO inequalities as those heuristics are only called if the cliques have fewer than eight nodes. The numbers for the Gaussian distributed instances are comparable and skipped here. In Table 2.14 we report the average violation of the inequalities for the Gaussian distributed instances of size 7^3. Numbers for $\pm J$ instances are comparable.

bicycle wheels	hypermetrics	big cliques	$P_C(K_{1,\dots,8})$	$P_C(K_{k-7,\dots,k})$
11 ± 1	0.39 ± 0.04	0.4 ± 0.03	15 ± 1	23 ± 2

Table 2.13: Number of generated inequalities per round of separation for the $\pm J$ instances of size 5^3.

The average violation of the inequalities generated in the SMAPO separation is bigger the bigger k is and is smallest for $P_C(K_6)$. The violation of inequalities generated in $P_C(K_{1,\dots,8})$ and $P_C(K_{k-7,\dots,k})$ is comparable as expected. The violation of the

$P_C(K_6)$	$P_C(K_7)$	$P_C(K_{1,\dots,8})$	$P_C(K_{k-7,\dots,k})$	bicycle-wheels
0.023 ± 0.001	0.036 ± 0.002	0.066 ± 0.005	0.061 ± 0.001	0.057 ± 0.007

Table 2.14: Average violation of the generated inequalities for 7^3 Gaussian instances.

bicycle-wheel inequalities is comparable to that of $P_C(K_{1,\dots,8})$. From the numbers in 2.14 we see no clear quality difference between the SMAPO and the bicycle-wheel inequalities. We conclude that we do find violated inequalities within the lift-project approach.

We now compare the behavior of branch–and–cut with and without the lift-project procedure. For both Gaussian and $\pm J$ distributed instances, the upper bounds improve through the lift-project procedure. Furthermore, branching is delayed as expected. However, the improvement in the bound is worse than the improvement we would archieve through branching. Usually, branching at a variable is better than tightening the cycle relaxation by adding the inequalities obtained through lift–and–project. In Figures 2.19 and 2.20 we show the development of the lower and local upper bounds during the run of the algorithm for two examples with $\pm J$ distribution and size 5^3. The development of the bounds is comparable for Gaussian instances. In Figure 2.21 we show an example for an instance of size 7^3.

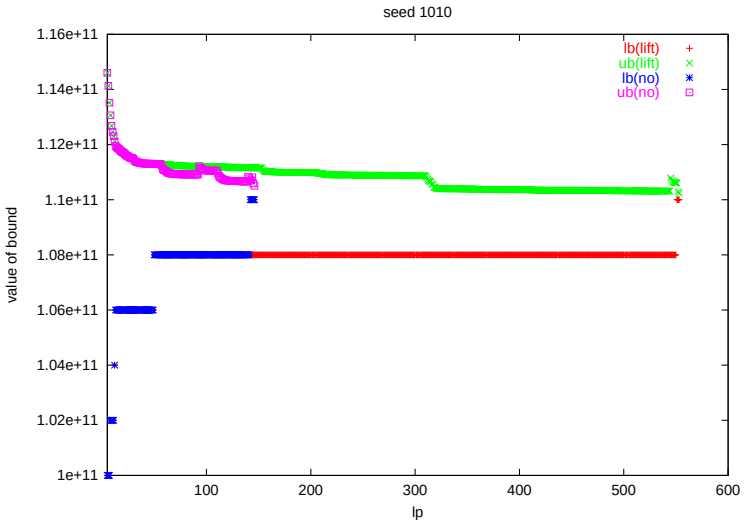


Figure 2.19: Example for the development of the lower and local upper bounds for two three-dimensional $\pm J$ spin-glass instances of size 5^3.

In Tables 2.15 and 2.16 we show the average number of linear problems and the number of sub problems solved. We compare the runs with and without lift–and–project. The number of sub problems is reduced when lift–and–project is included. The number of linear problems increases as branching is delayed and more rounds of separation are done in a sub problem. However, the running time increases sig-

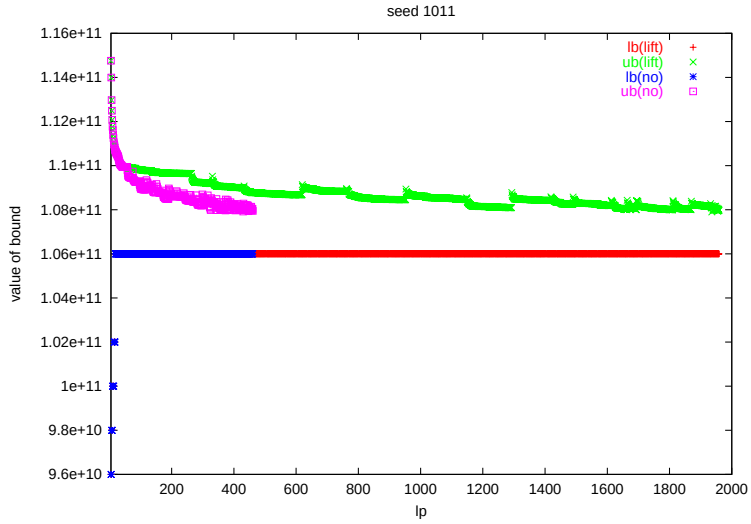


Figure 2.20: Another example for the development of the lower and local upper bounds for two three-dimensional $\pm J$ spin-glass instances of size 5^3.

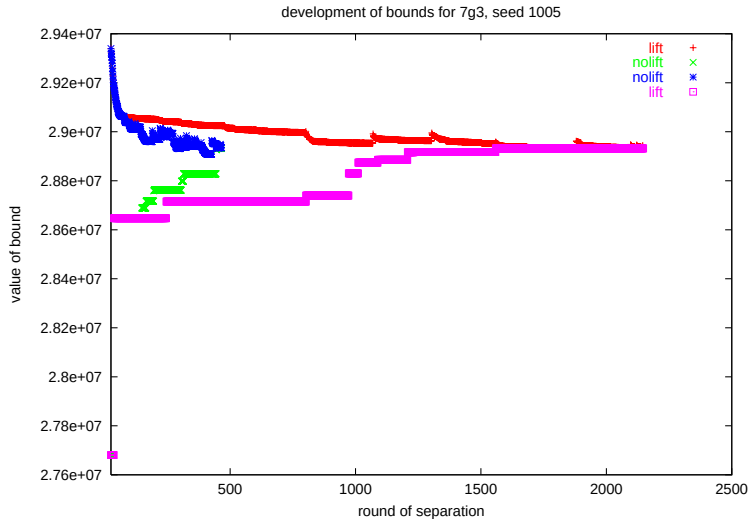


Figure 2.21: Example for the development of the lower and local upper bounds for a three-dimensional Gaussian spin-glass instances of size 7^3.

nificantly when lift–and–project is included. Especially the graph completion step of the algorithm takes long.

	# lps	# subs	
nolift	77 ± 24	5 ± 2	20 ± 6
lift–and–project	352 ± 188	3 ± 2	1121 ± 673

Table 2.15: Average number of linear problems, sub problems and CPU times for $\pm J$ distributed instances for size 5^3.

	# lps	# subs	
nolift	145 ± 56	7 ± 4	138 ± 55
lift–and–project	545 ± 296	3.4 ± 1.6	9573 ± 5799

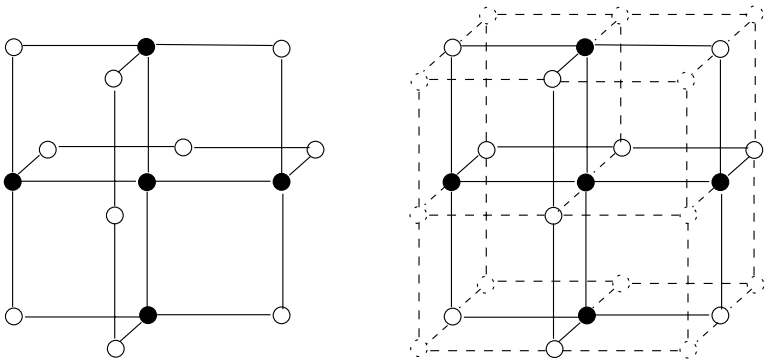
Table 2.16: Average number of linear problems, sub problems and CPU times for Gaussian distributed instances of size 7^3.

Whereas the number of sub problems can be significantively reduced by the lifting procedure, the needed CPU time increases strongly. The main reason for the long running time is that the graph completion by adding artificial edges takes long. Here we do a modification of the exact cycle separation from which we already know that it takes long from Section 2.2.4. It would certainly be possible to reduce the running time of the lift-project procedure, e.g., by doing a heuristic graph completion. However, as the improvement in the upper bound is not very strong, we assume that it would be hard to make the overall performance of the program with the lift–and–project approach competitive.

2.5.4 Lifted Clique Inequalities

In this section we apply the lift–and–project idea for a class of inequalities, the clique inequalities, in a more direct way than described in Section 2.5 above. We are mainly interested in max-cut problems on three-dimensional grid graphs. The cut polytope $P_C(G)$ coincides with the cycle polytope $P_{cyc}(G)$ in case the graph G has no K_5-minor. (A graph H is a *minor* of G if it obtained from G by a sequence of edge deletions and contractions.) We expect interesting inequalities from a subgraph of a three-dimensional grid having a K_5-minor. The smallest such subgraph (modulo symmetry) is displayed in Figure 2.22. Black nodes are present in K_5, white nodes are artificial. We call this graph $G_1 = (V_1, E_1)$.

The cut polytope of K_5 consists of the triangle inequalities (forming the cycle polytope) together with all possible switchings of the clique inequality $\sum_{e \in K_5} x_e \leq 6$ on the clique K_5. We apply the lifting and unshrinking procedure from Definition 2.4 starting from the clique inequality on K_5, unshrinking the graph and lifting the clique inequality until we end up with G_1 and a corresponding *lifted clique inequality*.



(a) Only edges of the subgraph G_1 are shown.

(b) The subgraph G_1 from Figure 2.22(b) embedded in the grid.

Figure 2.22: The smallest subgraph G_1 in a three-dimensional grid having a K_5-minor. Black nodes are present in K_5, white nodes are artificial. Dashed nodes and edges are not part of the subgraph.

Definition 2.8. *We call an inequality* lifted clique inequality *if it is obtained from a clique inequality by the lifting procedure of Definition 2.4.*

Lemma 2.9. *The inequality*

$$\sum_{e \in E_1} \pm x_e \leq 10, \tag{2.14}$$

where the signs can be read off from Figure 2.23, is a lifted clique inequality valid for $P_C(G_1)$ obtained from lifting $\sum_{e\in K_5} x_e \leq 6$. Furthermore, (2.14) is a facet of the cut polytope $P_C(G_1)$.

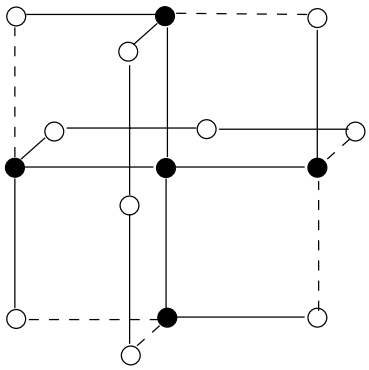


Figure 2.23: A lifted clique inequality on G_1. Solid edges have coefficient $+1$, dashed edges -1.

Theoretically, we can determine the facial description of a polytope from its incidence vertices e.g., by Fourier-Motzkin elimination which is a finite procedure. Practically efficient implementations exist yielding the facial description of 'small' polytopes, e.g., the program PORTA [107]. This program has exponential running time and determining the facial description of $P_C(G_1)$ takes too long to be solved by PORTA. However, let us consider the smaller graph $G_2 = (V_2, E_2)$ shown in Figure 2.24. We can prove the following result for the cut polytope $P_C(G_2)$.

Lemma 2.10. *Let the graph $G_2 = (V_2, E_2)$ be the graph displayed in Figure 2.24. The inequality*

$$\sum_{e\in E_2, e\notin\{e_1,e_2\}} x_e - x_{e_1} - x_{e_2} \leq 10, \tag{2.15}$$

with the edges e_1, e_2 shown in Figure 2.24, is a lifted clique inequality. (2.15) is a facet of the cut polytope $P_C(G_2)$.

Proof. We apply the lifting procedure from Definition 2.4 starting from the clique inequality $\sum_{e\in K_5} x_e \leq 6$ on K_5 as in Figure 2.25(a).
We set $N = \emptyset$, $S = \{s_1, s_2, s_3\}$ and $T = \{t_1\}$ and lift the clique inequality to

$$\sum_{e, e\neq(s,t)} x_e - x_{st} \leq 6$$

The graph after one step of unshrinking is displayed in Figure 2.25(b).

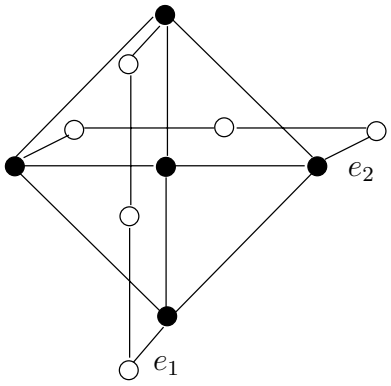
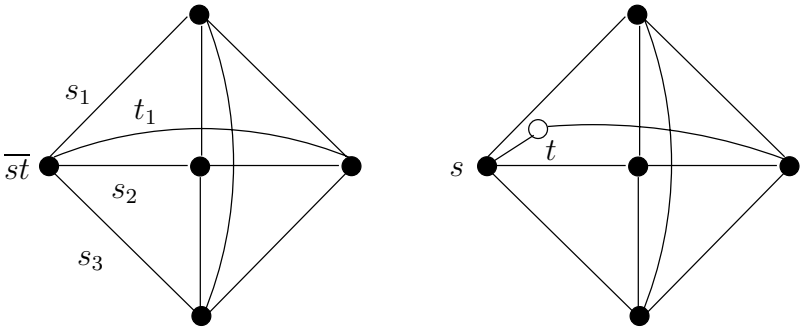


Figure 2.24: The subgraph $G_2 = (V_2, E_2)$ embedded in a three-dimensional grid having a K_5-minor plus some additional edges along the grid diagonals not present in the grid. Black nodes are present in K_5, white nodes are artificial. Grid edges are not shown.



(a) Complete graph K_5.

(b) Graph after one unshrink step.

Figure 2.25: Unshrinking K_5.

By analogous appropriate unshrink steps of the graph, simultaneous liftings and switchings along appropriate cuts, we end up with inequality (2.15). □

Proof of Lemma 2.9. Follows from the proof of Lemma 2.10 together with four times applying appropriate liftings of inequality (2.15). □

Furthermore, we can show by Fourier-Motzkin eliminination the following result on $P_C(G_2)$.

Lemma 2.11. *The cut polytope $P_C(G_2)$ consists of the cycle inequalities together with all switchings of the lifted clique inequality (2.15).*

We conjecture that the cut polytope $P_C(G_1)$ consists of the cycle polytope together with the lifted clique inequality.

Conjecture 2.12. *The cut polytope $P_C(G_1)$ consists of the cycle inequalities together with all switchings of the lifted clique inequality obtained from lifting $\sum_{e \in K_5} x_e \leq 6$.*

We implement the separation of lifted clique inequalities in three-dimensional spin-glass instances as follows. We scan through the grid and check whether violated lifted clique inequalities exist on the grid structures G_1. However, in practice the lifted inequalities are not strong for spin-glass instances. In case we start with separating cycle inequalities first, we almost never find violated lifted clique facets subsequently. We occasionally find a violated lifted clique if we start separating them early in the optimization process. However, the improvement in the bound is marginal.

In order to find out about the strength of the lifted clique inequalities we also generated optimum (with regard to violation) switchings of lifted clique inequalities by solving the corresponding integer program exactly. We tried instances of sizes $4^3, 5^3, 6^3$. For example, by doing this for a $5^3 \pm J$ spin-glass the number of linear problems reduces from 257 to 235 when lifted clique inequalities are separated and switched by a cut maximising the violation of the resulting inequality. (As can expected, the running time goes up considerably when determining optimum switchings. For our example, it increases from 40 seconds to 50 minutes). Thus we do not separate these inequalities within the branch–and–cut framework.

2.6 Primal Heuristics

Whereas in the last sections we were concerned with generating tight approximations of the cut polytope, we explain in this section how we can generate good primal solutions, i.e. good cuts within a branch–and–cut framework.

A graph $G = (V, E)$ is called *k-regular* if the degree of all nodes equals k. We can prove the following lemma on the value of a cut in a k-regular graph.

Lemma 2.13. *Let $G = (V, E)$ be a k-regular graph with $k \in \mathbb{N}$ and let the edge weights be chosen as $\{-t, t\}$ with $t \in \mathbb{N}$. Let k or t be an even number. Then for any $W \subseteq V$, the cut $\delta(W)$ has even value.*

Proof. Let a node set W and the cut $\delta(W)$ with value z_{cut} be given. We start from $\widetilde{W} := \emptyset$ and possibly add node after node to $\widetilde{W}$ until we have $\widetilde{W} = W$. Thus, we execute the algorithm

$\widetilde{W} = \emptyset$
$\widetilde{z}_{\text{cut}} = 0$
for *all nodes* $v \in V$ **do**
  if $v \in W$ **then**
    $\widetilde{W} = \widetilde{W} \cup \{v\}$
    $\widetilde{z}_{\text{cut}} = \widetilde{z}_{\text{cut}} + \Delta\widetilde{z}_{\text{cut}}$
$W = \widetilde{W}$
$z_{\text{cut}} = \widetilde{z}_{\text{cut}}$

We prove that $\Delta\widetilde{z}_{\text{cut}}$ is even in each update step. Assume $v \in W$ in the loop. Then the change of the cut value is $\Delta\widetilde{z}_{\text{cut}} = k_1 t - k_2 t$, where k_1 denotes the number of edges entering the cut and k_2 the number of edges leaving it when setting $\widetilde{W} = \widetilde{W} \cup \{v\}$. As $k_1 + k_2 = k$, we have

$$\Delta\widetilde{z}_{\text{cut}} = (2k_1 - k)t,$$

which is even. □

In particular, for two- and three-dimensional grids with $\pm J$ distribution, the maximum cut has even value. We can make use of this fact during the run of the branch–and–cut algorithm. Whenever the distance between the best known cut and the local upper bound is smaller than 2, we can fathom the node. Therefore, for $\pm J$ distributed instances knowing a good (or even optimum) cut early in the computation process helps reducing the running time.

We do not want to consider the upper and lower bound computations separately. Instead, we want to generate good cuts by making use of the optimum solution $x^\star$ of the current linear programming relaxation determining the upper bound. In a later

part of the optimization process, hopefully $x^\star$ is 'not too far away' from a good (or optimum) cut x', and we want to obtain x' from $x^\star$ by appropriate rounding of the latter. Obviously, coefficientwise rounding the entries in $x^\star$ is unlikely to yield a cut. However, we formulate an observation.

Observation 2.14. *A cut in a graph $G = (V, E)$ is determined by its values on a spanning tree $T = (V, E_T)$, with $E_T \subseteq E$.*

Therefore, we can generate a cut by determining a spanning tree T in the graph G and rounding the $x^\star$-values on the tree to the nearest integer. Then we determine the resulting cut values on the non-tree edges. We assume that the resulting cut is best if we include in T the edges that are 'nearly decided' by the lp solution vector, i.e., edges e with $x_e^\star \approx 0$ or $x_e^\star \approx 1$. To this end, we determine a maximum weight spanning tree, with edge weight chosen as $|x_e^\star - \frac{1}{2}|$ for $e \in E$. Finally, we can improve the resulting cut x' through a local search procedure, e.g., a one-node exchange heuristic. This algorithm is already used in the branch–and–cut program. It is described in detail elsewhere [10] and gives good results. We summarize it in the following.

Algorithm: *tree_heuristic*

determine a maximum weight spanning tree T in G, where for edge $e \in E$ the edge weight is chosen as $w_e = |x_e^\star - \frac{1}{2}|$
round the $x^\star$-values on the tree edges to the nearest integer
determine the resulting cut x'

Algorithm 10: *Tree_heuristic* for generating good cuts.

If we use Kruskal's algorithm for determining a maximum weight spanning tree, *tree_heuristic* runs in $\mathcal{O}(m \log n)$ time. In order to estimate the quality of the above procedure for ± 1 distributed instances, we experimentally compare maximum cuts with the cuts generated by *tree_heuristic*. Comparing the solutions in the beginning of the optimization process is not useful. Instead, we take the heuristic solution generated in the iteration i in which the upper bound falls the first time below $z_{\text{cut}} + 2$. If we know an optimum cut, we can prove its optimality in iteration i.

Surprisingly, the number of tree edges in which the optimum and heuristic solutions differ is small. As an example, for three-dimensional instances of size 6^3, 215 edges are in the tree. For 50 randomly chosen instances with $\pm J$ distribution and periodic boundaries, *tree_heuristic* chooses only 15 ± 3 ($\sim 6.98 \pm 0.01\%$) wrong values on the tree. The tree edges on which the optimum and heuristic solutions differ are usually single edges or very short paths. In Figure 2.6 we show a typical example for the wrong decisions *tree_heuristic* takes on the tree generated by Algorithm 10 for a three-dimensional 6^3 instance. We aim at improving the heuristic cuts by correcting the errors in the decisions *tree_heuristic* took. We can do this in different ways.

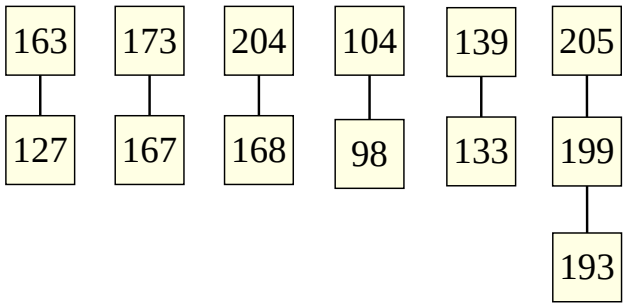


Figure 2.26: Typical example for the difference between the optimum solution and the heuristic solution for a 6^3 spin-glass instance. Shown are only the tree edges in which optimum und heuristic solutions differ.

One possibility is as follows. For the cut x' output by *tree_heuristic*, we check for each not yet fixed tree edge whether we can improve the cut by flipping x'_e to $1-x'_e$. Already fixed edges do not have to be taken into account. We experimentally find that the reduced cost of a tree edge gives a good measure whether flipping its cut value improves the solution. An edge e with $x'_e = 0$ ($x'_e = 1$) and highly positive (negative) reduced cost is likely to improve the solution when it is flipped to 1 (0). Thus, we first sort the tree edges with regard to their reduced cost and take first the promising edges into consideration. We find that it is enough to check the first $\frac{|V|}{2}$ tree edges in sorted order. We summarize the procedure in algorithm *subtree_flip*.

Algorithm: *subtree_flip*
call *tree_heuristic*
sort the tree edges with regard to the reduced cost
for *first* $\frac{V}{2}$ *edges* $e \in T$ *in sorted order* **do**
  if *the flip* $x'_e := 1 - x'_e$ *yields better cut* **then**
    update cut
  else
    flip back

Algorithm 11: *Subtree_flip* for generating better cuts.

The running time of *subtree_flip* depends on how fast the cut update is. As flipping the cut on a tree edge might force many other non-tree edges to also change its cut value, we calculate the new cut from scratch. This costs $\mathcal{O}(m)$ time. Thus, alltogether the running time of *subtree_flip* is $\mathcal{O}(nm)$.

We next show some running times for branch–and–cut including *subtree_flip*. For 50 randomly chosen three-dimensional $\pm J$ distributed instances of size $L = 6^3$, run on a 440 Mhz Sun Ultra Sparc, we get the running times in seconds shown in Table 2.17.

For a comparison, we also show the running times for the same instances run with *tree_heuristic*. When calling *subtree_flip* the number of lps (subs) and the running time of branch–and–cut can be reduced.

	# lps	# subs	CPU
tree_heuristic	562 ± 234	23 ± 8	234 ± 102
subtree_flip	315 ± 127	13 ± 4	167 ± 69

Table 2.17: Comparison of the performance of the branch–and–cut algorithm with *tree_heuristic* and *subtree_flip* respectively, for 50 randomly chosen three-dimensional $\pm J$ distributed spin-glass instances of size 6^3.

In a statistics of instances, there are samples where upper and lower bounds converge equally well and samples for which the known cut is poor. For these instances, *subtree_flip* can help reducing the running time. However, there are also samples in which the upper bound does not converge and for which *subtree_flip* won't help. We consider for the samples from Table 2.17 the five instances with worst lower bound, i.e., the five instances in which the optimum solution is found latest in the optimization process. We report the number of linear programs, the number of sub problems and the running times in Table 2.18 when branch–and–cut is run with *tree_heuristic* (denoted as (1)) or *subtree_flip* (denoted as (2)), respectively. For the instances in Table 2.18 the running time is roughly halved when *subtree_flip* is called.

seed	lps(1)	lps(2)	subs(1)	subs(2)	CPU(1)	CPU(2)
1054	341	164	21	11	131.07	81.07
1067	678	110	47	9	260.02	55.30
1070	30	12	1	1	12.79	5.59
1072	144	51	7	3	52.79	27.79
1081	406	223	11	7	154.66	123.98

Table 2.18: Comparison of branch–and–cut with *tree_heuristic* (1) and *subtree_flip* (2) for the instances with the worst known cut from the statistics of Table 2.17.

Another possibility of correcting the decisions *tree_heuristic* takes consists of formulating the problem of determining tree edges to be flipped as a linear constraint. Let $|V| = n$, x' the incidence vector of the cut generated by *tree_heuristic* and T the determined spanning tree. Assume without loss of generality that $x'_e = 1$ for all edges on the tree (this can always be obtained by appropriate switching.) Let $k \in \mathbb{N}$.

We formulate the restrictions

1. 'The values on exactly k tree edges should change' as $\sum_{e \in T} x_e = n - 1 - k$.
2. 'The values on at least k tree edges should change' as $\sum_{e \in T} x_e \leq n - 1 - k$.

3. 'The values on at most k tree edges should change' as $\sum_{e \in T} x_e \geq n - 1 - k$.

We add a constraint of the form 1., 2. or 3. to the current linear program and solve it. From its (possibly fractional) optimum solution we derive a cut as *tree_heuristic* does and check whether it improves the lower bound. We get the best results for constraints of the form 1., and we restrict ourselves to this case. As for each k the procedure requires the solution of a linear program (done with the dual simplex algorithm) which is time consuming, we have to restrict the computations to a small number of different k. For 50 instances of size 6^3 with $\pm J$ distribution we count the number of cut improvements for different values of k. We show the results as a histogram in Figure 2.27. Most improvements occur for small k. This is also intuitive: We expect that the optimum solution of the lp with the additional constraint will have more fractional values for bigger k than for small k. Then *tree_heuristic* might make additional mistakes. For studying the power of the procedure, we only consider $0 \leq k \leq 7$.

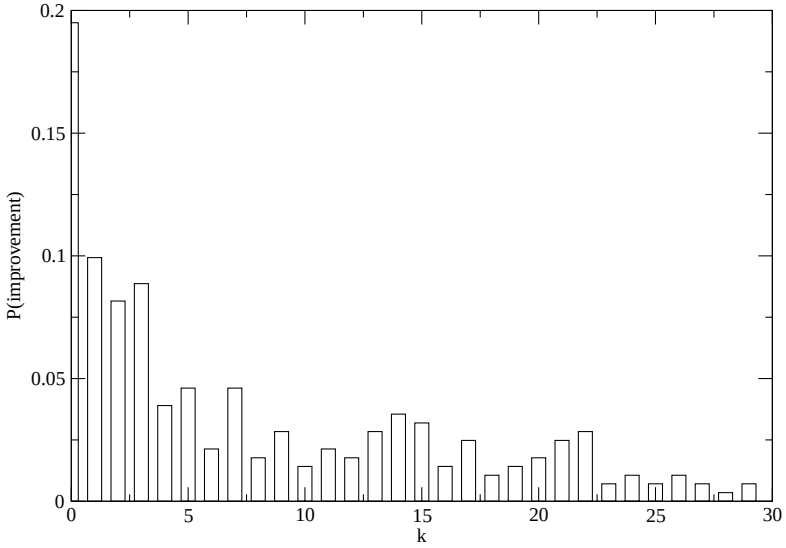


Figure 2.27: Histogram of the probability distribution for the cut improvement for different values of k for constraints of the form 1. We computed a statistics of 50 samples of size 6^3.

Following Fischetti and Lodi [37] who formulate branching rules in the same flavor as the heuristic we propose here, we call our cut generating heuristic *local_branching*. We compare the strength of *subtree_flip* and *local_branching*. For three-dimensional instances of size 6^3 we show for 10 randomly chosen instances in Table 2.19 the num-

ber of the lps that have to be solved until an optimum is found by *tree_heuristic* (denoted as (1)), *subtree_flip* (denoted as (2)) and *local_branching* (denoted as (3)). We find that *subtree_flip* usually finds an optimum solution earlier than *local_branching*. However, both heuristics find the optimum earlier than the old heuristic *tree_change*. *subtree_flip* is considerably faster than *local_branching*.

seed	# lps(1)	# lps(2)	# lps(3)
1010	89	14	52
1011	16	14	23
1012	21	11	15
1013	170	23	31
1014	78	25	29
1015	81	13	36
1016	1251	131	551
1017	30	14	28
1018	30	29	20
1019	193	14	40

Table 2.19: Number of lps that have to be solved before an optimum solution can be generated by *tree_heuristic* (1), *subtree_flip* (2) and *local_branching* (3) for instances of size 6^3.

We conclude that for three-dimensional $\pm J$ distributed instances, *subtree_flip* is well suited for generating good cuts, whereas *local_branching* is not only weaker but also needs more CPU time. We tested constraints of form 1.– 3. as branching rules within the branch–and–cut algorithm. However, the performance is worse than taking the most fractional variable as branching variable.

Usually two-dimensional distributed instances perform somewhat better if *subtree_flip* is *not* called. In order to improve the performance for two-dimensional spin-glass instances, we implement a faster heuristic *fast_flip* that works as follows. Starting from cut x' output by *tree_heuristic*, we sort the nodes with regard to their tree level in descending order. (The *tree level* of a node u is the length of the unique path from u to the root of T.) Starting from the nodes with higest level, the *leaves*, we flip for each node u in sorted order the value of the tree edge $(u, \text{father}(u))$. If this procedure improves the best known cut we update the lower bound accordingly. If not, we leave $(u, \text{father}(u))$ flipped and proceed to the next node in sorted order. One after the other we flip all tree edges and check for cut improvements until we encounter the root. As we don't flip the tree edges back, up to $|\text{width}(T)|$ edges might be flipped simultaneously. However, the cut update is fast. As flipping a tree edge $(u, \text{father}(u))$ only affects the edges adjacent to u, we only have to scan once through the adjacency list of u for the cut update which is fast. The running time of *fast_flip* is $\mathcal{O}(m \log n + n^2)$.

Algorithm: *fast_flip*
call *tree_heuristic*
sort the tree edges in decending order with regard to their tree level
for *nodes* $u \in V$ *in sorted order* **do**
  $e := (u, \text{father}(u))$
  $x'_e := 1 - x'_e$
  if *resulting cut is better* **then**
    update of cut

Algorithm 12: *Fast_flip* for generating good cuts fast.

instance	# lps(1)	# lps(2)	CPU(1)	CPU(2)
1390	31076	904	6395.8	197.92
1391	9554	9553	2785.97	2867.72
1392	3741	1775	1241.65	502.65
1393	7839	5493	2333.38	1676.14
1394	759	758	232.56	210.5
1395	1764	1484	656.42	537.43
1396	2042	1061	570.17	270.74

Table 2.20: Number of lps and CPU time for branch–and–cut with *tree_heuristic* and *fast_flip* respectively, for seven randomly chosen two-dimensional $\pm J$ distributed spin-glass instances of size 70×70.

In Table 2.20 we give the number of linear programs and the CPU time in seconds for seven randomly chosen $\pm J$ instances of size 70^2 for branch–and–cut with *tree_heuristic* (denoted as (1)) and *fast_flip* (denoted as (2)). For 200 randomly chosen instances, the average number of lps reduces from 1677 ± 488 to 1204 ± 217 and the CPU time from 166 ± 40 to 135 ± 22 seconds, when *tree_heuristic* is extended to *fast_flip*. Results for sizes 50^2 and 60^2 are comparable. The development of the primal bound is better with *fast_flip* than with *tree_heuristic*. As a conclusion of this section we note that we generate improved cuts by calling *fast_flip* for two-dimensional spin glasses and *subtree_flip* for three-dimensional Ising spin-glass instances.

Chapter 3

Branch–Cut& Price for the One-Dimensional Ising Chain

In this chapter we extend the branch–and–cut algorithm to a *branch–cut&price algorithm* for computing ground states of one-dimensional Ising chain instances. The model is fully connected. However, the interactions fall off with a nonnegative power σ of the spin distance. This work is motivated by the studies of H.G. Katzgraber and A.P. Young [63],[64]. In [63], the authors point out that the one-dimensional Ising chain model inherits several features making the study of its physical properties interesting. Firstly, large system sizes L can be studied that can be larger than the treatable system sizes for short-range models. Secondly, depending on how the parameter σ is chosen, the model either is an infinite-range Sherrington-Kirckpatrick (SK) model or a short-range model. Thus, only by varying *one* parameter, models with significantively different physical characteristics can be studied.

With respect to our algorithm, we have an additional motivation for studying this model, apart from its interesting physics. We can exploit its characteristics and obtain a considerable algorithmic speedup. In the following we introduce the model. Then we present the algorithmic details and show experimental results for the performance of the algorithm.

3.1 Model

The one-dimensional Ising chain model was already studied by Bray et al.[22] and Fisher and Huse [35]. It was rediscovered in 2002 by H. Katzgraber and A.P. Young,[63]. Following [63], we let the model consist of L spins lying equally spaced on a cycle of perimeter L, see Figure 3.1. We use a cycle instead of a chain for installing periodic boundary conditions. All spins i, j are connected with each other by a coupling strength J_{ij}.

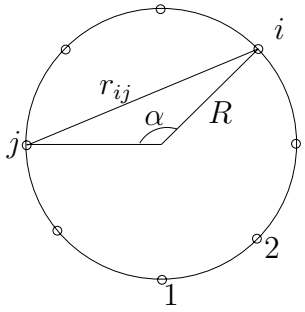


Figure 3.1: One-dimensional Ising chain model. The model is fully connected; not all bonds are shown.

The Hamiltonian H is of the usual form

$$H = -\sum_{i,j} J_{ij} S_i S_j. \tag{3.1}$$

The sum is over all pairs of spins on the cycle. The coupling J_{ij} between spins i and j is given by

$$J_{ij} = c(\sigma)\frac{\epsilon_{ij}}{r_{ij}^{\sigma}}, \tag{3.2}$$

where ϵ_{ij} is chosen according to a Gaussian distribution with zero mean and variance one. $c(\sigma)$ is a constant and r_{ij} denotes the distance of i and j on the chain. $c(\sigma)$ is chosen so that the mean-field transition temperature T_c^{MF} is equal to $T_c^{MF} = 1$.

The distance of spin i and j on the chain is $r_{ij} = 2R\sin(\frac{\alpha}{2})$, as can be seen in Figure 3.1. R is the radius of the cycle and α the angle between i and j. We can reformulate

$$r_{ij} = \frac{L}{\pi}\sin\Big(\frac{\pi \mid i-j \mid}{L}\Big). \tag{3.3}$$

For small nonnegative σ, $J_{ij} \approx \epsilon_{ij}$, and the model is the fully connected Sherrington-Kirckpatrick model. For σ chosen suitably big, the long-range couplings become unimportant and the problem is a short-range model.

3.2 Pricing

Following Chapter 1.1, we can transform the problem of determining an exact ground state of a one-dimensional Ising spin-glass instance to a max-cut problem on the corresponding interaction graph. Let us consider the max-cut problem for an instance coming from the one-dimensional Ising chain. The underlying graph is complete, and

for a system with $L = 100$ spins and $\sigma = 3.0$ the default version of the branch–and–cut algorithm needs roughly 7h cpu time on average on a 400 Mhz Athlon processor. In case we wanted to generate a reasonable ground-state statistics, we could go up to roughly this size. In [63] the authors use their parallel tempering Monte Carlo method for generating heuristic ground states of system sizes up to $L = 256$ and $0.1 \leq \sigma \leq 1.0$. With the default version of branch–and–cut, generating a statistics for this range of system sizes would probably not be possible within reasonable time.

As the couplings fall off with a nonnegative power of the spin distance, we may assume that for big enough σ the long-range couplings between two spins that are 'far apart' from each other won't affect the ground state. Hence, we may temporarily neglect the long-range edges with 'weak' couplings, as long as we make sure that the computations are correct at termination. As the running time of the algorithm basically scales with the number of edges, we expect a speed-up. In the following we explain the extension of the branch–and–cut algorithm to a *branch–cut&price algorithm*. The details about general pricing algorithms with ABACUS can be found in [56].

For solving an instance of the one-dimensional Ising chain, we start doing branch–and–cut. However, for the upper bound computations we work on a graph $G = (V, E)$ consisting of all nodes (spin sites) but only of a suitable subset of the edges. We add and delete edges (corresponding to the variables) dynamically during the optimization process. We refer to the edges in the current graph G as *active* edges and to the other edges as *nonactive* edges. Assume we have solved the current branch-and-bound node, having an lp solution vector at hand that is optimum for the relaxation defined on G. The lp solution is an upper bound on the max-cut value on G (maybe the bound is only locally valid in case variables are already set in the current sub problem). However, it is not clear whether the lp solution is also a (local) upper bound on the max-cut value for the original problem. Hence, before we fathom the node, we do a check as follows: For a nonactive edge e with variable $x_e (= 0)$, we compute its *reduced cost* (see formula (4.1)) that determines the rate of change of the objective function when x_e changes from zero to a nonzero value. If all nonactive variables have nonpositive reduced cost, the (local) bound on the max-cut problem defined on G is also an upper bound for the complete problem, and we can fathom the node. We say all nonactive variables *price out correctly*. In case we do find a nonactive variable with positive reduced cost, it is possible that including it into the problem formulation yields a better solution than we have at hand. Therefore, a variable with positive reduced cost is added to the sparse problem formulation and the new lp is solved. For reoptimizing the lp, we apply the primal simplex algorithm as the current solution is primal feasible. The corresponding edge is inserted in G. Figure 3.2 shows the flowchart of the branch-cut&price algorithm. With `lub` we denote a (local) upper bound generated in a sub problem. `gub` denotes the globally valid global upper bound. `lb` denotes the lower bound, i.e., the value of the best known cut.

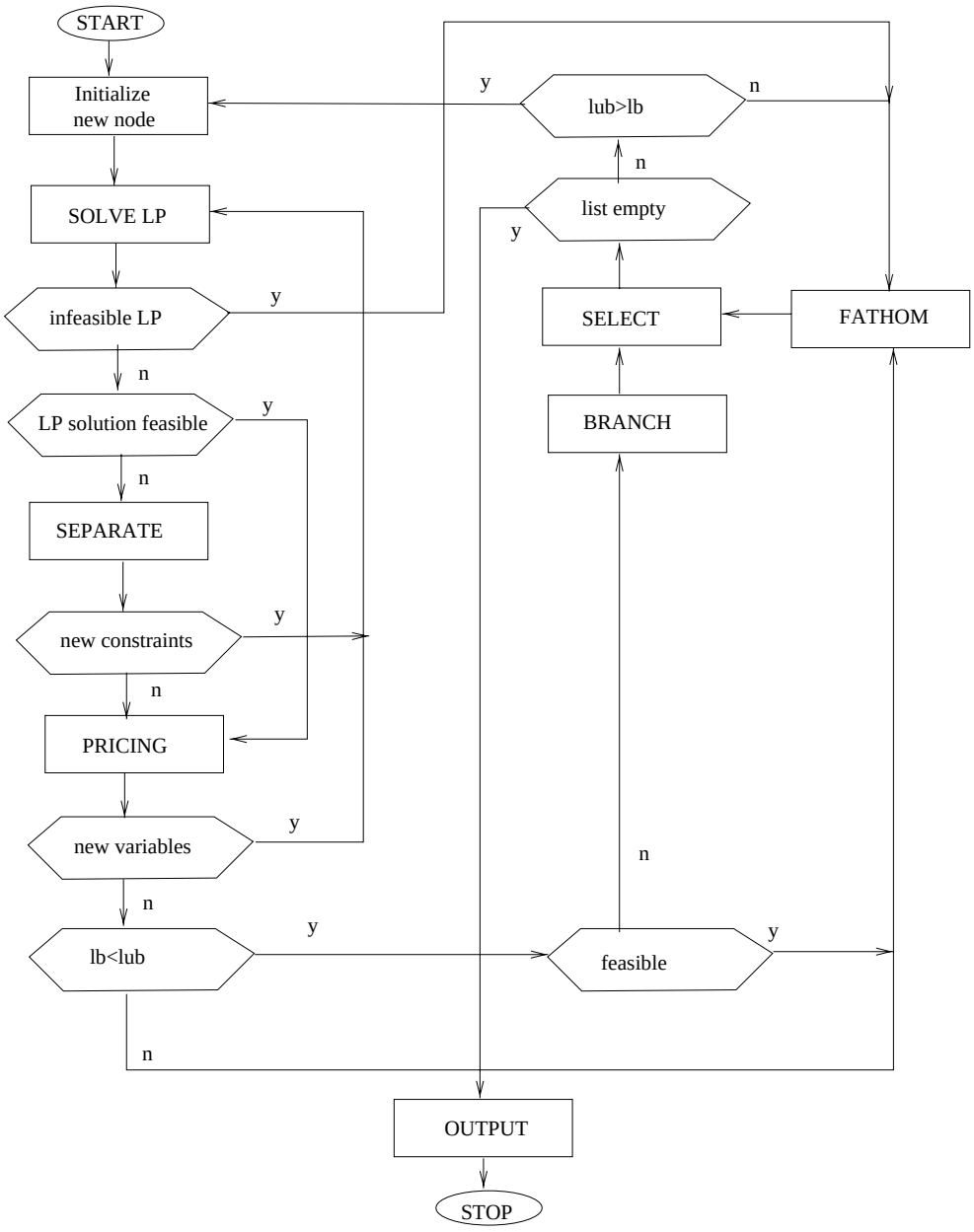


Figure 3.2: Flowchart of the branch-cut& price algorithm.

Within a pricing algorithm, variables must be added if indicated by the reduced cost or if the linear program turns out to be infeasible. We have explained the former above. The latter might occur under the following circumstances: As fixed and set variables are removed from the linear program, some inequality only involving fixed and set variables could have a void left hand side but could be violated. In case there exists a nonactive variable with nonpositive coefficient, the violation might be removed by adding a nonactive variable to the problem formulation. However, this case does not occur for our max-cut computations. We separate inequalities defined on the current support graph. When a variable x_e is added to the problem formulation, we zero-lift the inequalities present, i.e., the coefficients of e in the current set of inequalities are all set to zero.

Different possibilities exist for generating the sparse input graph G:

1. G consists of an *r-neighbour graph*, i.e., the set E contains the edges $(i, i+1), (i, i+2), \ldots, (i, i+r)$ for all nodes i. Indices are taken modulo $|V|$ and r is suitably chosen.

2. G consists of the $k\%$ edges with highest weights, measured in absolute value.

In our tests, choice 2. performs best. For $\sigma = 3.0$ it is best to set $k = 20$. Other values for k or option 1. with various values for parameter r are slower.

In Table 3.1 we show the number of lps solved (# lps) and CPU times in seconds, averaged over ten randomly chosen instances, for different sizes L and $\sigma = 3.0$, with and without pricing respectively, run for the same set of instances. We denote with (1) the branch–and–cut algorithm without pricing and with (2) the branch-cut&price algorithm. For (1), the branch–and–cut algorithm operates on the complete problem formulation. We have not attempted to run instances of size bigger than $L = 100$ without pricing. The runs were performed on a 440 Mhz Sun Ultra Sparc. From the numbers is it obvious that the running times are considerably improved when pricing is included. Whereas a problem with $L = 100$ spins might already be a hard problem for branch–and–cut, we can solve systems with $L = 280$ spins with branch-cut&price within reasonable time. The number of linear programs seems not to be as strongly correlated with the CPU time as for branch–and–cut results. (For comparison, see Chapter 6, where we find that the number of lps can be used as a performance measure for branch–and–cut.) However, after having generated more data (not shown here) it became clear that this phenomenon is due to too small statistics.

3.3 Cutting Planes

Within the branch-cut&price algorithm for the one-dimensional Ising chain, we can improve the separation procedure resulting in a tighter approximation of the cut

L	# lps(1)	# lps(2)	CPU(1)	CPU(2)
40	516 ± 102	552 ± 24	63 ± 18	6 ± 1
60	2481 ± 207	1178 ± 29	782 ± 105	47 ± 5
100	23474 ± 5357	2464 ± 145	20040 ± 1639	426 ± 55
200		2706 ± 76		2395 ± 78
280		2187 ± 97		5161 ± 275

Table 3.1: Performance of branch–and–cut (denoted as (1)) and of branch-cut&price (denoted as (2)), averaged over ten randomly chosen instances of different sizes L and $\sigma = 3.0$.

polytope. As explained in the pricing step above, we start with a sparse input graph $G = (V, E)$. We expect that an r-nearest neighbour graph is a subgraph of G. Let r be taken maximum, i.e., G does not contain an $(r+1)$-neighbour graph. (This is clear if G is constructed by option 1. Otherwise, when G contains the edges with highest absolute weights (option 2.), we also expect an r-neighbour graph to be a subgraph of G because of the choice of the weights.)

In Figure 3.3 we show an example for an input graph G consisting of 9 nodes that contains a 3-neighbour subgraph.

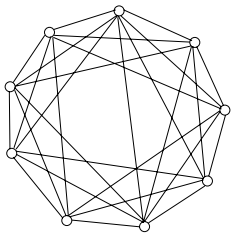


Figure 3.3: Example for an input graph G with 9 nodes and a 3-neighbour subgraph.

An r-nearest neighbour graph with n nodes, $r \geq 2$, has $n\binom{r}{2}$ triangles that can be enumerated in $\mathcal{O}(nr^2)$ time. The corresponding triangle inequalities (1.12) can be checked for violation. In case the triangle separation cannot generate enough violated inequalities, we call the forest heuristic for generating general violated cycle inequalities. If we cannot find enough violated inequalities, we call the odd cycle separation routine randomly for a fraction $\frac{1}{n}$ of the nodes. If the current graph has an r-nearest neighbour subgraph with r at least 2, violated circulant and parachute inequalities (for the definition of these inequalities, see Chapter 1.3) might be present. Separating circulant inequalities is $\mathcal{NP}$-hard. We explain a heuristic separation for the circulant inequalities that runs in polynomial time.

For convencience, we repeat again Theorem 1.13: A circulant $C(n, r)$ is a graph consisting of n nodes $1, \ldots, n$ and the edges $(i, i+1), (i, i+r)$ for all nodes $i = 1, \ldots, n$, with indices taken modulo n. If $n = kr + 1$ with $k, r \geq 2$ even, then

$$\sum_{ij \in C(n,r)} x_{ij} \leq 2n - k - r \tag{3.4}$$

is a facet of the cut polytope.

Let the support of a circulant inequality $C(n, r)$ consist of an outer circle consisting of the nodes $i, i+1, \ldots, i-1, i$ and of inner edges of the form $(i, i+r)$. Node indices are always taken modulo $|V|$. The outer circle in a circulant inequality has odd length. Hence, we check whether by starting from a specific first node we find a cycle in the graph G of a given odd length l.[1] If we succeed, we test for all possible choices of r and k with $l = kr + 1$ and k, r even, whether all inner edges $(u, u + r)$ are present in G. In case an inner edge is missing, we proceed to the next tuple k, r. In case we can generate the support $C(l, r)$ of a circulant inequality for specific values l, r, k we either immediately test whether $\sum_{e \in C(l,r)} x_e \leq n - k - r$ is a violated inequality, or we improve its violation by switching the inequality along an appropriate cut. The outline of the resulting heuristic *separate_circulants1d* is as follows:

Algorithm: *separate_circulants1d*
$l = (|V| \text{ odd })? \, |V| : |V| - 1$
while *we can generate a cycle of length* l*, starting from node* 1 **do**
  for *all even divisors* r *of* $l - 1$ **do**
    if $k = \frac{l-1}{r}$ *even* **then**
      if *circulant edges* $(i, i + r)$ *are present for all* i *on* C **then**
        if *switching* **then**
          call *good_switching*
        if *corresponding (switched) circulant inequality on* $C(l, r)$ *violated* **then**
          push it into the constraint buffer
$l- = 2$;

Algorithm 13: Heuristic separation of the circulant inequalities in r-neighbour graphs.

In practical computations, calling the circulant inequality separation routine does not pay off. If we don't do the switching step, we usually find violated circulant inequalities only in the very first iterations. Those found inequalities only marginally improve the upper bound. In case we do the switching, we succeed in finding some violated inequalities also later in the optimization process. However, these inequalities do not improve the upper bound significantively. Additionally, the needed overall cpu time is higher than if we don't call the circulant inequality separation routine.

[1] Restricting ourselves to cycles in which a specific node is contained might seem restrictive. However, it turns out that the circulant inequalities are not strong for our class of problems, so we decided that implementing a more general routine would not pay off.

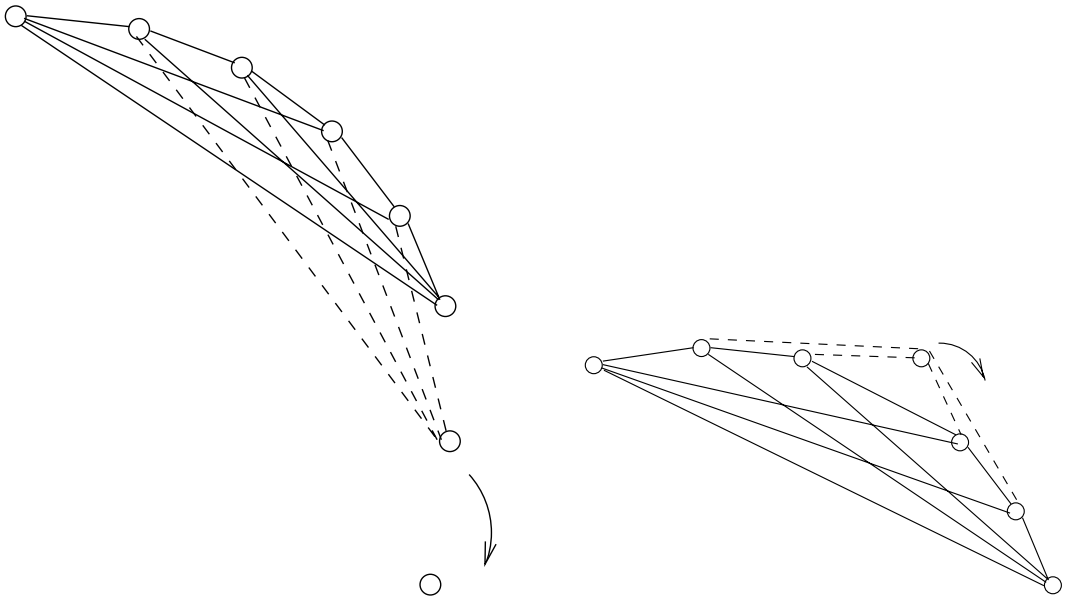
We conclude that the circulant facets are not very strong for our class of instances. Hence, for calculating ground states of one-dimensional Ising chain instances we don't separate the circulant inequalities.

Because of the structure of the input graph in the one-dimensional Ising chain, we expect 4-neighbour graphs introduced in Section 2.4 to be present. We have implemented a heuristic separation of the 4-neighbour facets. We briefly summarize how it works. We build up a 4-neighbour graph $H^4 = (V^4, E^4)$ as follows. We add node after node to the graph and label the nodes in increasing order. We always maintain a list of the last four nodes $u_1, \ldots, u_4$ that we have inserted in V^4. We inserted u_4 last. For all neighbours u of u_4 we check whether u is adjacent to u_1, u_2, u_3. If this is true, u is a candidate for insertion into V^4 after u_4. We determine the contribution to the left hand side of the 4-neighbour inequality when u is inserted in V^4. This is the amount $\sum_{i=1}^{4} x^\star_{ui}$. We choose a random number that is smaller than the number of candidates and a small natural number max that we fixed before. We insert the t-th best candidate where we rank the candidates with regard to their contribution to the left hand side of the inequality. In case we end up with a 4-neighbour graph we possibly have to correct the number of nodes $n = |V^4|$ such that $n = 3r \pm 1$, $r \geq 4$ even, by adding or deleting nodes from H^4 whichever is possible.

In practice, we do find 4-neighbour graphs. However, the same phenomenon as in the separation of the circulant inequalities occurs. The found 4-neighbour facets are usually not violated. In practical computations for one-dimensional Ising chain instances we hence do not separate the 4-neighbour facets.

Separating the parachute inequality is an $\mathcal{NP}$-hard problem. We want to heuristically separate parachute inequalities in a graph G that contains an r-neighbour graph as a subgraph. For an r-neighbour graph with $r \geq 5$, G contains the support of parachute inequalities as shown in Figure 3.4(a). The parachute consists of nodes $i, i+1, i+2 \ldots, i+s$ that are neighbours in the cycle. In case $r \geq 6$, we also have parachute inequalities where the jumper node is $i+l$ and the parachute is $i, \ldots, i+l-1, i+l+1, \ldots, i+s+1$, see Figure 3.4(b).

We do an enumeration heuristic for separating parachute inequalities in an r-neighbour graph. We only consider parachutes of the form displayed in Figures 3.4. s is the size of the parachute. For each node i, we start with $s = 6$. Starting at a node i in the chain, we check whether i together with the subsequent nodes $i+1, \ldots, i+s-1$ on the cycle form a parachute. If an edge is missing, we proceed to the subsequent node and start the procedure again. Otherwise, $i, \ldots i+s-1$ forms a parachute and we search for inequalities of the type shown in Figure 3.4(a). We proceed along the cycle in clock-wise and anti-clockwise directions and test whether the nodes $i+s-1+k$ or $i-k'$ with $k, k' > 0$ can serve as a jumper, i.e., whether all necessary jumper line edges are present in G. In case we encounter a missing jumper line, we stop increasing k or k', respectively. Subsequently we check whether we can gen-



(a) Parachute inequalities contained in an r-neighbour graph with $r \geq 5$

(b) Parachute inequalities contained in an r-neighbour graph with $r \geq 5$

Figure 3.4: Example for parachute inequalities present in an r-neighbour graph. Dashed edges have coefficient -1, solid lines coefficient 1 in $\mathrm{Par}_7 x \leq 0$. In the heuristic separation, we search for 'jumpers' in the direction of the arrows.

erate violated parachute inequalities of the form displayed in Figure 3.4(b). Nodes $i, i+1, \ldots, i+l-1, i+l+1, i+s$ form the parachute, and the jumper is node $i+l$. In case we find a violated parachute inequality, we can improve its violation by switching it along an appropriate cut. Violated inequalities are added to the linear program. In case we have found the support of at least one parachute inequality of size s, we increase the size from s to $s+2$ and restart searching for parachutes of the increased size, starting again from node i. In case we don't find the support of a parachute inequality of a given size, we proceed to the next node in the chain.

In Table 3.2 we show the number of linear programs solved and the needed CPU time for five randomly chosen instances of size $L = 100$ and $\sigma = 3.0$. We show the numbers for branch-cut&price without separating the parachute inequalities (denoted as (1)), for branch-cut&price with the parachutes separated but without switching them along an appropriate cut (denoted as (2)) and with switching them along an appropriate cut (denoted as (3)). In practice it is fastest to do branch-cut&price with separating the parachute inequalities with option (2), i.e., skipping the switching routine. It turns out that separating the parachute inequalities improves the upper bound resulting in a smaller running time. However, the speedup is not very strong.

seed	# lps(1)	# lps(2)	lps(3)	CPU(1)	CPU(2)	CPU(3)
1000	2264	2098	2143	307.50	294.94	404.92
1001	2392	2647	2536	398.11	483.95	444.28
1002	2191	2183	2229	361.82	361.54	375.67
1003	3676	3682	3723	894.38	880.50	880.98
1004	2108	2017	2011	290.75	275.34	275.70

Table 3.2: Performance of branch-cut&price without separating parachutes (denoted as (1)), for branch-cut&price with the parachutes separated but without switching (denoted as (2)) and with switching them along an appropriate cut (denoted as (3)).

In the next algorithm we summarize the separation routine for the one-dimensional Ising chain model.

Algorithm: *isingchain_separation*
separateTriangles
separateForest
separateOddCyclesRand
separateParachutes

Algorithm 14: Separation routine for the one-dimensional Ising chain.

In [63] the authors report that parallel tempering is less efficient in finding the ground state for bigger σ. They studied system sizes $L \leq 256$ for $\sigma \leq 1.0$ and $L \leq 64$ for $\sigma \geq 1.5$. In this case, parallel tempering needs longer to relax an inconvenient configuration, [60]. Instead, with respect to branch–and–cut we expect pricing to be more effective for bigger σ. For small σ we have to work on the fully connected graph and cannot exploit sparse graph techniques as explained above. We have generated a small sampling of ten instances per system size L and smaller values of σ, $\sigma = \{1.0, 2.0\}$. We report the number of the linear and sub problems and the needed cpu time in seconds. The instances are run on a 440 MHz Sun Ultra Sparc. In Table 3.4 the results are for $\sigma = 1.0$, in Table 3.3 for $\sigma = 2.0$. The treatable system sizes are considerably smaller for smaller σ than the sizes reported in Table 3.1.

L	# lps	# subs	cpu(s)
20	187 ± 31	11 ± 8	0.7 ± 0.3
40	748 ± 52	18 ± 9	13 ± 3
60	2696 ± 222	298 ± 71	300 ± 40
80	6480 ± 440	767 ± 60	2543 ± 268

Table 3.3: Average running times with branch-cut&price for $\sigma = 2.0$.

L	# lps	# subs	cpu
20	246 ± 55	18 ± 12	1.1 ± 0.5
40	5252 ± 363	468 ± 49	396 ± 32
60	23213 ± 2820	1278 ± 290	10621 ± 1280

Table 3.4: Average running times with branch-cut&price for $\sigma = 1.0$.

As a conclusion we notice that for big enough σ the branch-cut&price algorithm performs well for instances defined on the one-dimensional Ising chain making it possible to study significantively larger sizes than with the branch–and–cut algorithm.

Chapter 4

Appendix: Reduced Cost within Branch–And–Cut

The *reduced cost* of a variable plays an important role within the branch–and–cut algorithm. It is possible to fix some variables to the value they have attained by *redued cost fixing*. In the following we explain the reduced cost and the fixing. Let us consider a primal linear program of the form

$$\begin{array}{rrl} & \max & c^t x \\ (P) & Ax & = b \\ & x & \geq 0 \end{array}$$

Suppose that $\text{rank}(A) = m$, which means that the redundant inequalities are removed from the lp. Let the constraint matrix A be $A = (A_1, \ldots, A_m)$, where a_i denotes the i-th column of A. As rank $= m$, there exists an $m \times m$ nonsingular sub matrix $A_B = (A_{B_1}, \ldots, A_{B_m})$. Let $B = \{B_1, \ldots, B_m\}$ and $N = \{1, \ldots, n\} \setminus B$. By possibly reordering of the columns of A we can rewrite $A = (A_B, A_N)$. We call the nonsingular matrix A_B a *basis* of A.

We rewrite the equality constraints $Ax = b$ in the form $A_B x_B + A_N x_N = b$, where $x = (x_B, x_N)$. Then a possible solution to $Ax = b$ is given by $(x_B, x_N) = (A_B^{-1} b, 0)$. We call a solution of this form a *basic solution* of $Ax = b$. x_B are the *basic variables*, x_N the *nonbasic variables*. If $A_B^{-1} b \geq 0$, then (x_B, x_N) is called a *primal feasible basis*.

Given a basis A_B, it is convenient to write problem (P) in the form

$$\begin{array}{rrl} & \max & c_B A_B^{-1} b + \max(c_N - c_B A_B^{-1} A_N) x_N \\ (P_B) & & x_B + A_B^{-1} A_N x_N = A_B^{-1} b \\ (x_B, x_N) & & \geq 0 \end{array}$$

With the notation $\overline{A_N} = A_B^{-1}b, \overline{b} = A_B^{-1}b$ and $\overline{c_N} = c_N - c_B A_B^{-1} A_N$ we write problem (P_B) as

$$\begin{array}{rrll} & & \max & c_B\overline{b} + \max \overline{c_N} x_N \\ (\overline{P_B}) & x_B + \overline{A_N} x_N & & = \overline{b} \\ & (x_B, x_N) & & \geq 0 \end{array}$$

We call

$$\overline{c_N} = c_N - c_B \overline{A_N} \tag{4.1}$$

the *reduced cost* of the nonbasic variables. We see from the formula above that a primal feasible solution is optimum if $\overline{c_N} \leq 0$. So if we have for a specific basis matrix A_B a primal feasible solution (x_B, x_N) at hand with the reduced cost of the nonbasic variables being nonpositive, then the feasible solution is optimum. We also see from the formulation $(\overline{P_B})$ above that the reduced cost of a variable x_i gives the rate of change of the objective function if x_i changes its value from zero to a nonzero value.

Let $x^\star$ be an solution vector optimized over the current relaxation inside the branch–and–cut framework. Let $z^\star = c^{\mathrm{T}} x^\star$ be the value of the upper bound and z^p be the value of the best known cut. Let for a nonbasic variable $x_e^\star = 0$ (the case $x_e^\star = 1$ is analogous). In case $x^\star$ was set to one instead of zero, the solution value would change to $z^\star + \overline{c}_e$, where $\overline{c}_e$ is the reduced cost of $x_e^\star$. Therefore, we can *fix* the variable $x_e^\star$ to its value zero if $z^\star + \overline{c}_e \leq z^p$.

Part II

Spin-Glass Physics

Chapter 5

Introduction to Spin-Glass Physics

In this chapter we give a short introduction to the physics of spin glasses. We restrict ourselves to some important facts relevant for understanding the subsequent chapters of this thesis. Furthermore, we only go as much into detail as is needed for this work. For detailed introductions into spin-glass physics nice surveys and books exist, e.g., [82], [122], [19], [93] and [34].

5.1 'Real' Spin Glasses

Historically, the first types of spin glasses that have been studied consist of noble magnetic host metals in which magnetic transition metal impurities occupy sites randomly. For the hosts, for example copper (Cu) or gold (Au) can be used, and as impurities e.g., manganese (Mn), iron (Fe) or gadolinium (Gd). 'Classical' spin glasses are the alloys $Cu_{1-x}Mn_x$ or $Au_{1-x}Fe_x$, where x specifies the concentration of the impurities. Spin glasses usually have $x < 0.1$. Also many other combinations of elements constituing a spin glass can be found in nature and produced in the laboratory.

As a first picture for understanding what happens in the 'classical' spin glass we note that the spins (magnetic dipoles) of the impurities produce a magnetic polarization of the host metal conduction electrons that is ferromagnetic at some distances and antiferromagnetic at others. Other impurity spins then feel the local magnetic field that is produced by the polarized conduction electrons and try aligning themselves accordingly. As the impurities are randomly scattered in the host, some interactions are ferromagnetic and some are antiferromagnetic. To be more specific, in the classical spin glasses the interaction $J(r)$ between two impurities of distance r is of RKKY-type (named after Ruderman, Kittel, Kasuya, and Yosida) according to

$$J(r) \propto \frac{\cos(Br)}{Cr^3}, \tag{5.1}$$

with constants B, C. Hence, the coupling strengths fall off with $\frac{1}{r^3}$ and oscillate between ferro- and antiferromagnetic nature.

Research on spin glasses started developing rapidly in the 1970s when surprising results were found in spin-glass experiments. E.g., the *susceptibility* of CuMn and AuFe, when brought into a small oscillating magnetic field, shows a sharp cusp at a specific temperature. The susceptibility χ is defined as

$$\chi = \frac{\delta M}{\delta h}. \tag{5.2}$$

h is the external magnetic field and M the magnetisation induced by it. In Figure 5.1 we show results from [23], where $Au_{1-x}Fe_x$ with $1 \leq x \leq 0.08$ is brought in a magnetic field of 5 Gauss strength and frequency $50 \leq \omega \leq 155$ Hz. In each curve a sharp cusp is visible that increases in magnitude and in temperature for increasing concentration of the impurities. Especially the cusp and the decrease of the susceptibility when lowering the temperature are surprising and need to be understood.

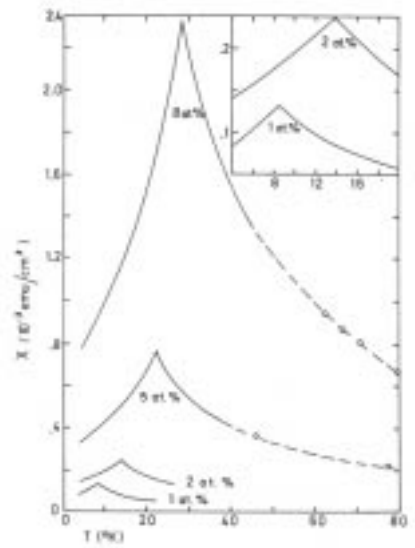


Figure 5.1: Susceptibility of AuFe for $0.01 \leq x \leq 0.08$. From Canella and Mydosh [23].

For the specific heat instead, no surprising behavior has been found at small temperatures. (The specific heat is the amount of energy per unit mass that is required for raising the temperature by one degree.) A diverging specific heat would mean that the energy fluctuations in the system diverge which is not the case in spin glasses.

The high temperature behavior of a spin glass is paramagnetic. The spins rotate randomly and independently from each other. The physically interesting behavior

takes place at low temperatures T. When T is decreased, some of the independent spins combine into correlated clusters that can rotate as a whole. A well defined temperature T_f exists where the spin-glass phase sets in. As we approach $T \to T_f$, the temperature disorder is removed and the spins start interacting with each other over a long range. However, neutron-scattering experiments reveal that no structural long-range order of the spins as in ferro- or antiferromagnets is present. Whereas the average magnetisation vanishes, a local spontaneous magnetisation exists. This nonvanishing local magnetisation causes the decrease of the susceptibility at low temperatures. At $T = 0$ the system is in its *ground state*, i.e., the energy of the system is minimum and the spins are rigidly *frozen* and *disordered*. Evidence for the freezing is gained by using the *Mössbauer effect*. This effect consists of recoil-less absorption and emission of gamma rays from some 'Mössbauer nuclei'. For example, ^{57}Fe is such a nucleus. The local magnetic field induced by the electron spins splits the nuclear spin into ground state and excited states. Gamma rays are emitted when a transition from an excited state to a ground state takes place. The decay processes have an intrincic lifetime of $\sim 10^{-7}$s. In case the so-called hyperfine spectrum of the Mössbauer nucleus can be measured, the electron spins are frozen on a time-scale longer than this.

5.2 The Ising Model for Spin Glasses

Two basic ingredients seem to be necessary for constituting a spin glass: *randomness* and *competing interactions*. There either is randomness in the position of the spins (site randomness) or in the couplings (bond randomness). Competing interactions means that no spin configuration is uniquely favoured by all interactions, which is called *frustration*.

In 1975, Edwards and Anderson [31] proposed the prominent and still widely used model for spin glasses, the *Edwards-Anderson (EA) model*. A spin is located on each site of a hypercubic lattice in d dimensions. Interactions between nearest neighbor spins are chosen according to a Gaussian distribution

$$P(J_{ij}) = \frac{1}{\sqrt{2\pi\delta}} \exp\left(\frac{-(J_{ij}-\mu)^2}{2\delta^2}\right), \tag{5.3}$$

where δ is its variance and μ its mean. (Instead of a Gaussian distribution, often also $\pm J$ couplings are used with 50% negative interactions.) If a coupling between two spins is chosen once, it keeps its value. Hence, we call the disorder *quenched*. (In contrast, a system in which the J_{ij}'s fluctuate on a similar time scale as the spins would be called *annealed*.) In the following we will be concerned with *Ising spins* which means that the spins are represented by vectors that either point 'up' or 'down'. We study the standard Hamiltonian

$$H = -\sum_{i,j} J_{ij} S_i S_j - h \sum_i S_i, \tag{5.4}$$

where h is the strength of an external magnetic field and the variable $S_i \in \{\pm 1\}$ represents spin i. A positive coupling is ferromagnetic, a negative coupling antiferromagnetic. This model inherits the necessary ingredients randomness and competition. An example for a frustrated Ising spin system can be seen in Figure 5.2. Four spins sit in the corners of a square. One interaction is negative, three are positive. No matter how the spins align, one bond cannot be satisfied. We say the square in Figure 5.2 is *frustrated*.

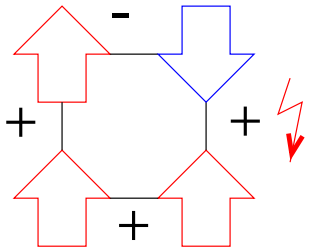


Figure 5.2: Example for a frustrated square.

It has long been a question of debate at what temperature T_f the transition to the spin glass phase takes place in the EA-model. Now it is widely believed that there is no ordering for any $T > 0$ in two dimensions for the Gaussian model, and the transition takes place at $T_f = 0$. For the $\pm J$ model, the issue is less clear. In three-dimensional spin glasses most people believe that $T_f > 0$, which is supported by several numerical simulations. Estimates for $\frac{T_f}{J}$ range from 0.88 to 1.2 for the $\pm J$ model, and 0.88 to 1.02 for the Gaussian case, see e.g., p. 18 from [66].

Until now, the nature of the spin glass phase is not fully understood. In order to be able to present what possibly is its nature, we first have to undertake a detour to mean-field models in which each spin is coupled with each other regardless of the distance. Those models can be treated theoretically, and we will study them in the next section. Finally, we come back to short-range models in Section 5.2.2.

5.2.1 Mean-Field Solution

A big step in understanding spin glasses was undertaken by Sherrington and Kirckpatrick. In 1975 they proposed [111] that the correct mean-field model for spin glasses consists of a spin system in which all spins are connected with each other. The couplings are chosen from a Gaussian distribution regardless of the distance of the spins. We call this model the *SK-model*. This model seems quite unphysical, and a short-range model would be more realistic for spin glasses occurring in nature.

However, by studying the long-range model we gain useful insights into the nature of the spin-glass state. Usually, after a mean-field theory has been established, necessary corrections due to short-range interactions, additional correlations etc. can be added later for treating 'real' systems.

In random systems two kinds of averages have to be performed: a thermal average $\langle\cdot\rangle_T$ that has to be carried out for each sample. Additionally, we have to average over the disorder $[\cdot]$ which is a 'configurational average'.

In order to understand the physics of the system, we want to determine its average free energy $[\langle F\rangle_T]$. To this end, we study the partition function defined as

$$Z = \sum_i \exp \frac{-H_i}{k_B T}, \tag{5.5}$$

where k_B is the Boltzmann constant. The free energy F is basically determined by the logarithm of the *partition function* Z,

$$F = -k_B T \ln Z. \tag{5.6}$$

For determining the average $[\langle F\rangle_T]$, we would have to average $\ln Z$ over the distribution of the couplings which is difficult. Instead, Sherrington and Kirckpatrick used the following *replica ansatz*. They made use of the equality

$$\ln Z = \lim_{n\to 0}\left[\frac{1}{n}(Z^n - 1)\right] \tag{5.7}$$

as follows: Instead of studying the logarithm of the partition function, they studied the n-th power of Z. The average $[\langle Z^n\rangle_T]$ *can* be carried out for integer n. The picture behind this ansatz is to view the n-th power of the partition function as representing n (identical) replicas of the original system. After having performed the averages, the (unmathematical) limit $n \to 0$ is calculated.

Sherrington and Kirckpatrick established a magnetic phase diagram that can be seen in Figure 5.3. Different possibilities exist for traversing through it. When lowering the temperature, we can either pass from the paramagnetic phase to the spin-glass phase or start from the paramagnetic phase, cross the ferromagnetic phase and pass to the spin-glass phase. We call this solution the *SK-solution* or the *replica symmetric solution*, as the replicas are all treated equal.

The resulting expressions for the susceptibility show a cusp at a specific temperature, consistent with the experiment. However, Sherrington and Kirckpatrick already recognized that in their solution the entropy of the system becomes negative at $T = 0$ which is unphysical. Moreover, de Almeida and Thouless showed 1978 [4] that the SK solution is *unstable* at low temperatures, both in the spin glass and in the ferromagnetic phase. They determined a stability line (the *AT-line*) in the $h - k_B T$

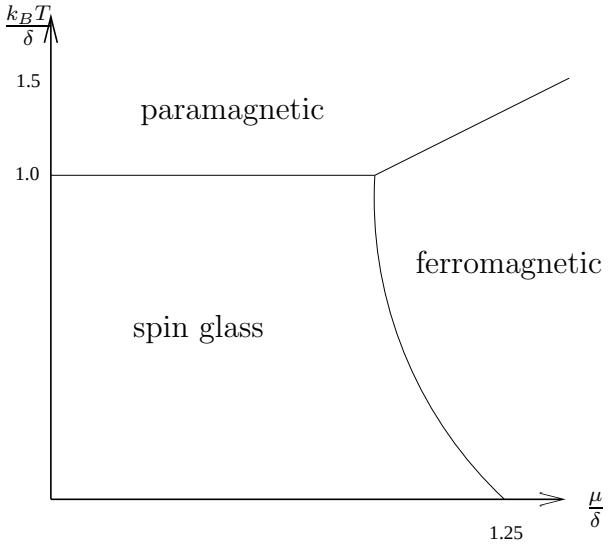


Figure 5.3: Magnetic phase diagram obtained in the replica symmetric ansatz for the SK-model.

plane. In Figure 5.4 we display the AT line together with the regions of stability and instability.

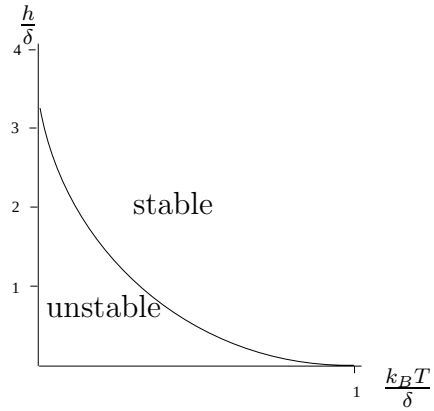


Figure 5.4: The AT line below which the SK-solution is unstable in the $h - k_BT$ plane.

It took until 1979 that Giorgio Parisi presented a solution that is believed to be correct for the SK-model. He also applied the replica ansatz. However, his key idea is not to consider the replicas of the system indistinguishable but to break the symmetry of the replicas in a very specific way. The resulting solution is at least marginally stable. We call the Parisi ansatz the *replica symmetry breaking (RSB)* solution. The physical meaning of RSB is the following. At low temperatures the energy landscape is complicated. There exist many states resembling each other in all possible degrees, which means they can also be arbitrarily different from each

other. For comparing two different configurations $\{S_i^\alpha\}, \{S_i^\beta\}$, we consider the *spin overlap* q that is defined as

$$q = \frac{1}{N} \sum_i S_i^\alpha S_i^\beta, \tag{5.8}$$

where N is the number of spins in the system. If the two configurations are equal we obtain $q = 1$, whereas $q \approx 0$ if they only coincide in half of the spins. Because of the Gaussian distribution of the couplings, the ground state is unique. However, by also taking into consideration low lying excited states, the distribution $P([q])$ of the overlap q contains a large peak and a long tail down to $q = 0$. The peak reflects the overlap with the states in the same energy valley, whereas the long tail comes from the overlap with the configurations in different valleys.

(In real spin glasses, we measure an entity q_t analogous to q that considers the spin fluctuations over the time t,

$$q_t = [\langle S_i(t_0) S_i(t_0 + t) \rangle_T], \tag{5.9}$$

where t_0 denotes a specific time. For $T = 0$ the spins will remain frozen for all t yielding $q = 1$, whereas $q = 0$ for $T \to T_f$. The order parameter q_t, in the limits $\lim_{t\to\infty}$ and $\lim_{N\to\infty}$, was introduced by Edwards and Anderson.)

Whereas mathematicians might dislike the procedure of taking a limit $n \to 0$ with n being a natural number, the RSB solution is believed to be the correct mean-field solution of the SK-model. Meanwhile, the replica ansatz has been successfully applied to other problems in statistical mechanics and even optimization problems. (As an example for the latter, see [49]). Recently, Guerra [44] presented lower bounds on the Parisi solution avoiding replicas. Talagrand [116] claims to have proven the full Parisi solution.

5.2.2 Spin Glass State for Short-Range Models

Let us now come back to the more 'realistic' EA model in d dimensions in which only nearest-neighbor interactions are considered.

One of the major challenges in spin-glass research consists in settling the question what theory correctly describes the nature of the spin-glass phase in short-range models. Different models have been proposed in the literature.

Motivated by studies of McMillan [85] and Bray and Moore [21], Fisher and Huse proposed the *droplet scaling model (DS)* [36] which is a phenomenological scaling approach for understanding the spin-glass phase. The assumption is made that the ground state strongly influences the whole spin-glass phase. A 'droplet' is the lowest energy excitation of length scale L around a particular spin site i, see Figure 5.5 for

a schematic picture. It is assumed that droplet excitations are the dominant low-lying excitations in the spin-glass phase, and that flipping a droplet of length scale L needs an energy of order L^θ. Furthermore, the droplets have a surface of fractal dimension, d_s, smaller than the space dimension d. Hence, droplet excitations are not space-filling. Several numerical studies have convincingly shown that in two dimensions $\theta \sim -0.28$ for Gaussian distributed couplings. Thus, as θ is negative in two dimensions, the low-lying excitations can be created on longer and longer length scales destroying the spin-glass phase for any $T > 0$. This result is consistent with what we know about spin glasses. Numerical studies, in which the excitations are generated e.g., by changing the boundary conditions from periodic to antiperiodic or vice versa, show that $\theta \sim 0.2$ in three dimensions, e.g., [47], so spin-glass ordering is present. In this model, the distribution of the overlap q is a delta function as only one state (and its global flip) exists. (We note that for finite systems, the distribution of overlaps always has finite width, and the delta function is only found for $N \to \infty$.)

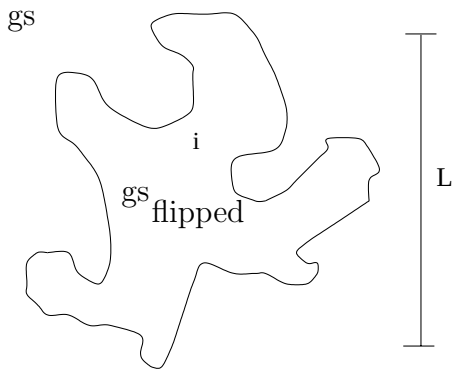


Figure 5.5: Schematic picture of a droplet. The spins that are part of the droplet are flipped compared to the ground state.

It is not clear whether the droplet ansatz is correct for the EA-model. Instead, it is possible that Parisis RSB-solution is not only an artefact of the infinite-range couplings but remains valid for short-range models. We review the phenomenological picture of RSB. In contrast to DS, in the RSB picture the energy of flipping a cluster of length scale L remains of order $\mathcal{O}(1)$, even if L grows infinitely big. The surface d_s of the excitations is space-filling, i.e. $d_s = d$.

Recently, Krzkala and Martin [72] and Palassini and Young [100] have argued that an intermediate picture between droplet scaling and RSB is correct. In this *TNT picture* ('trivial-nontrivial'), the energy of the low-lying excitations does not increase with the size of the excitation, as proposed by the RSB-solution. However, the surface of the excitations are not space-filling as suggested in the droplet scenario. In addition to the exponent θ that measures the growth of the energy of an excited cluster when the boundary conditions are changed, we need a new exponent θ' that might be different from θ. In this scenario, the energy of clusters that are excited within

the system scale with $L^{\theta'}$, where the boundaries are fixed. Obviously, in the droplet picture it is $\theta = \theta'$. In [72] and [100] an exponent θ' compatible with zero was found.

Different studies have found consistency with different models, e.g., [47], [108] and [80]. However, until now none of the projects could rule out the possibility of a crossover to a different model at larger sizes than those studied. Hence, the nature of the spin-glass state remains an interesting field for further research.

5.2.3 How 'Realistic' is the Ising Model for Spin Glasses?

We might ask under which circumstances the simplificiation of studying one-dimensional Ising spins is a 'good' approximation. A possible answer is that there exist spin-glass systems that are of 'ideal' Ising nature, e.g., $Rb_2Cu_{1-x}Co_xF_4$. In this lattice a fraction of the Rb_2CuF_4 compounds is replaced by Rb_2CoF_4. These spin glasses show a strong magnetic anistropy which means that the impurity spins prefer one direction for aligning. As the spins are constrained in one dimension, they can be represented as Ising spins. (We note that the classical spin glasses $Cu_{1-x}Mn_x$ or $Au_{1-x}Fe_x$ mentioned earlier are Heisenberg systems without a strong anisotroy. Here it is more realistic to consider the spins as three-dimensional vectors.) Furthermore, the system $Rb_2Cu_{1-x}Co_xF_4$ is an 'ideal' two-dimensional spin glass. Here the spins are separated by a large distance along one axis; in the other two dimensions a simple square lattice of strongly interacting moments is found. Also 'good' three-dimensional spin glasses exist, e.g., the system $Fe_{0.5}Mn_{0.5}TiO_3$ (with $T_f \sim 22$ K). In one compound we have a random substitution of Fe and Mn spins situated in a hexagonal lattice. As a summary we note that there exist spin-glass systems that can be treated nicely as Ising spin glasses in two or three dimensions.

In the next section we argue why we need exact ground states for Ising spin glasses. Subsequently, in Chapter 6 we study the feromagnet- spin glass transition for z-regular graphs or so-called *Bethe lattice* and relate it to the performance of the branch–and–cut algorithm. The latter results were obtained in a cooperation with Matteo Palassini and Alexander K. Hartmann and are already published in [77]. Finally, in Chapter 7 we give results obtained in a cooperation with Matteo Palassini and A. Peter Young that are published in [101]. In this cooperation, we study the nature of the spin-glass state for three-dimensional Ising spin-glasses.

5.3 Why We Need Exact Ground States

Many researchers interested in Ising spin glasses use different heuristic methods for calculating ground states, including genetic algorithms and Monte Carlo simulations. Those heuristics are not *exact* meaning that it is not clear whether the generated low energy states are *true* ground states or not.

We expect that an implementation of a genetic algorithm or a Monte Carlo method is able to find an exact ground state in a 'not complicated' energy landscape, see Figure 5.6.

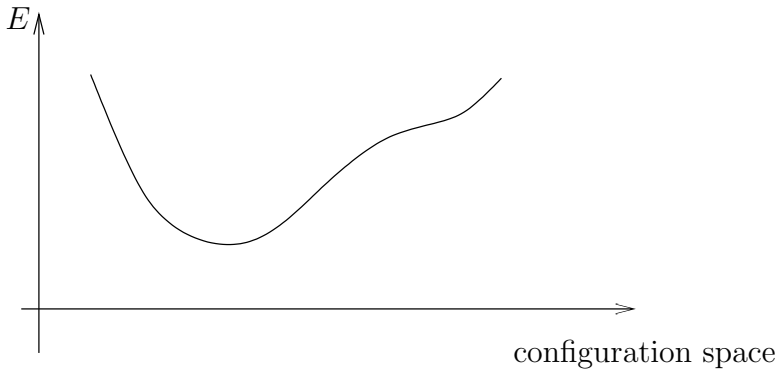


Figure 5.6: Sketch of an 'easy' energy landscape.

The problem of determining an exact ground state gets more difficult in a 'more complicated' landscape in which there are many sub optimal states with energy only slightly above the ground state that are separated from each other by high energy barriers as we have in spin glasses, see Figure 5.7.

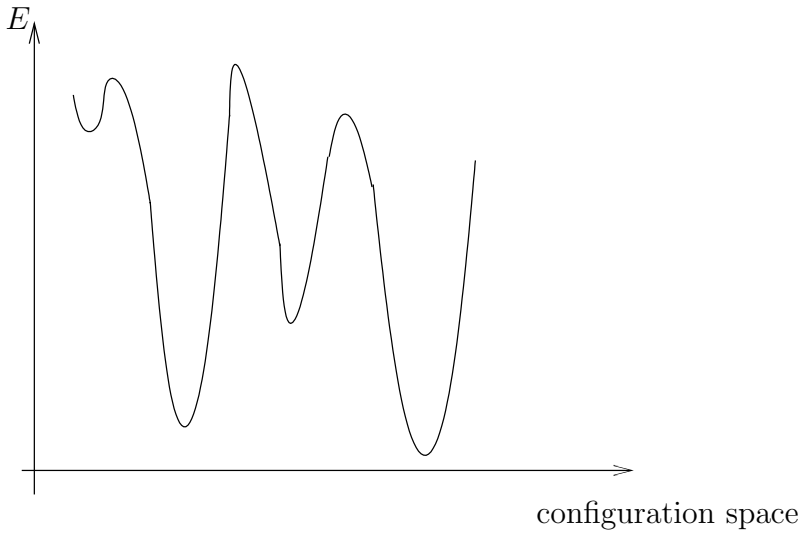


Figure 5.7: Sketch of a 'more complicated' energy landscape.

Recently, it has become popular to analyze numerically generated data on correlation functions like spin overlap, link overlap etc. We refer to Chapter 7 for the definitions and a possible application of these entities. It is clear that a heuristic algorithm introduces a bias in the correlation function data.

In the following we give a measure for the complicatedness of the energy landscape of a sample allowing us to distinguish between samples that can 'easily' or only 'hardly' be solved by an easy simulated annealing method. For Gaussian distributed spin-glass instances, different levels of 'complicatedness' exist, and we qualitatively show that the not complicated samples can be well solved by an easy simulated annealing method. However, there is a small percentage of complicated samples that cannot be solved. The complicatedness of a sample is an underlying reason for a sample to be easily or hardly solved by SA.

There is no obvious immediate measure for the complicatedness of an energy landscape. We measure it indirectly as follows. For an instance, we determine an exact ground state, apply a specific perturbation to the couplings and determine a ground state of the perturbed system. Intuitively, we call a landscape *complicated* if the spin configurations of the unperturbed and perturbed ground states are 'very' different. Otherwise the energy landscape is *uncomplicated* as the perturbation only slightly changes the ground state.

To be more specific, we apply the bulk ϵ-perturbation of Palassini and Young, [102] (see also Chapter 7). Let m be the number of couplings and $\epsilon > 0$ small. For an instance, we determine an exact ground state S_i^0 and perturb all couplings. If two spins i, j that are coupled by a nonzero coupling strength J_{ij} point parallel in the ground state, we substract the amount $\frac{\epsilon}{m}$ from J_{ij}, otherwise we add $\frac{\epsilon}{m}$ to it. By applying this perturbation to all couplings we ease a change in the ground state. It is easy to see that in the perturbed system the energy of the former ground state increases by ϵ, whereas the energy of the other states increases by an amount smaller than ϵ.

We determine a ground state S_i^ϵ of the perturbed system and compare the two ground states by measuring the absolute value of the spin overlap

$$|q|^{0,\epsilon} = \frac{1}{N}\Big|\sum_{i=1}^{N} S_i^0 S_i^\epsilon\Big|,$$

where N is the number of spins. $|q|^{0,\epsilon}$ represents the fraction of spins in which the unperturbed and perturbed ground states coincide (up to the global spin flip).

Definition 5.1. *For $0 < \alpha < 0.5$ we say a sample with $|q|^{0,\epsilon} > 1.0 - \alpha$ is* α-uncomplicated *whereas a sample with $|q|^{0,\epsilon} < \alpha$ is* α-complicated.

Our notion of complicatedness depends on the perturbation parameter ϵ and takes on different values for different perturbations. However, the results remain qualitatively the same for different perturbations.

With respect to branch–and–cut, we found in the project [101] (see Chapter 7) that computing a ground state of a complicated sample (by using the complicatedness definition above) takes considerably longer to be computed than an uncomplicated

sample. The main reason for the longer running times is that the reduced–cost–fixing is less efficient for complicated than for uncomplicated samples.

Coming back to our problem, we have implemented a straight forward simulated annealing method for calculating ground states of spin glasses. We start with a temperature T and decrease it iteratively. For each temperature, we choose both single spins and spins adjacent at the same edge and flip it whenever it decreases the system energy or according to the Boltzmann weight. We summarize the procedure in the following.

Algorithm: *simulated_annealing (SA)*
$T = 200$;
maxtimes $= 50$;
for $t = 0$; $t <$ *maxtimes*; $t{+}{+}$ **do**
  while $T \geq 0$ **do**
    for $i = 0$; $i < N$; $i{+}{+}$ **do**
      possibly flip a randomly chosen single spin
    for $e = 0$; $e < m$; $e{+}{+}$ **do**
      possibly flip spins adjacent to a randomly chosen edge
    $T{-} = 1$;

Algorithm 15: An easy simulated annealing (EA) algorithm.

In the block 'possibly flip a randomly chosen single spin' we randomly choose a spin s and flip it if this decreases the system energy; otherwise we flip s with probability $\sim \exp(\frac{-\Delta H}{T})$, where ΔH is the amount of change in the system energy when s flips. Analogously, in block 'possibly flip spins adjacent to a randomly chosen edge' we choose an edge $e = (i, j)$ randomly and flip i and j simultaneously if it decreases the system energy. Otherwise, we flip i and j according to the Boltzmann weight.

We consider 1924 two-dimensional spin-glass instances of size $L = 10^2$ with periodic boundaries. We choose Gaussian distributed couplings in order to have a unique ground state making overlap discussions useful. We first study the complicatedness of the energy landscapes. To this end, we determine an exact ground state with branch–and–cut, apply the bulk perturbation and solve the perturbed system by branch–and–cut. We determine the complicatedness of a sample by calculating the spin overlap $|q|^{0,\epsilon}$ between the unperturbed and perturbed systems. Inspired by the project [101], we choose the perturbations $\epsilon = \{\frac{\tau}{2}, \tau, 2\tau\}$ with $\tau = \sqrt{6}$. (In that work, for comparison of the perturbation's size within different spin-glass models, τ is chosen as the mean-field transition temperature $\tau = \sqrt{z}$, where z is the connectivity of a node. Continuing in this flavor, we would choose $\tau = \sqrt{4}$. However, we keep $\tau = \sqrt{6}$ and will see next that this range of perturbations is reasonable.)

For perturbation $\epsilon = \frac{\tau}{4}$ the exact ground state changes for 43% of the instances. For $\epsilon = 2\tau$ it changes for 97% of the samples. Hence, in our range of perturbations the ground state of most instances changes.

In Table 5.1 we show the distribution of the complicatedness of the energy landscape for perturbation 2τ. We group the samples according to their complicatedness. We consider intervals with $|q|^{0,\epsilon} \in [0..0.2],]0.2..0.4],]0.4..0.6],]0.6..0.8],]0.8..1.0]$. In Table 5.1 we show the numbers for $\epsilon = 2\tau$. We see from the table that 23.39% of the samples are 0.2-complicated whereas 18.60% of the samples are not-0.2-complicated.

$\|q\|^{O,2\tau} \in$	% samples
[0..0.2]	23.39
]0.2..0.4]	21.93
]0.4..0.6]	20.69
]0.6..0.8]	15.39
]0.8..1.0]	18.60

Table 5.1: Number of samples with different overlaps $|q|^{0,2\tau}$ between the unperturbed and the perturbed ground state. The perturbation is chosen as $\epsilon = 2\tau$.

As expected, for smaller perturbation values the number of not complicated samples is higher and the number of complicated samples is smaller than for $\epsilon = 2\tau$.

For the same samples, we determine low-energy states with SA and test whether there is a correlation between the quality of the approximate solution and the complicatedness of the energy landscape.The SA procedure needs roughly four minutes on average per sample on a 1400MHz Athlon computer. For $\epsilon = 0$, SA generates an exact ground state for 25.5% of the samples.

In order to quantify the quality of the heuristic solution, we determine the absolute value of the spin overlap $|q|^{\mathrm{SA}}$ between the exact solution generated by branch–and–cut and the heuristic solution generated by SA. Clearly, if the overlap $|q|^{\mathrm{SA}}$ equals 1, the heuristic succeeded in generating the exact solution. A small overlap value $|q|^{\mathrm{SA}} \approx 0$ indicates that the approximate SA-solution is poor.

For 71.72% of the samples, SA succeeds in generating a solution that has an overlap of at least 0.8 with the exact solution, for 4.37% it generates a solution with overlap less than or equal to 0.2 with the exact solution. For the numbers in Table 5.2, we group the statistics into samples having overlap $|q|^{\mathrm{SA}}$ with the exact solution in the intervals $[0..0.2],]0.2..0.4],]0.4..0.6],]0.6..0.8]$, or $]0.8..1.0]$, respectively.

Next, we study the strength of SA for different levels of complicatedness. It is interesting to ask whether there is a correlation of the complicatedness of the energy landscape $|q|^{0,\epsilon}$ with the quality of the SA solution, $|q|^{\mathrm{SA}}$, for a perturbation ϵ. We expect that for samples with complicated landscape the SA solution has a smaller overlap with the exact solution than for uncomplicated ones.

$\|q\|^{\mathrm{SA}} \in$	% samples
$[0..0.2]$	4.37
$]0.2..0.4]$	5.15
$]0.4..0.6]$	6.76
$]0.6..0.8]$	12.00
$]0.8..1.0]$	71.72

Table 5.2: Percent of samples having overlap $|q|^{\mathrm{SA}}$ of the exact with the SA solution.

Our main interest is in the samples for which SA is able to generate a state S^{SA} 'close to' the ground state. We say a state is 'close to' the ground state if it coincides with it in at least $\beta * 100\%$ spins, i.e., if

$$|q|^{\mathrm{SA}} \geq \beta,$$

where $\beta \in [0 \ldots 1]$.

For different levels of complicatedness of the energy landscape, we study the probability that simulated annealing is able to generate a state close to the ground state. We chose $\beta = 0.8$ and checked that the results remain qualitatively comparable (but less clear) for various β between 0.6 and 0.9. In Figure 5.8 we show a histogram for the probabilities that SA finds a state close to the ground state for different levels of complication of the energy landscape. We choose perturbation $\epsilon = 2\tau$. The rightmost column corresponds to the samples with a 0.2-uncomplicated energy landscape. Here the unperturbed and perturbed ground states coincide in at least 80% of the spins. In this case, the SA algorithm finds a state close to the ground state with high probability ($\sim$ 90% for 2τ). The leftmost column corresponds to the 0.2-complicated samples where the unperturbed and perturbed ground states coincide in at most 20% of the spins. Here, states close to the ground state are less likely to be generated by SA ($\sim$ 60% for 2τ). We have comparable results for the other perturbations and different α-values.

As a conclusion, we have seen that an easy simulated annealing method is more likely to generate a solution close to the ground state for samples with uncomplicated energy landscapes than for complicated ones. The complicatedness of a sample is an underlying reason for a sample to be easily or hardly solved by SA. The heuristic algorithms used in the literature are much more advanced than our SA avoiding some of the pitfalls. Many people using approximate solvers state they 'almost always' find the true ground state, and that the error due to sometimes missing the exact ground state is smaller than the statistical errors. However, the used method might introduce a bias in the sense that the generated solution for complicated samples are less likely to be exact.

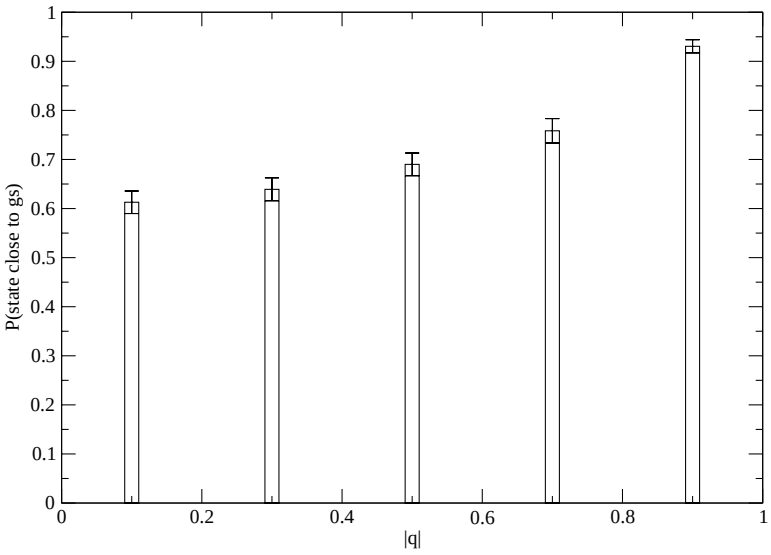


Figure 5.8: Histogram of the probability that SA finds a state close to the ground state for different levels of complication of the energy landscape. A state is close to the ground state if its spin overlap with the ground state is at least 0.8. We group the results in intervals of size 0.2 and show the results for $\epsilon = 2\tau$.

Chapter 6

Ground-State of the Bethe Lattice Spin Glass and the Performance of Branch–And–Cut

The results of this chapter stem from a cooperation with Matteo Palassini and Alexander K. Hartmann. They are already published in [77]. We study Ising spin-glass instances defined on random z-regular graphs, i.e., instances in which each spin ist connected to exactly z spins. The couplings are chosen from a Gaussian distribution with mean μ and variance one. We study the energy and the magnetisation for different values of μ and locate the spin glass/ferromagnetic transition at $\mu = 0.77 \pm 0.02$ for $z = 4$ and at $\mu = 0.56 \pm 0.02$ for $z = 6$. Furthermore, we study the performance of the current implementation of the branch–and–cut algorithm across the phase transition. We find that the running time undergoes a sharp change consistent with polynomial dependence on the system size deep in the ferromagnetic phase and a superpolynomial dependence in the spin-glass phase.

We motivate our study and introduce the model in the next section. In Section 6.2, we present the results for the ground-state energy and the magnetisation. In Section 6.3, we study the performance of the branch–and–cut algorithm.

For several decision or optimization problems easy/hard thresholds analogous to phase transitions have been observed in random instances. The most prominent are the satisfiability problem (SAT) [88, 89], vertex cover [120] (VC), and number partitioning [81]. An interesting question is how phase transitions of the system affect the performance of the solution algorithms, following the observation [24] that the 'typical', i.e., median, running time undergoes a sharp change in the vicinity of the phase transition. For example, in 3SAT and in VC, the typical running time of exact backtracking algorithms changes [26, 121] from a polynomial dependence on the input size (in the "solvable" region) to exponential dependence (in the "unsolvable" region).

We want to study the typical running time of the algorithm accros the zero-temperature spin glass/ferromagnet phase transition that occurs when varying the mean value of the Gaussian distributed couplings. To our knowledge, the performance of branch–and–cut for this model has not been investigated in detail before (see, however [113], [50] and [101]). Furthermore, there is a physical motivation for this project. Unlike in other optimization problems like 3SAT or VC, averaging over random instances is physically motivated and arises naturally in spin-glass computations. The only other study that we are aware of in which the performance of a combinatorial algorithm is studied around a 'physical' phase transition is the study of Middleton [87] for the random-field Ising model. Here the typical running time of the matching algorithm is investigated, which however is polynomially solvable. We note that we have to keep in mind that by studying the running time of *one* special implementation of *one* solution algorithm, we cannot necessarily draw conclusions about the "hardness" of the *problem* itself.

Another motivation is to study the ground-state properties of the Ising model with fixed connectivity. Recently these so-called Bethe lattices have gained a renewed interest, [83, 84, 105, 17]. We compute the ground-state energy and magnetisation which yields a useful tests for the theory.

6.1 Model and Methods

We investigate Ising spin glasses on random z-regular graphs $G = (V, E)$ with N spins that lie on the nodes V. Spins i and j might be coupled by a coupling strength J_{ij} that is drawn from a Gaussian distribution with mean μ

$$P(J) = \frac{1}{\sqrt{2\pi}} \exp[-(J-\mu)^2/2]. \tag{6.1}$$

Each spin is connected to exactly z spins. We study the standard Hamiltonian

$$\mathcal{H} = -\sum_{(i,j)\in E} J_{ij} S_i S_j. \tag{6.2}$$

This model provides a convenient realization of a Bethe lattice [83]. As in those graphs the typical size of a loop is of order $\log(N)$, small loops are rare. Therefore we have a local tree-like structure, and the mean-field approximation is exact. The Viana-Bray model [119] is related to the model studied here. In the Viana-Bray model the connectivity is a Poisson variable with finite mean. These diluted models represent a better approximation to finite-dimensional spin glasses than the infinitely-connected Sherrington-Kirkpatrick model.

It is well known that replica symmetry is broken both in the Bethe lattice and the Viana-Bray model, [119, 92, 117]. However, until recently solutions could be derived

only in some cases. Mézard and Parisi introduced [83, 84] a population dynamics algorithm based on the "cavity method" and determined a numerical solution at the level of one step of replica symmetry breaking. They explicitely give results for the Bethe lattice spin glass with $\pm J$ distribution, but not for the Gaussian distributed spin glass studied here. Previous numerical studies of this model can be found in Refs. [73, 17, 6]. For a complete discussion of the Bethe lattice spin glass, see Ref. [83] and references therein.

Determining a ground state of the Hamiltonian (6.2) is a hard problem. Heuristic algorithms recently used include simulated annealing [65], "multicanonical" simulation [15], genetic algorithms [97, 100], extremal optimization [17], a hierarchical renormalization-group based approach [53], and the cluster-exact approximation algorithm [46]. In the following we present the results we derived with the branch–and–cut algorithm. As the exact algorithm is deterministic, the running time to find the ground state is a well defined quantity.

In the publication [77], we also derived the energy and the magnetisation with the Bethe-Peierls (BP) approximation which is equivalent to the replica symmetric solution, using the population dynamics approach proposed by Mézard and Parisi in [84]. For the BP approximation we only show the results here and refer to the publication [77] for their derivation. The replica symmetric solution is wrong for our model. In [83] an algorithm is presented that solves the problem at the level of one step of replica symmetry breaking. We have not attempted to doing this as we would need significant computing time. Furthermore, in the following it will turn out that the BP approximation gives sufficiently accurate results for our purposes.

6.2 Results for Energy and Magnetisation

We study Ising spin-glass instances on z-regular graphs, with $z = 4$ and $z = 6$. A sample (or an instance) is generated by first building up a random regular graph with the method proposed in [115]. Then the couplings J_{ij} taken from distribution (6.1) are assigned to the edges.

We investigate graph sizes up to $N = 400$ for $z = 4$ and $\mu \leq 0.9$, and up to $N = 200$ for $z = 6$ and $\mu \leq 0.7$. For larger values of μ, we consider sizes up to $N = 1280$. The running times for the smallest systems are less than a second, while the longest computations take at most one day on a typical workstation. As we cannot use the heuristics speeding up the computations for the Edwards Anderson model, we cannot go to as big sizes as for regular grids.

We average the results over many samples. The largest number of samples are considered in the region of the phase transition, where the fluctuations of the magnetisation is larger. Near the transition, for sizes $N \leq 240$ ($z = 4$) and $N \leq 160$ ($z = 6$) we compute around 5000 samples for each value of μ; for $N = 400$ ($z = 4$) and $N = 200$

($z = 6$), around 500 samples for each value of μ. For sizes larger than these, we compute up to 280 samples for each μ. For the analysis of the ground-state energy and magnetization, we consider only sizes up to $N = 400$ ($z = 4$) and $N = 200$ ($z = 6$), since for larger sizes the statistical error is quite large. In the analysis of running times we include all sizes.

6.2.1 Ground-State Energy

In Figure 6.1 we show the average ground-state energy $E(\mu, N)$ divided by zN, as a function of μ for the two connectivity values $z = 4, 6$ and different sizes. The symbols correspond to the branch–and–cut data, the lines to the replica symmetric solution obtained with the population dynamics approach. For μ big enough, the system is basically ferromagnetic, and the ground-state energy depends linearly on μ, as visible in the figure. For small μ frustration becomes important and the curves bend down. As $E(0, N)$ scales as $\sqrt{z}$, not as z, the two curves diverge for small μ. The branch–and–cut data and the replica symmetric solution agree well.

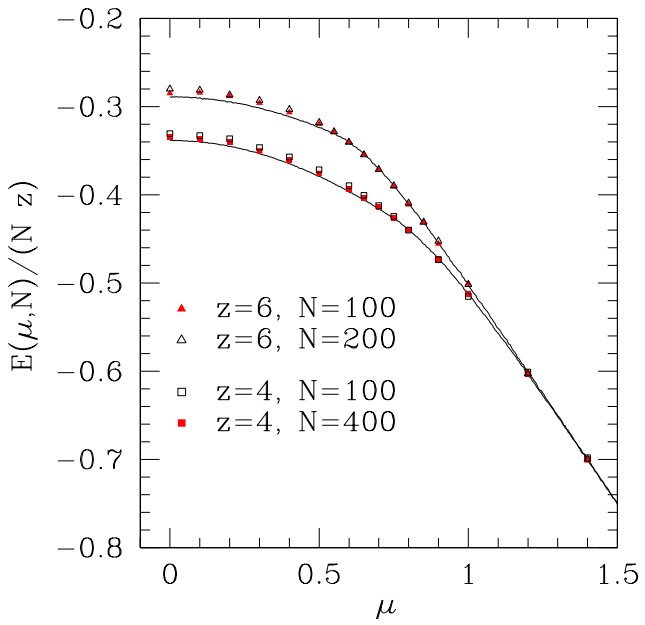


Figure 6.1: Average ground-state energy, divided by the number of spins, N, and the connectivity, z, as a function of μ. The symbols show the branch–and–cut data. The statistical errors are smaller than the symbol sizes. The lines represent the numerical solution of the BP recursion and are obtained by connecting points spaced by $\Delta\mu = 0.005$ ($\Delta\mu = 0.001$ near the transition). The statistical error is comparable to the line thickness.

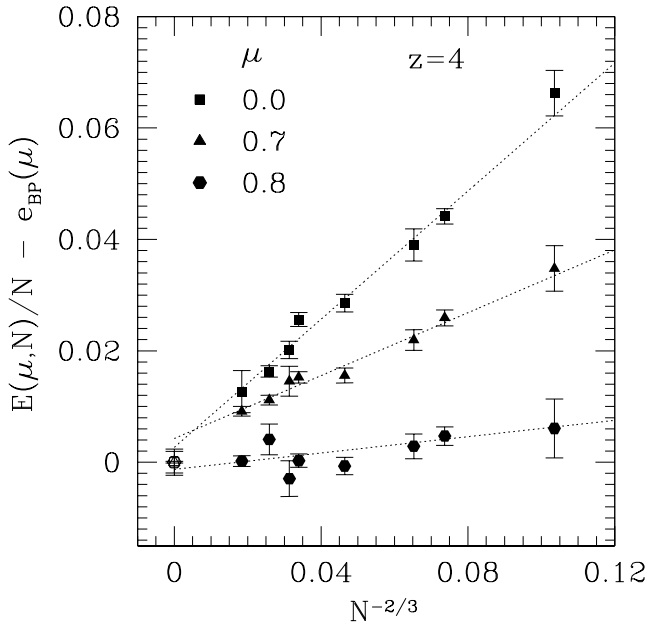


Figure 6.2: Ground-state energy as a function of the size, N, for $z = 4$ and $\mu = 0, 0.7$ and 0.8. The lines show the best fits with $E/N = e_\infty + bN^{-2/3}$. The $N = \infty$ data displays the BP solution.

For extrapolating the the ground-state energy to infinitely big systems, we fit a function $E/N = e_\infty + bN^{-2/3}$ to the data. From Figure 6.2 we see that the corrections due to finitely big systems are well described by a $N^{-2/3}$ behavior for small μ. (However, we could also reasonably fit an $N^{-\omega}$ term with ω between 0.6 and 1.) Boettcher [17] has studied the Bethe lattice with a heuristic algorithm and also found an $N^{-2/3}$ correction to fit the data well. Mézard found in Ref. [83] an exponent of $\omega = 0.767(8)$ at $T = 0.8$ for the $\pm J$ distribution and $z = 6$, which is close to 2/3. Palassini [98] found the value $\omega = 0.62 \pm 0.05$, compatible with 2/3 for the Viana-Bray model with an average of 6 neighbors. The data was generated with a heuristic.

For $\mu = 0$ we obtain $e_\infty = -1.38 \pm 0.04$ ($z = 4$) and $e_\infty = -1.72 \pm 0.02$ ($z = 6$), where the errors take into account the uncertainty on the correction exponent ω. For the replica symmetric solution we find $e_{BP} = -1.351 \pm 0.002$ ($z = 4$) and $e_{BP} = -1.737 \pm 0.002$ ($z = 6$).

It is also interesting to compare this with the ground state energy per spin found in two [45] and three dimensions [97] (which have coordination number $z = 4$ and $z = 6$, respectively) with Gaussian couplings and $\mu = 0$, which is $e_\infty = -1.31453(3)$ and $e_\infty = -1.7003(1)$ respectively.

We see in Figure 6.2 that the energy e_∞ for infinitely big system sizes, extrapolated from the branch–and–cut data, is close to the result for the replica symmetric solution e_{BP}. For sufficiently large μ the replica symmetric solution is exact, therefore the good agreement can be expected. However, the agreement is still good in the spin glass phase in which replica symmetry is broken. We conclude that the corrections due to replica symmetry breaking are quantitatively small on the energy, less than 1%.

6.2.2 Ground-State Magnetization

In this section we study the average ground-state magnetisation $m = [M]_J$, which is defined as $M = \frac{1}{N}\sum_i S_i$. $[\ldots]_J$ denotes the sample average. We show in Figures 6.3 and 6.4 the magnetisation as a function of μ for different system sizes N, both for connectivity values $z = 4$ and $z = 6$. The lines represent the $N = \infty$ result in the replica symmetric solution.

For small μ the magnetization vanishes as $1/\sqrt{N}$. For large μ, the branch–and–cut data agrees with the BP result within the error bars. The critical point at which the phase transition occurs is characterized by the vanishing of the magnetisation. From the vanishing of the BP magnetisation we get the critical μ at $\mu_c = 0.742 \pm 0.005$ ($z = 4$) and $\mu_c = 0.546 \pm 0.005$ ($z = 6$).

The critical mean μ_c can also be determined by studying the so-called Binder cumulant [18]

$$g(\mu) = \frac{1}{2}\left(3 - \frac{[M^4]}{[M^2]^2}\right), \tag{6.3}$$

where $[\cdots]$ is the "time" average when used in the context of the population dynamics approach. This cumulant can be used because in the limit $\mathcal{N} \to \infty$ it is $g(\mu) = 0$ for $\mu < \mu_c$ and $g(\mu) = 1$ for $\mu > \mu_c$.

We get from the largest population size

$$\begin{aligned} \mu_c^{BP} &= 0.743 \pm 0.005 \quad (z = 4) \\ \mu_c^{BP} &= 0.547 \pm 0.005 \quad (z = 6) \end{aligned}$$

These values agree with the above estimate from the average magnetization. We verified that the expected scaling $m_{BP} = a(\mu - \mu_c)^\beta$ for $\mu \simeq \mu_c$, is satisfied with the mean-field exponent $\beta = 1/2$ and $a \simeq 0.23$ for these values for μ_c. Klein et al. [68] solved the BP recursion in the region of μ_c with the mean random field approximation (MRF). Their results $\mu_c^{MRF} = 0.775$ $(z = 4)$ and $\mu_c^{MRF} = 0.587$ $(z = 6)$ (obtained after rescaling their value by an appropriate normalization factor $\sqrt{z}$) are slightly larger than our result μ_c^{BP}.

By replacing the time average by the sample average in Eq.(6.3), we can estimate μ_c from the finite-size branch–and–cut data. We will see later from the scaling of the Binder cumulant that the curves for $g(\mu, N)$ as a function of μ for various N must cross at the critical point $\mu = \mu_c$.

We plot in Figure 6.5 the Binder cumulant in the region of the intersection point. We obtain

$$\begin{aligned} \mu_c &= 0.77 \pm 0.02 \quad (z = 4) \\ \mu_c &= 0.56 \pm 0.02 \quad (z = 6). \end{aligned}$$

The branch–and–cut value agrees with μ_c^{BP} within the error bars. With the same reasoning as above we conclude that also for the magnetization replica symmetry breaking corrections are small. They cause a shift of μ_c of less than $3-4\%$. Effects due to replica symmetry breaking are expected to increase with z. In the Sherrington-Kirkpatrick model (which is the $z \to \infty$ limit of the present model), corrections shift μ_c from 1.25 to 1. Our numerical estimate of μ_c is slightly *larger* than μ_c^{BP} which could be a statistical fluctuation or a finite size effect.

For $\mu \simeq \mu_c$, the Binder cumulant is expected to satisfy a finite-size scaling relation [106] as follows:

$$g(\mu, N) = \tilde{g}(N^{1/(d_u \nu)}(\mu - \mu_c)) \tag{6.4}$$

where d_u is the upper critical dimension, i.e., the dimension above which the mean field solution is exact. It is $d_u = 6$ for the Ising spin glass. So $g(\mu = \mu_c, N) = \tilde{g}(0)$,

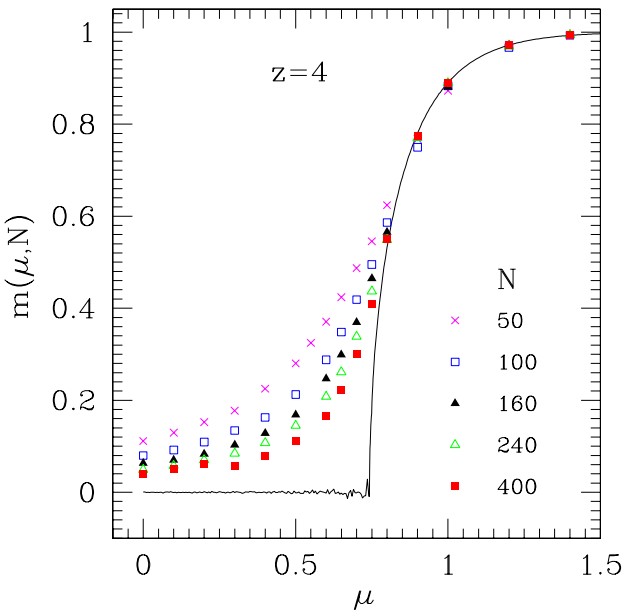


Figure 6.3: Average ground-state magnetization m as a function of the mean μ of the Gaussian distribution for $z = 4$. The symbols represent the branch–and–cut data for various system sizes N. The statistical errors, not shown, are smaller than the symbol sizes. The line represents the numerical BP recursion.

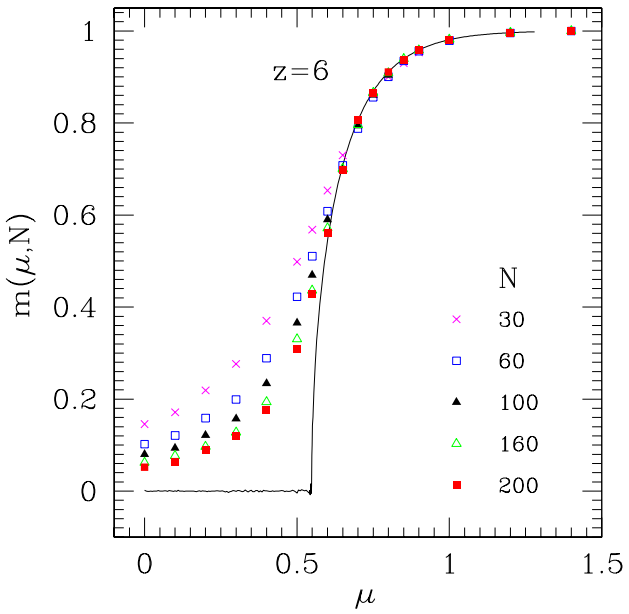


Figure 6.4: Same as Figure 6.3 but for $z = 6$.

and the curves for $g(\mu, N)$ in Figure 6.5 as a function of μ for various N have to cross at μ_c.

(The scaling behavior can be understood as follows. In the EA model, near the phase transition the only existing length scales are the linear size L and the correlation length ξ. The number of spins N is $N = L^d$ and the scaling of the correlation length is $\xi \propto (\mu - \mu_c)^{-\nu}$. The Binder cumulant g is dimension less, so g is a function $g(\frac{L}{\xi}) = \tilde{g}(N^{1/(d_c\nu)}(\mu - \mu_c))$. We know that the mean field solution is exact in dimensions equal to or bigger than the critical dimension d_c. In our model the mean field solution is exact. The scaling law follows by taking our model as if it was a regular graph in $d = d_c$ dimensions.)

In Figure 6.6 we show the scaling of the Binder cumulant for the branch–and–cut data. The plot shows $g(\mu, N)$ as a function of $N^{1/(d_u\nu)}(\mu - \mu_c)$ with the mean-field exponent $\nu = 1/2$. We see that the data collapses into a single curve near $(\mu - \mu_c) = 0$. We observe that finite-size scaling is well satisfied in our range of sizes.

We show in Figure 6.7 scaling plots for the average magnetization $m(\mu, N) = [M]_J$. Its scaling is

$$m(\mu, N) = N^{-\beta/(d_u\nu)} \tilde{m}(N^{1/(d_u\nu)}(\mu - \mu_c)) \,, \tag{6.5}$$

with the mean field exponent $\beta = 1/2$. The data show a good data collapse for $\mu \leq \mu_c$.

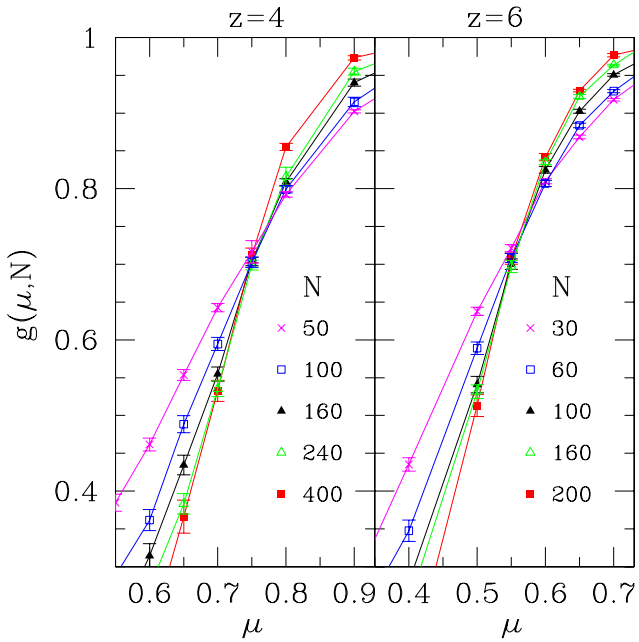


Figure 6.5: Binder cumulant as a function of the mean of the disorder distribution, μ for various sizes. On the left panel we show the connectivity $z = 4$, the right shows the results for $z = 6$.

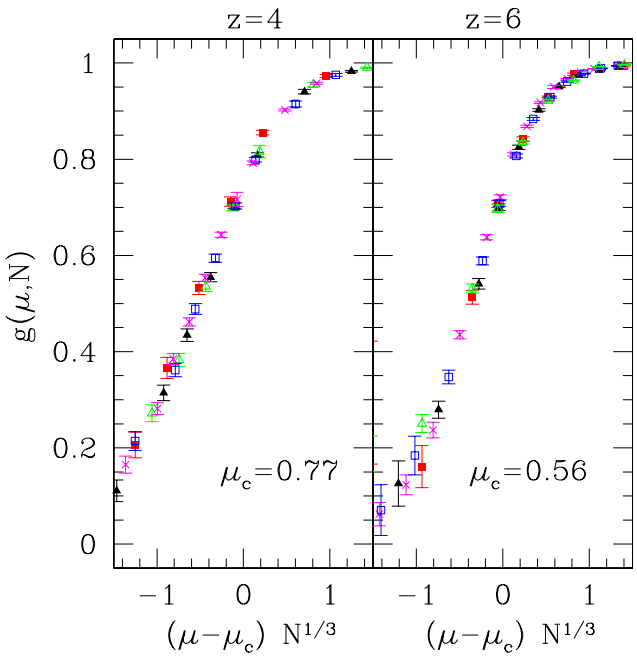


Figure 6.6: Scaling of the Binder cumulant. The symbols for each panel are the same as the corresponding panels in Figure 6.5.

6.3 Performance of Branch–and–Cut

The max-cut problem remains $\mathcal{NP}$-hard for the class of z-regular graphs. Hence, we only know solution algorithms with exponential *worst-case* running time. In practical computations, the running time of an algorithm can vary much from one instance of the problem to another. In case we are determining ground-states of the Bethe lattice in the ferromagnetic region, the 'worst case' might very rarely occur, the algorithm might be 'fast' and the notion of worst-case running time might be a very coarse measure in practice. Therefore recent work focuses on the *average* running time of solution algorithms. We can determine the average running time in practise by determining the average over the running times of random instances drawn from some probability distribution. As the median is less affected by rare samples with huge running times, we analyse the median of the running time and call it the *typical* running time. We would like to stress that the worst-case complexity is an algorithmic independent feature of the problem itself whereas the average or typical running time is dependent on the solution algorithm and can even vary between different implementations of the same algorithm.

In the following we study the running time as a function of the mean μ of the Gaussian distribution. We expect that if we choose μ big enough, the instance is

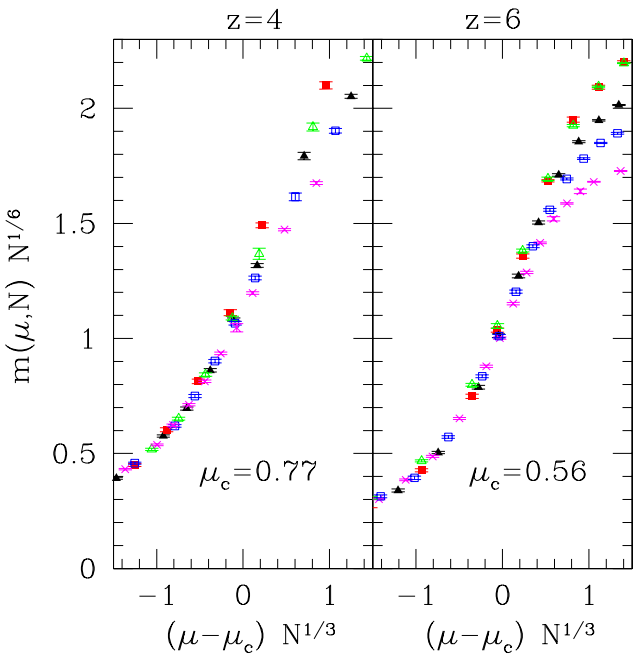


Figure 6.7: Scaling plot for the ground-state magnetization. The symbols for each panel are the same as those of Figure 6.5. We have significant corrections to scaling for $\mu > \mu_c$.

'easy' and the algorithm will be 'fast'; the typical running time will be 'high' for 'small' μ in the spin-glass phase. In the following we ask whether we see a *sharp* transition in the running time and, if so, whether the transition takes place in the vicinity of the spin glass/ferromagnet phase transition.

As branch–and–cut is basically branch–and–bound with cutting planes, we also did some experiments with a pure branch–and–bound algorithm [51, 69], which however can only deal with much smaller system sizes. We found that the running time behaves in a qualitatively similar but less clear way to branch–and–cut, in the accessible range of sizes.

As a measure for the running time, we might use the CPU time. However, we used different machines making the CPU time not suitable. Furthermore, it is hard to separate out the influence of size-dependent hardware effects on the CPU time. Small problems will be faster because they can be fully stored in the cache. Therefore we don't use the CPU time as performance measure. Instead, we use the number of linear problems solved, n_{lps}, as a measure of the running time [77]. The number of linear problems is strongly correlated with the CPU time. However, we note that the time needed for solving a linear program increases with the system size, and thus the number of linear problems underestimates the running time. (In Chapter 7 we quantify this behavior for the EA model.)

In Figure 6.8 we show the median running time as a function of μ for $z = 4, 6$ and different system sizes. The algorithm is fast in the ferromagnetic phase, whereas the running time increases significantly in the spin-glass phase. Within the spin-glass phase it is approximately constant. The effect gets stronger for bigger system sizes. This behavior suggests a sharp discontinuity in the $N \to \infty$ limit around $\mu \approx 0.8$ ($z = 4$) and $\mu \approx 0.6$ ($z = 6$), which is close to the spin glass/ferromagnetic transition μ_c.

In Figure 6.9 we see that deep in the ferromagnetic phase the data is consistent with a polynomial dependence of the running time on N. In contrast, for smaller values of μ, the curves are bending upwards, indicating that the running time increases faster than any polynomial. This is also visible for $\mu = 0.8$ ($z = 4$) and for $\mu = 0.6$ ($z = 6$, not shown). From this data it seems that the change in the typical running time of the branch–and–cut algorithm occurs at a value of μ larger than μ_c for the tested distribution of instances. It has been observed before that the phase transition and the change of the running time do not necessarily coincide, e.g., for a simple backtracking algorithm solving vertex cover [121].

We have tried an exponential fit for the data in Figure 6.9 of the form $n_{lps}(N) \sim \exp(bN^c)$. For $\mu = 0$, we find $b = 0.026(9)$, $c = 0.87(5)$ for $z = 4$, and $b = 0.007(3)$ and $c = 1.24(8)$ for $z = 6$. As there is a considerable scatter around the fitting region (and we obviously expect c to be smaller than 1), we cannot conclude that the running time is exponential. Nevertheless, the data strongly suggests that in the spin-glass phase the typical running time is superpolynomial.

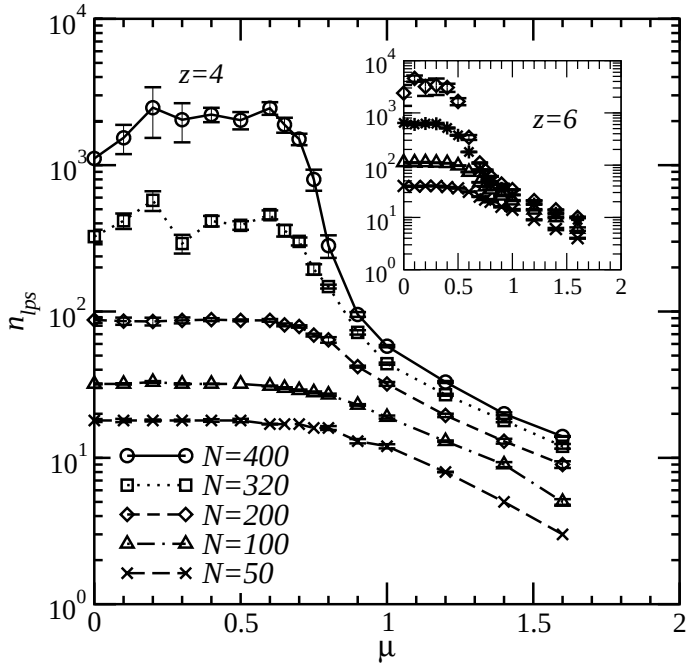


Figure 6.8: The median running time (measured in the number of linear problems solved) as a function of μ for different system sizes $N = 400$, 320, 200, 100, 50 for $z = 4$. The inset displays the running time for $z = 6$ and sizes $N = 200, 140, 100, 50$.

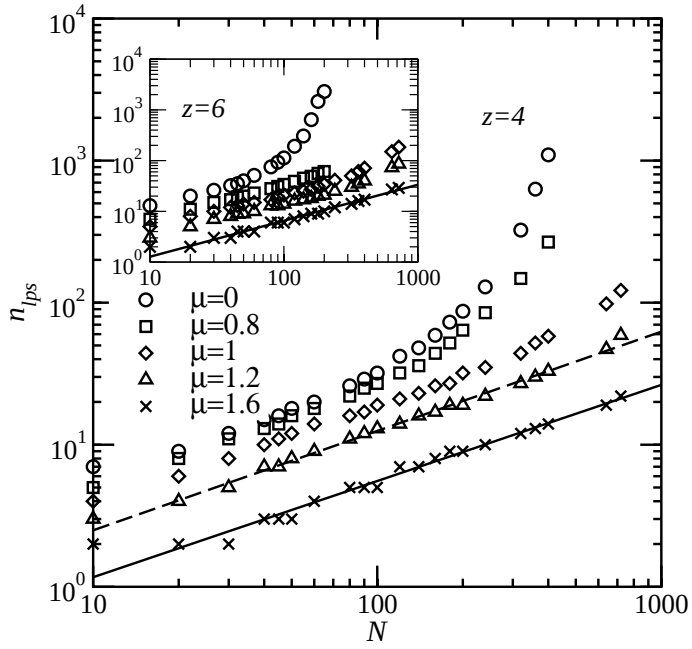


Figure 6.9: The median running time (measured in the number of linear problems solved) as a function of the number of spins N for different means of the Gaussian distribution $\mu = 0$,0.8,1, 1.2, 1.6 in a log-log plot. The straight lines represent power laws $c*N^{\zeta}$ with $\zeta = 0.699$ $(z = 4, \mu = 1.2)$, $\zeta = 0.677$ $(z = 4, \mu = 1.6)$ and $\zeta = 0.709$ $(z = 6, \mu = 1.6)$, respectively, showing that in the ferromagnetic phase the median running time of our program is polynomial.

In this project, we have shown that a solution algorithm for a standard problem from physics, the Ising spin glass, exhibits an easy-hard transition near a "physical" phase transition. We expect that phase transitions will have an effect on many solution algorithms. We believe similar phenomena can occur in other well known physical models.

In the following chapter we leave mean-field models and go over to studying the nature of the spin-glass state in the three-dimensional Edwards Anderson model.

Chapter 7

Low Energy Excitations in Spin Glasses

In this chapter we present results from a cooperation with Matteo Palassini and A. Peter Young. The full paper is already published in [101]. As described in Chapter 5, the nature of the spin-glass state for short-range models is not yet fully understood. In order to try to gain insight in its physics, we investigate the nature of the low-energy, large-scale excitations in the three-dimensional Edwards-Anderson model. We choose Gaussian distributed couplings and free boundary conditions and determine ground states for system sizes up to 12^3 spins. We study the response of the ground state to the bulk ϵ-perturbation introduced by Palassini and Young, [100] and used earlier in Section 5.3. The branch–and–cut data are consistent with a picture where the surface of the excitations is not space-filling, such as the droplet or the "TNT" picture. When allowing for large finite size corrections, the data are also consistent with a picture with space-filling surface, such as replica symmetry breaking. Finally, we analyze the performance of the current implementation of the branch–and–cut algorithm, finding a correlation between the running time and the existence of large-scale, low-energy excitations.

7.0.1 Introduction

As introduced in Chapter 5, two main theories for the nature of the spin-glass state have been proposed, the droplet theory and replica symmetry breaking. Recently, Krzkala and Martin (KM) [72] and Palassini and Young (PY) [100] have argued in favor of an intermediate scenario, the *TNT picture* ('trivial-nontrivial'). The TNT picture has been challenged (although in opposite senses) by Marinari and Parisi [78] and by Middleton [86]. Subsequently, low temperature Monte Carlo simulations [61] have found results consistent with the TNT scenario. The RSB, droplet, TNT

and some other scenarios have been also studied by Newman and Stein [94, 95]. For some recent related work, see Refs. [74, 91].

The work of KM and PY determined the ground state with and without a certain perturbation (which was different in the two cases), designed such that the ground state of the perturbed system is a large scale excitation of the original system. They used *heuristic algorithms*, and argue that they do find the exact ground state in most cases.

In the following we determine θ' and d_s from exact data. Our branch–and–cut code can handle significantly larger sizes for free boundary conditions (bc) than for periodic bc, so we use free bc here. (This behavior is intuitive: The cycle polytope coincides with the cut polytope for graphs without a K_5-minor, e.g., for planar graphs. The more edges we insert that 'destroy planarity', the worse the cycle polytope approximates the cut polytope. So we expect a better performance for instances with free than with periodic boundary conditions.) To our knowledge, the use of exact optimization algorithms in three-dimensional spin glasses has been restricted to smaller sizes than those studied here, and they were not used to investigate the real-space structure of the low-energy excitations.

For summarizing our results, we first note that for *free* bc, unlike for periodic bc, *each* scaling ansatz requires corrections to scaling. Making a natural assumption that these corrections are small we find $d - d_s = 0.45 \pm 0.02$ and $\theta' = 0.18 \pm 0.03$, which is compatible with the droplet picture. However, if we allow for very large corrections to scaling, we also cannot rule out the TNT or RSB pictures.

In the second part of the chapter we analyze the running time of the algorithm for the EA model. We find an exponential dependence on the system size which can be expected as we are studying an $\mathcal{NP}$-hard problem. We find that the computations take considerably longer for samples in which there is an excited state close in energy to the ground state energy, yet 'far away' in Hamming distance,i.e., very different in spin configuration. We are not aware of any previous quantitative measures of this trend, which we expect to be common to other algorithms as well.

The rest of this chapter is organized as follows. In Section 7.1 we describe the perturbation method. Our results for the nature of the large scale, low energy excitations are given in Section 7.2. The performance of the algorithm is analyzed in Section 7.3.

7.1 Model And Methods

We study the standard Hamiltonian

$$\mathcal{H} = -\sum_{\langle i,j \rangle} J_{ij} S_i S_j, \tag{7.1}$$

where $N = L^3$ spins i with spin variables $S_i = \pm 1$ occupy the sites of a simple lattice in $d = 3$ dimensions with free boundaries. The couplings J_{ij} are chosen from a Gaussian distribution with zero mean and standard deviation one.

The bulk perturbation of PY is already explained in Section 5.3. By applying the bulk perturbation, the *total* energy of the states changes by an amount of order unity.

Let the exact ground state of an instance have configuration $S_i^{(0)}$. Then the *link overlap* between the state "0" and any other state α is defined by

$$q_l^{(0,\alpha)} = \frac{1}{N_b} \sum_{\langle i,j \rangle} S_i^{(0)} S_j^{(0)} S_i^{(\alpha)} S_j^{(\alpha)}, \tag{7.2}$$

in which the sum is over all the N_b nearest neighbor bonds.

We denote the ground state of the perturbed system by $\tilde{S}_i^{(0)}$, and indicate by q_l and q, with no indices, the link- and spin overlap between the new and old ground states $S_i^{(0)}$ and $\tilde{S}_i^{(0)}$, where q is defined by $q = 1/N \sum S_i^{(0)} \tilde{S}_i^{(0)}$.

In the rest of the chapter we will restrict ourselves to $q \geq 0$ without loss of information.

In [102] and [101] the following scaling relations for the correlation functions link- and spin overlap are obtained.

$$\langle 1 - q \rangle = F_q(\epsilon L^{-\mu}) \tag{7.3}$$
$$\langle 1 - q_l \rangle = L^{-(d-d_s)} F_{q_l}(\epsilon L^{-\mu}), \tag{7.4}$$

where

$$\mu \equiv \theta' + d - d_s \tag{7.5}$$

and $\langle \cdots \rangle$ is the average with respect to the random couplings. By measuring $\langle 1 - q \rangle$ and $\langle 1 - q_l \rangle$ we can then determine $d - d_s$ and θ', and discriminate between the various pictures for the spin-glass phase.

The *asymptotic* behavior for $L \gg \epsilon^{1/\mu}$ is [101]

$$\langle 1 - q \rangle \sim \epsilon L^{-\mu}, \tag{7.6}$$
$$\langle 1 - q_l \rangle \sim \epsilon L^{-\mu_l}, \tag{7.7}$$

where

$$\mu_l \equiv \theta' + 2(d - d_s)\,. \tag{7.8}$$

In the RSB picture, the excitations are space-filling, thus $d - d_s = \theta' = 0$, and therefore $\mu = \mu_l = 0$. The scaling relations in Eqs. (7.3), (7.4) reduce to

$$\langle 1 - q \rangle = F_q(\epsilon)\,, \quad \langle 1 - q_l \rangle = F_{q_l}(\epsilon) \qquad (\text{RSB})\,, \tag{7.9}$$

and the asymptotic behavior for $L \to \infty$ is

$$\langle 1 - q \rangle, \quad \langle 1 - q_l \rangle \sim \epsilon \quad \text{(RSB)}. \tag{7.10}$$

In the following we will also analyze just those samples in which the unperturbed and perturbed ground states are very different, i.e., where $q \leq q_{\max}$, where $q_{\max}$ is a threshold value. Denoting such restricted averages by $\langle \cdots \rangle_c$, it is

$$\langle 1 - q_l \rangle_c = L^{-(d-d_s)} F_{q_l}^c(\epsilon L^{-\mu}). \tag{7.11}$$

In [101] we argue that the asymptotic behavior at large L is

$$\langle 1 - q_l \rangle_c \sim L^{-(d-d_s)}. \tag{7.12}$$

In particular, in RSB this becomes

$$\langle 1 - q_l \rangle_c \sim \text{const.} \quad \text{(RSB)}\,. \tag{7.13}$$

Note that in both cases the asymptotic behavior is independent of ϵ.

When analyzing the numerical data, we must be aware that there are corrections to finite-size scaling which occur when the treated system sizes L are not large enough. These take the form of *additive* corrections to scaling relations such as Eqs. (7.3), (7.4), and (7.11), whose amplitude is characterized by a correction to scaling exponent ω. For example, including the leading correction, Eq. (7.11) becomes

$$\langle 1 - q_l \rangle_c = \frac{1}{L^{d-d_s}} \left\{ F_{q_l}^c(\epsilon L^{-\mu}) + \frac{1}{L^\omega} G_{q_l}(\epsilon L^{-\mu}) \right\}, \tag{7.14}$$

and for $\epsilon L^{-\mu} \to 0$, the asymptotic result corresponding to Eq. (7.12) is

$$\langle 1 - q_l \rangle_c = \frac{1}{L^{d-d_s}} \left(a + \frac{b}{L^\omega} \right). \tag{7.15}$$

For the RSB case, this goes over to

$$\langle 1 - q_l \rangle_c = a + \frac{b}{L^\omega}, \tag{7.16}$$

rather than Eq. (7.13).

Even when these corrections are negligible and the scaling *form*, such as Eq. (7.11), is valid, the argument of the scaling function may not be sufficiently small for a simple power law dependency of the data on system size, such as Eq. (7.12), to be valid. In this regime, expanding the scaling function gives rise to *further* additive corrections to the *asymptotic* behavior. For example, the leading correction to Eq. (7.12), coming from expanding the $F_{q_l}^c$ in Eq. (7.11) to second order, will be

$$\langle 1 - q_l \rangle_c = \frac{1}{L^{d-d_s}} \left(a + b \frac{\epsilon}{L^\mu} \right) \tag{7.17}$$

which goes over to $\langle 1 - q_l \rangle_c = a + b\,\epsilon$ in RSB. In general, both types of corrections need to be borne in mind when fitting the data.

L	$\epsilon/\tau = \frac{1}{4}$	$\epsilon/\tau = \frac{1}{2}$	$\epsilon/\tau = 1$	$\epsilon/\tau = 2$	$\epsilon/\tau = 4$
4	50000	50000	50000	50000	50000
6	20000	20000	20000	20000	20000
8	15000	13467	13467	6000	6000
10	10000	7440	6000	4918	4000
12	5670		4202		

Table 7.1: Number of independent realizations of the disorder (samples) used in the computations.

7.2 Results for the Correlation Functions

We apply the bulk perturbation of PY to systems of size $L = 4, 6, 8, 10$, and 12. For each size, we consider five values of the perturbation strength ϵ, namely $\epsilon/\tau = \frac{1}{4}, \frac{1}{2}, 1, 2$, and 4, where $\tau = \sqrt{6}$ is the mean field transition temperature, except for $L = 12$ for which only $\epsilon/\tau = \frac{1}{4}$ and 1 are considered. We choose this value of τ for being able to compare our results with those of PY for periodic bc. In order to discriminate between the different pictures, it is important to have high statistics. Table 7.1 reports the number of samples computed for each size. Note that the number of samples necessary to achieve a given statistical error increases as ϵ decreases, since the fraction of samples in which the $\tilde{S}^{(0)} \neq S^{(0)}$ also decreases.

7.2.1 Box overlaps

In the paper [101] we first studied the nature of the spin glass phase by analyzing the results for the spin- and link overlaps. Then we studied the *box overlap*. Here we restrict ourselves to the results for the absolute value of the box overlap q_B defined as

$$q_B = \frac{1}{L_B^d} \sum_i S_i^{(0)} \tilde{S}_i^{(0)}, \tag{7.18}$$

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 q and q_l.

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 $1 - q_B \sim L^{-(d-d_s)}$ from which we obtain the scaling laws

$$\langle 1 - q_B \rangle = L^{-(d-d_s)} F_{q_B}(\epsilon/L^\mu) \tag{7.19}$$

$$\langle 1 - q_B \rangle_c = L^{-(d-d_s)} F^c_{q_B}(\epsilon/L^\mu) \tag{7.20}$$

where, as for the corresponding scaling functions for q_l, $F_{q_B}(x) \sim x$ and $F^c_{q_B}(x) \sim$ const. for small x. Hence the asymptotic scaling for $L \to \infty$ is

$$\langle 1 - q_B \rangle \sim \epsilon L^{-\mu_l} \tag{7.21}$$

$$\langle 1 - q_B \rangle_c \sim L^{-(d-d_s)} . \tag{7.22}$$

In RSB, this reduces to $\langle 1 - q_B \rangle \sim \epsilon$ and $\langle 1 - q_B \rangle_c \sim$ const.

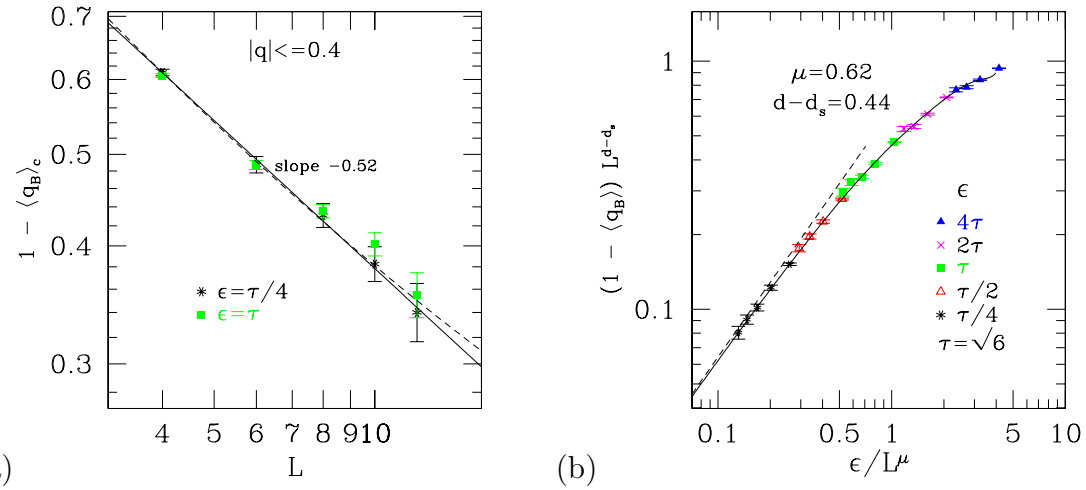


Figure 7.1: Box overlaps. (a) Logarithmic plot of the average box overlap, restricted to samples such that $q \leq 0.4$. The lower continuous line is a power-law fit for $\varepsilon/\tau = 4$. The dashed line is the fit with $\langle 1 - q_B \rangle_c = a + b/L + c/L^2$. (b) Scaling plot of the box overlap according to Eq. (7.19). The continuous line is a polynomial fit of order $n = 6$, which gives $\chi^2/\text{d.o.f} = 0.63$, 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 7.1(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. The exponent $d - d_s$ can be read off from the log-log plot Figure 7.1(a) as the slope of the straight line. We obtain the estimate

$$d - d_s = 0.48 \pm 0.03. \tag{7.23}$$

which is in agreement with the estimates $d - d_s = 0.44 \pm 0.03$ from $\langle 1 - q_l \rangle_c$, [101].

We observe that our data scale well, according to (7.19), see Figure 7.1(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.

Figure 7.2 shows the unrestricted average $\langle 1 - q_B \rangle$ multiplied by τ/ϵ, which asymptotically should be independent of ϵ. The data show a small curvature and a significant

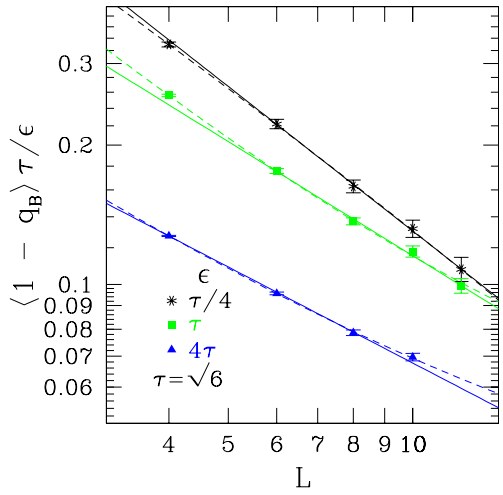


Figure 7.2: Logarithmic plot of the average box overlap, multiplied by τ/ϵ in order to highlight the deviation from the asymptotic behavior of Eq. (7.21) in which the data for various ϵ should collapse on a single curve. The continuous lines represent fits with the power-law $\langle 1-q_B\rangle_c = b/L^c$ excluding $L = 4$. The dashed lines represent fits of the form $\langle 1 - q_B\rangle_c = a + b/L^c$.

ϵ dependence, indicating that for this quantity we are not yet in the asymptotic region.

However, the data also fit the RSB picture well, if we allow large corrections to scaling. Under the RSB assumption we thus estimate $\lim_{L\to\infty}\langle 1 - q_B\rangle_c = 0.25 \pm 0.10$. In this case, the good scaling behavior we observed would only be a finite-size artifact, and would disappear at larger sizes.

It has turned out in [101] that the data for the box overlap can be fitted with smaller corrections to scaling than the data for the (bulk) link overlap. A fit to the generic scaling picture, with no corrections to scaling, gives results for the exponents $d-d_s$ and μ in agreement with those from the bulk quantities q, q_l analyzed in [101]. However, as with the bulk observables, assuming large corrections to scaling, the data can also be fitted to the RSB picture.

In the paper [101] the data of PY was analyzed again from [102] and $L \leq 8$ with periodic boundaries, obtained with a hybrid genetic algorithm. By imposing that corrections to scaling are less than the statistical errors of 1%, for periodic boundary conditions we obtain $\theta' \simeq 0$. The data are compatible with the TNT picture and show smaller corrections to scaling and deviations from the asymptotic scaling than the exact data for free bc. We find that the surface of the excitations is smaller for free bc than for periodic bc. For periodic bc, the domain wall has to "bend" to return to the same point on the "top surface" as it had on the "bottom surface". This may

be the source of the extra surface area. In general, it is reasonable to expect that there are some corrections to scaling which are larger for free bc, because these bc have a free surface on which lie a fraction of sites which is quite substantial for moderate sizes. For the same reason, the asymptotic behavior would set in for larger L. Evidence that free bc have larger corrections was also found recently in Monte Carlo simulations [62]. Then it is conceivable that the positive value of θ' obtained for free bc is also an effect of finite size corrections, and hence we cannot rule out a crossover to the TNT picture for larger sizes.

For free boundary conditions we obtain $\theta' = 0.19 \pm 0.06$, which fits well the droplet picture ($d - d_s > 0$, $\theta' > 0$). The results from analyzing the spin- and link overlap and the box overlap are in agreement. By relaxing this requirement and allowing larger corrections to scaling of order 10%, the data for free bc can be also fitted by a scenario with $\theta' \simeq 0$. Therefore the data for free bc are also consistent with the TNT picture provided moderate corrections to scaling are allowed, larger than those for periodic bc. We have also provided direct evidence that indeed free bc have larger corrections to scaling.

For both free and periodic bc, the data are also fitted well by the RSB picture ($d - d_s = 0$, $\theta' = 0$), but only if we allow very large corrections to scaling. In this case, the good scaling behavior we observed for all the observable considered would only be a finite size artifact, and would disappear at larger sizes. However, a droplet- or TNT scenario is more "natural" for our data.

Therefore, by current standards, it is not possible to clarify the nature of the spin-glass state by the work [101]. In order to do this, larger system sizes will be needed. This concludes the first part of this chapter. In the next section, we will analyze the performance of the branch–and–cut algorithm for the EA model.

7.3 Performance of Branch–And–Cut for the EA Model

In this section we study the performance of the current implementation of our branch–and–cut algorithm for the Edwards Anderson model. As the results for size $L = 12$ are obtained with a faster version of the code, the running times for this size cannot be compared with those for the smaller sizes. Hence, we will consider only sizes up to $L = 10$ in this section.

In Chapter 6, in which we studied the performance of branch–and–cut for the Bethe lattice, we already argued that it is useful to investigate experimentally the 'typical' running time needed for solving a max-cut instance, as the typical running time might be different to the worst-case running time which depends exponentially on the size of the system. The number of operations can vary significantly from one instance to another, and investigating the typical performance might be helpful for

identifying which parameters of the problem affect most the performance. De Simone et al. [112] measured the average CPU time used by the branch–and–cut algorithm to find the ground state of the two-dimensional $\pm J$ spin glass with periodic bc, up to $L = 70$, showing that the average CPU time was approximated by a function proportional to L^6.

In the following we analyze the performance of the branch–and–cut algorithm for the three-dimensional EA spin glass with free bc and Gaussian couplings. In Table 7.2, we summarize the average running time needed for calculating an unperturbed ground state for different system sizes. Similarly as in the Bethe lattice project, the CPU time is not a feasible performance measure because of hardly measurable hardware effects and because of the fact that the computations were performed on several different machines. As in Chapter 6, we take as performance measure n_p, the number of linear problems solved. n_p is a well-defined and machine independent

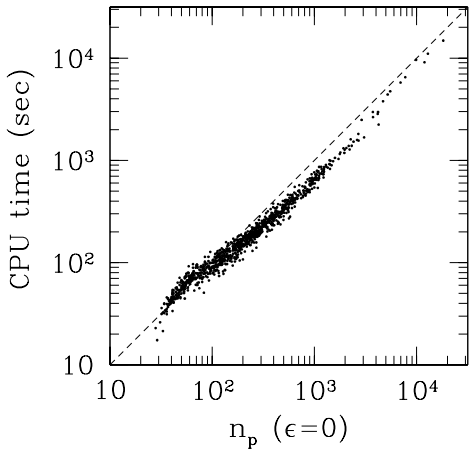


Figure 7.3: Scatter plot of the CPU time to find the unperturbed ground state ($\epsilon = 0$) versus the corresponding number of linear programs solved (n_p). Each point represents a randomly generated sample with $L = 10$. All the computations for this set of samples were run on the same machine. The dashed line indicates a linear behavior.

quantity. For three-dimensional EA spin-glass instances about 95% of the time is spent in solving linear programs. Furthermore, for a fixed system size, n_p correlates strongly, and almost linearly, with the CPU time. This is shown in Figure 7.3, which plots the CPU time versus n_p for 1000 randomly generated samples with $L = 10$, computed on the same machine. Note that since the *size* of the linear programs is also growing with the system size, the CPU time per linear program increases strongly with L: the average (resp. median) CPU time goes from 0.00770 (resp. 0.833) seconds for $L = 4$ to 0.833 (resp. 0.784) seconds for $L = 10$. Hence, n_p

L	mean CPU time per sample
4	0.065
6	0.662
8	10.11
10	338

Table 7.2: Mean CPU time per sample in seconds for the calculation of the unperturbed ground state, averaged also over different machines.

severely underestimate the rate at which the number of operations increases with L. As we see in Figure 7.3, the distribution of n_p (and CPU times) is very broad, extending over three orders of magnitude.

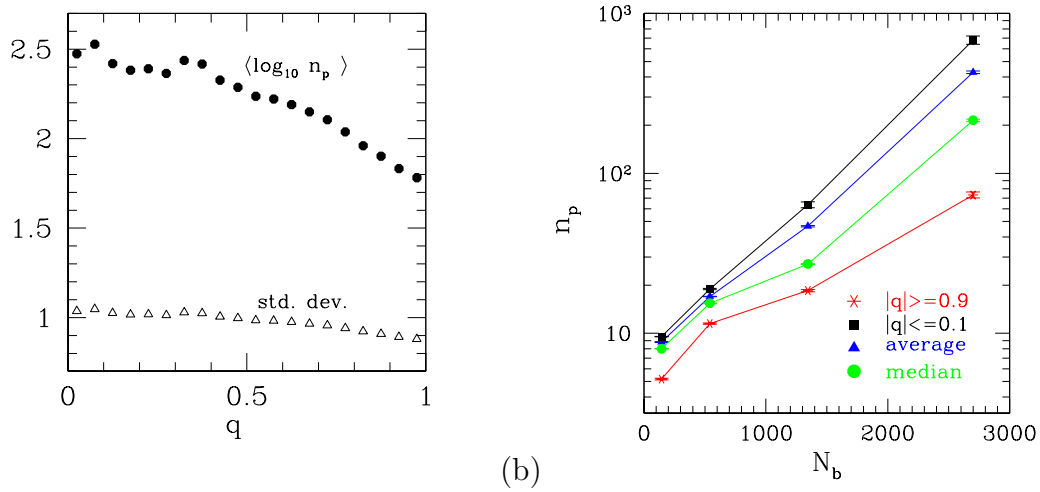
In order to identify which parameters of the problem, in addition to the size, affect the performance, we ask whether n_p correlates with the physical observables we measure. No significant correlation was observed with the ground-state energy. Figure 7.4 plots $\langle \log_{10} n_p \rangle$ for the unperturbed ground state ($\epsilon = 0$) and $L = 10$ versus the overlap between this state and the perturbed ground state with $\epsilon/\tau = 4$. (The perturbation method does not generate a uniform distribution of q, therefore Figure 7.4 was produced by selecting 1000 samples from a random ensemble, such that there is the same number of samples in each consecutive q interval of length 0.1 in the range $q \in [-1, 1]$.) We observe a distinct correlation between n_p and q: for small q, more linear programs are solved than for large q. The figure shows that the typical number of linear programs is close to an order of magnitude larger if $q \simeq 0$ than if $q \simeq 1$. We observed a similar correlation for other values of ϵ as well, and also between the CPU time and q. Again, the distribution of n_p is quite broad as shown by the data for the standard deviation of $\langle \log_{10} n_p \rangle$ in Figure 7.4.

In order to quantify how the correlation between n_p and q changes with the system size, we show in Figure 7.4 the average and median of n_p as a function of N_b, as well as the conditional averages of n_p restricted to samples with large ($|q| \geq 0.9$) and small ($|q| \leq 0.1$) overlap. We take the number of bonds, N_b, as a measure of the input size, since the branch–and–cut algorithm works on edge variables. First, all measures show an approximately exponential increase with N_b, with corrections for small N_b. Second, the difference between the conditional averages with small and large q seems to increase with the system size, and is about one order of magnitude for $L = 10$.

For understanding this behavior we note that we expect that samples with a small $|q|$ have a rougher "energy landscape" than samples with a big overlap, namely there are states with an energy close to the ground-state energy, however with a spin configuration very different from the ground state. It is then intuitively clear why one would observe a correlation between q and the running time for a stochastic

algorithm employing local search heuristics, such as simulated annealing, since when the algorithm encounters one of these configurations with small overlap, it must retrace its steps by a large amount.

However, for the branch–and–cut algorithm, a reason for the correlation between n_p and q is less obvious, but some insight is provided by an analysis of "reduced cost fixing" as explained in Chapter 4. The more variables that can be fixed during the run of the algorithm, the faster the algorithm is in practice.



(a) (b)

Figure 7.4: (a) The circles are a plot of $\langle \log_{10} n_p \rangle$, where n_p is the number of linear programs solved to compute the unperturbed ground state S^0, versus the overlap between $S^{(0)}$ and the perturbed ground state $\tilde{S}^{(0)}$. The data is for $\epsilon/\tau = 4$ and the samples were selected from a set of randomly generated samples with $L = 10$, in such a way that the same number of samples is plotted for each consecutive q interval of length 0.1, in order to sample equally all regions of q. The triangles show the standard deviation, among samples, of $\log_{10} n_p$ as a function of q. (b) Average n_p, median n_p, and conditional averages of n_p restricted to $|q| \leq 0.1$ and to $|q| \geq 0.9$, as a function of N_b. The data for n_p are for $L = 10$ and $\epsilon = 0$ (unperturbed ground state), and q is the overlap between the $\epsilon = 0$ and $\epsilon/\tau = 4$ ground states.

Since the samples with small overlap have "almost optimal" solutions with spin configurations far away in Hamming distance from the ground state, only a smaller number of variables can be fixed. Here we do not have the "correct" edge values for fixing available until the end. As an example, we checked that for $L = 10$ and $\epsilon = \tau$, for 100 randomly chosen samples with small overlap ($|q| \leq 0.1$), in average 409 ± 39 of the 2700 edge variables could be fixed in the first sub problem, i.e., before branching takes place. In contrast, for 100 randomly chosen samples with big overlap ($|q| \geq 0.9$), 921 ± 34 of the edge variables could be fixed in the first sub problem, about twice as many. Of course, the fewer variables that can be fixed

in the first sub problem, the more overall branching is necessary, resulting in more overall computational effort for samples with small overlap.

As a conclusion, we have shown that the performance of the algorithm is worse when the energy landscape is 'rough', i.e., in case there exists a solution near the ground-state energy that is very different in spin configuration.

Conclusion

The close connection between the max-cut problem from combinatorial optimization and the determination of exact ground states of Ising spin glasses makes it an exciting field of research. On one hand it is interesting to focus on the inclusion of advanced optimization techniques for the improvement of the performance of the branch–and–cut algorithm. On the other hand, analyzing exact ground-state data generated by branch–and–cut makes it possible to draw conclusions on the physics of spin glasses that are more reliable than those obtained by analyzing heuristically generated data.

In this thesis, we studied the problem of determining exact ground states of Ising spin glasses. This problem can be mapped on the prominent max-cut problem. It is $\mathcal{NP}$-hard, and we use a branch–and–cut algorithm for determining exact solutions for reasonably big system sizes. In the first part of the thesis we presented approaches for speeding up the max-cut computations within the branch–and–cut framework. To this end, we first summarized the state-of-the-art in the studies on the max-cut problem. We then found that linear relaxations perform better than the quadratic relaxation that can be solved by positive semidefinite optimization. We explained how to generate the cycle polytope fast for spin-glass instances. Subsequently we studied the question of how to solve the linear relaxations. We found that it is best to solve them with the traditional simplex method instead of interior point methods or the volume algorithm recently introduced by Barahona et al. After having focused on the cycle polytope, we asked ourselves how we can further strengthen the max-cut relaxations by adding inequalities beyond the cycles. Whereas the cut polytope is well-studied for dense and complete graphs, not much is known about the cut polytope on sparse graphs, e.g. instances defined on regular grids. In order to be able to use the knowledge for complete graphs, we applied a lift-project approach for the generation of tighter relaxations by adding cutting planes outside the template paradigm. We project to problem to a problem of smaller dimension. The resulting graph is dense, and we can separate classes of inequalities known for the complete graph. We can lift the generated facets to valid (under certain circumstances facet defining) inequalities for the cut polytope of the original sparse graph. We introduced a new facet for the max-cut problem, the 4-neighbor facet. Subsquently we presented improved primal heuristics for generating good cuts within the branch–and–cut framework. In the next chapter, we studied Ising spin-glasses on

the one-dimensional chain. We extended the branch–and–cut approach to a branch-cut&price algorithm that considerably speeds up the computations. Furthermore, it is possible to generate tighter relaxations than the cycle polytope in this model.

In the second part of this thesis we studied the physics of spin glasses. We first summarized the historical development. We started from the first surprising laboratory experiments done with spin glasses that showed that at low temperatures a random disorder of the magnetic moments is present. We then introduce the Ising model that is still widely used for theoretical studies. We introduce the Sherrington-Kirckpatrick model and summarize Parisi's solution. We point out that the nature of the spin-glass state is not yet fully understood for short-range models like the Edward Anderson model. We present several models that are proposed in the literature.

In the subsequent chapter we presented results from a cooperation with M. Palassini and A.K. Hartmann. In this cooperation we studied Ising spin-glass instances on z-regular graphs, i.e., Bethe lattices. The couplings are Gaussian distributed with mean μ. We determined the critical mean μ_c where the ferromagnetic/spin-glass phase transition takes place. As the usual worst-case notion for the running time gives us only a very coarse picture of the difficulty of determining ground states of spin glasses, we studied the performance of branch–and–cut in more detail. We analyzed the performance of the algorithm around the phase transition. We found that deep in the ferromagnetic phase the data is consistent with a polynomial dependence of the running time on the size of the input, whereas it seems to increase faster than polynomial in the spin-glass phase. In the last chapter of the thesis, we presented results from a cooperation with M. Palassini and A.P. Young. We studied the nature of the low-energy excitations in three-dimensional short-range spin glasses with Gaussian distribution and free boundaries. We found that the data is consistent with the droplet picture. However, we cannot rule out a crossover to a different picture, e.g., replica symmetry breaking or the TNT picture for system sizes bigger than those studied.

In order to clarify the nature of the spin-glass state beyond doubt, we need exact ground-state data of three-dimensional lattices with bigger sizes than we can study at present. Therefore, it is still an interesting problem how to further speed up the ground-state computations such that we can treat bigger sizes. At present, further improvements of the used relaxations yielding the upper bounds would be desirable. One promising approach is projecting and lifting. At present, the improvements in the upper bound through a lifting and projecting procedure are still too weak for a significant improvement of the upper bound. Hence it remains as an open problem whether different project- and lift approaches could help improving the performance. A different question would be how to design a powerful branching rule that branches on more than one variable. One approach in this flavor is the local branching procedure the we explained in Section 2.6.

It is also possible to study spin-glasses in external magnetic fields by branch–and–cut, and one can also try to clarify the nature of the spin-glass state by studying short-range models into an external magnetic field. Whereas the RSB picture predicts that the spin-glass phase can survive as long as the external magnetic field is not too big, e.g. in the droplet scaling picture the spin-glass phase is destroyed whenever a finite magnetic field is present.

It would be interesting to study one-dimensional Ising chain with the exact branch-cut&price method. Whereas parallel tempering cannot go to very large sizes for short-range models, this is possible with branch-cut&price. On the other hand, branch-cut&price cannot go to large sizes for the long-range model, so these two methods are complementary.

We believe it is very fruitful and stimulating to study a hard problem like spin glasses by exact algorithmic methods. We hope that the branch–and–cut algorithm for max-cut can further contribute to the understanding of its physics. Finally, we believe that solution algorithms can be improved when more insight is gained into the physics of the underlying problem.

Bibliography

[1] K. Aardal, S. P. M. van Hoesel, A. Koster, C. Mannino, and A. Sassano, *ZIB-Report 01-40*, December 2001.

[2] `www.informatik.uni-koeln.de/abacus`

[3] M.J. Alava, P.M. Duxbury, C.F. Moukarzel, H.Rieger, in *Phase transitions and critical phenomena, Vol. 18*, C. Domb and J. Lebowitz (eds.), Academic Press, (2001).

[4] J.R. de Almeida and D.J. Thouless, *J. Phys.*, **A 11**, 983, (1978).

[5] L. Bahiense, N. Maculan, and C. Sagastizábal, `www.optimization-online.org/DB_HTML/2001/11/400.html`

[6] J.R. Banavar, D. Sherrington and N. Sourlas, *J. Phys. A* **20** L1 (1987).

[7] F. Barahona *J. Phys. A*, Math. Gen. **15** 3241 (1982).

[8] F. Barahona, *Operations Research Letters* 2, 107 (1983).

[9] F. Barahona and R. Anbil, *Mathematical Programming* **87**, 385 (2000).

[10] F. Barahona, M. Grötschel, M. Jünger, and G. Reinelt, *Operations Research* **36**, 493 (1988).

[11] F. Barahona, M. Jünger, and G. Reinelt, *Mathematical Programming* **44**, 127 (1989).

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

[13] F. Barahona and A.R. Mahjoub, *Mathematical Programming*, **44**, 157, (1986).

[14] M. Bellare, O. Goldreich, M. Sudan Proc. of 36th Ann. IEEE Symp. on Fundations of Computer Science IEEE Computer Society 422 (1995).

[15] B. A. Berg and T. Celik, *Phys. Rev. Lett.* **69**, 2292 (1992).

[16] I. Bieche, R. Maynard, R. Rammal, and J.P. Uhry, *J. Phys. A* **13**, 2553 (1980).

[17] S. Boettcher, *Physical Review B* **67**, Rapid Communications 060403 (2003).

[18] K. Binder, *J. Phys. B* **43**, 119 (1981).

[19] K. Binder and A. P. Young, *Rev. Mod. Phys.* **58** 801 (1986).

[20] E. Boros and P. L. Hammer, *Annals of Operations Research* **33**, 151 (1991).

[21] A. J. Bray and M. A. Moore, in *Heidelberg Colloquium on Glassy Dynamics and Optimization*, L. Van Hemmen and I. Morgenstern (eds.), Springer-Verlag, Heidelberg (1986).

[22] A.J. Bray, M.A. Moore, and A.P. Young, *Phys. Rev. Letters* **56**, 2641 (1986).

[23] V. Cannella and J.A. Mydosh, (1972) *Phys. Rev. B*, **6**, 4220.

[24] P. Cheeseman, B. Kanefsky, and W.M. Taylor, Proceedings of the Twelfth International Joint Conference on Artificial Intelligence, IJCAI-91, Sidney, Australia.

[25] T. Christof and G. Reinelt *Algorithmica* **30**, 597 (2001).

[26] S. Cocco and R. Monasson, *Phys. Rev. Lett.* **86**, 1654 (2001).

[27] `http://oss.software.ibm.com/developerworks/opensource/coin/`

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

[29] M. Deza and M Laurent, *Mathematical Programming*, **56** 121 (1992).

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

[31] S.F. Edwards and P.W. Anderson, *Journal of Physics*, **F5**, 965.

[32] M. Elf, M. Jünger, and G. Rinaldi, Report No. 2001.409, Angewandte Mathematik und Informatik, Universität zu Köln, to appear in: *Operations Research Letters*, (2003).

[33] P. Festa, P.M. Pardalos, M.G.C. Resende, C.C. Ribeiro *Randomized Heuristics for the Max-Cut Problem* (2002).

[34] K.H. Fischer and A.A. Hertz, *Spin glasses* Cambridge University Press (1991).

[35] D.S. Fisher and D.A. Huse, *Phys. Rev B* **38**, 386 (1988).

[36] 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)

[37] M. Fischetti, A. Lodi, *Mathematical Programming* **98**, 23 (2003).

[38] S. Franz and M. Leone, *J. Stat. Phys.* **111** 535 (2003).

[39] A. Galluccio, M. Loebl, and J. Vondrák, *Mathematical Programming A*, **90**, 273 (2001).

[40] B. Gerards, *Mathematics of Operations Research* **10**, 359 (1985).

[41] M. X. Goemans and D. P. Williamson ACM Symposium on Theory of Computing (STOC) (1994).

[42] V.P. Grishukhin, *European Journal of Combinatorics*, **11**, 115 (1990).

[43] M. Grötschel, L. Lovász, and A. Schrijver, *Combinatorica* **4**, 169 (1981).

[44] F. Guerra, *Communications Mathematical Physics* **233**, 1, (2003).

[45] A.K. Hartmann, from exact ground-state calculation of 2d spin glasses up to size $N = 480^2$ (unpublished).

[46] A.K. Hartmann, *Phys. Rev. E* **59**, 84 (1999).

[47] A. K. Hartmann, *Phys. Rev. E* **59**, 84 (1999). A. J. Bray and M. A. Moore, *J. Phys. C*, **17**, L463 (1984); W. L. McMillan, *Phys. Rev. B* **30**, 476 (1984).

[48] A. K. Hartmann, H. Rieger, *Optimization Algorithms in Physics* (Wiley-VCH 2002)

[49] A.K. Hartmann and M. Weigt, *Theor. Comp. Science* **265**, 199 (2001).

[50] A.K. Hartmann and A.P. Young, *Phys. Rev. B* **66**, 094419 (2002).

[51] A. Hartwig, F. Daske and S. Kobe, *Comp. Phys. Commun.* **32** 133 (1984)

[52] B. Hayes, *Amer. Scient.* **85**, 108 (1997).

[53] J. Houdayer and O. C. Martin, *Phys. Rev. Lett.* **83**, 1030 (1999).

[54] M. Jünger and D. Naddef, eds., *Computational Combinatorial Optimization* (Lecture Notes in Computer Science 2241, Tutorial, Springer Verlag Heidelberg, 2001).

[55] M. Jünger, G. Reinelt, and G. Rinaldi, Technical Report, Universität zu Köln, in preparation.

[56] M. Jünger, G. Reinelt and S. Thienel, in *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, Volume 20, W. Cook, L. Lovasz and P. Seymour eds. (American Mathematical Society, 1995)

[57] M. Jünger and G. Rinaldi, in: P. Kischka et al. (eds.), *Operations Research Proceedings 1997*, Springer, 74 (1998).

[58] M. Jünger and S. Thienel, *Software Practice and Experience* **30**, 1325 (2000).

[59] R.M. Karp, C.H. Papadimitriou, *SIAM J. Comput.* 11 **4** 620 (1982).

[60] H. Katzgraber, personal communication (2004).

[61] H. G. Katzgraber, M. Palassini and A. P. Young, *Phys. Rev. B* **63**, 184422, (2001).

[62] H.G. Katzgraber and A.P. Young, *Phys. Rev. B* **65**, 214402, (2002).

[63] H.G. Katzgraber and A.P. Young, *Phys. Rev. B* **67**, 134410 (2003).

[64] H.G. Katzgraber and A.P. Young, *Phys. Rev. B* **68**, 224408 (2003).

[65] S. Kirkpatrick, C.D. Gelatt Jr. and M.P. Vecchi, *Science* **220**, 671 (1983).

[66] N. Kawashima and H. Rieger, in *Frustrated Spin Systems*, H.T.Diep (ed.), cond-mat/0312432 (2003).

[67] N. Kawashima and A.P. Young, *Phys. Rev. B* **53**, R484 (1996).

[68] Klein et al. *Phys. Rev.* **B 19**, 1492 (1979).

[69] T. Klotz and S. Kobe, *J. Phys. A*: Math. Gen. **27**, L95 (1994).

[70] S. Kobe and A. Hartwig, *Comp. Phys. Comm.* **16**, 1 (1978).

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

[72] F. Krzakala and O.C. Martin, *Phys. Rev. Lett.* **89**, 267202 (2002).

[73] P.Y.Lai and Y.Y.Goldschmidt, *J. Phys. A* **22**, 399 (1989).

[74] J. Lamarcq, J.-P. Bouchaud, O. C. Martin and M. Mezard, *Europhys. Lett.* 58(3), 321 (2002).

[75] A. H. Land and A. G. Doig, *Econometrica* **28**, 497 (1960).

[76] F. Liers, M. Jünger, G. Reinelt, and G. Rinaldi, Report No. 2003.452, Angewandte Mathematik und Informatik, Universität zu Köln, to appear in A.K. Hartmann and H. Rieger (eds.) *New Optimization Algorithms in Physics*

[77] F. Liers, M. Palassini, A.K. Hartmann and M. Jünger, *Phys. Rev. B* **68**, 094406 (2003).

[78] E. Marinari and G. Parisi, *Phys. Rev. Lett.* **86**, 3887 (2001).

[79] E. Marinari, G. Parisi, F. Ricci-Tersenghi, J.J. Ruiz-Lorenzo, *J. Phys. A* **31**, L481 (1998).

[80] E. Marinari, G. Parisi and J.J. Ruiz-Lorenzo, in *Spin Glasses and Random Fields*, edited by A.P. Young, World Scientific, Singapore, (1997).

[81] S. Mertens, *Phys. Rev. Lett.* **81**, 4281 (1998).

[82] M. Mézard, G. Parisi and M. A. Virasoro, *Spin Glass Theory and Beyond* (World Scientific, Singapore, 1987).

[83] M. Mézard and G. Parisi, *Eur. Phys. J. B* **20**, 217 (2001).

[84] M. Mézard and G. Parisi, cond-mat/0207121.

[85] W. L. McMillan, *J. Phys. C*, **17**, 3179 (1984).

[86] A. A. Middleton, *Phys. Rev. B* **63**, 060202(R) (2001).

[87] A. Middleton, *Phys. Rev. Lett.* **88**, 017202 (2002).

[88] D. Mitchell, B. Selman, and H. Levesque, in Proc. 10th Natl. Conf. Artif. Intell. (AAAI-92), 440 (AAAI Press/MIT Press, Cambridge, Massachusetts, 1992).

[89] R. Monasson, R. Zecchina, *Phys. Rev. E* **56**, 1357 (1997).

[90] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky, *Nature* 400, 133 (1999)

[91] M. A. Moore, cond-mat/0203469.

[92] P.Mottishaw, *Europhys. Lett.* **4** (1987) 333.

[93] J.A. Mydosh, *Spin glasses: an experimental introduction*, Taylor & Francis London, Washington, DC, (1993).

[94] C. M. Newman and D. L. Stein, *Phys. Rev. B* **46**, 973 (1992); *Phys. Rev. Lett.*, **76** 515 (1996); *Phys. Rev. E* **57** 1356 (1998).

[95] C.M. Newman and D.L. Stein, *Phys. Rev. Lett.* **87**, 077201-1, (2001)

[96] `www.neos.mcs.anl.gov/neos/`

[97] K.F. Pál, *Physica A*, **233**, 60 (1996).

[98] M. Palassini, 2000, unpublished.

[99] M. Palassini and A.P. Young, unpublished.

[100] M. Palassini and A.P. Young, *Phys. Rev. Lett.* **83**, 5126 (1999).

[101] M. Palassini, F. Liers, M. Jünger, and A. P. Young, *Phys. Rev. B* **68**, 064413 (2003).

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

[103] R.G. Palmer and J. Adler, *Int. J. Mod. Phys.C* **10**, 667 (1999).

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

[105] G. Parisi and F. Tria, cond-mat/0207144.

[106] V. Privman ed., *Finite Size Scaling and Numerical Simulation of Statistical Systems* World Scientific, Singapore (1990).

[107] `www.zib.de/Optimization/Software/Porta/`

[108] J.D. Reger, R.N. Bhatt and A.P. Young, *Phys. Rev. Lett.*, 64, 1859 (1990).

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

[110] H. Schramm and J. Zowe *SIAM J. Opt.* **2** 121, (1992).

[111] D. Sherrington and S. Kirkpatrick, *Phys. Rev. Lett.* **35**, 1792, (1975).

[112] C. De Simone, M. Diehl, M. Jünger, P. Mutzel, G. Reinelt and G. Rinaldi, *J. Stat. Phys.* **80**, 487 (1995).

[113] C. De Simone, M. Diehl, M. Jünger, P. Mutzel, G. Reinelt and G. Rinaldi, *J. Stat. Phys.* **84**, 1363 (1996).

[114] Small 0/1 Polytopes in Combinatorial Optimization, `www.iwr.uni-heidelberg.de/groups/comopt/software/SMAPO`

[115] A. Steger and N.C. Wormald, *Combin. Probab. Comput.* **8**, 377 (1999).

[116] M. Talagrand, *CRAS*, **337**, 111, (2003).

[117] D.J. Thouless, *Phys. Rev. Lett.* **56** (1986) 1082.

[118] Poljak and Turzik, *Discrete Mathematics* **108**, 379 (1992).

[119] L. Viana and A.J. Bray, *J. Phys. C* **18**, 3037 (1985).

[120] M. Weigt and A.K. Hartmann, *Phys. Rev. Lett.* **84**, 6118 (2000).

[121] M. Weigt and A.K. Hartmann, *Phys. Rev. Lett.* **86**, 1658 (2001).

[122] A. P. Young (ed.), *Spin Glasses and Random Fields*, World Scientific, Singapore, (1997).

Erklärung

Ich versichere, dass ich die von mir vorgelegte Dissertation selbstständig angefertigt, die benutzten Quellen und Hilfsmittel vollständig angegeben und die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken im Wortlaut oder dem Sinn nach entnommen sind, in jedem Einzelfall als Entlehnung kenntlich gemacht habe; dass diese Dissertation noch keiner anderen Fakultät oder Universität zur Prüfung vorgelegen hat; dass sie - abgesehen von unten angegebenen Teilpublikationen - noch nicht veröffentlicht worden ist sowie dass ich eine solche Veröffentlichung vor Abschluss des Promotionsverfahrens nicht vornehmen werde. Die Bestimmungen der Promotionsordnung sind mir bekannt. Die von mir vorgelegte Dissertation ist von Prof. Dr. Michael Jünger betreut worden.

Köln, 04.05.2004

Teilpublikationen

Matteo Palassini, Frauke Liers, Michael Jünger und A. P. Young Low energy excitations in spin glasses from exact ground states *Physical Review B* **68**, 064413 (2003).

Frauke Liers, Matteo Palassini, Alexander K. Hartmann und Michael Jünger. Zero-temperature transition of the Bethe lattice spin glass and the running time of an exact optimization algorithm *Physical Review B* **68**, 094406 (2003).