Engineering Branch-and-Cut Algorithms for the Equicut Problem

Miguel F. Anjos[1] , Frauke Liers[2] , Gregor Pardella[2] , Andreas Schmutzer[2]

[1] Canada Research Chair in Discrete Nonlinear Optimization in Engineering, GERAD & École Polytechnique de Montréal, Montreal, QC, Canada H3C 3A7
[2] Universität zu Köln, Institut für Informatik, Pohligstraße 1, 50937 Köln

Abstract. A minimum equicut of an edge-weighted graph is a partition of the nodes of the graph into two sets of equal size such that the sum of the weights of edges joining nodes in different partitions is minimum. We compare basic linear and semidefinite relaxations for the equicut problem, and find that linear bounds are competitive with the corresponding semidefinite ones but can be computed much faster. Motivated by an application of equicut in theoretical physics, we revisit an approach by Brunetta et al. and present an enhanced branch-and-cut algorithm. Our computational results suggest that the proposed branch-and-cut algorithm has a better performance than the algorithm of Brunetta et al.. Further, it is able to solve to optimality in reasonable time several instances with more than 200 nodes from the physics application.

1 Introduction

The maximum cut problem on an edge-weighted graph $G = (V, E)$ is to find a partition of the set of nodes V into two shores (node sets) such that the weight of the cut (the sum of the weights of edges with endpoints in different shores) is maximum. It is a prominent NP-hard combinatorial optimization problem that has been studied intensively in the literature.

We consider the equicut problem which is the max-cut problem with the additional restriction that the sizes (number of nodes) of the shores must be equal. This work is motivated by an application in theoretical physics: equicuts can be used for the calculation of minimum-energy states, or ground states, for so-called Coulomb glasses. In a Coulomb glass, charges may be placed on the sites of a lattice. The number of charges is exactly half the number of sites. Randomly chosen local fields act on the charges. Since a quadratic function is used to represent the energy of a state as a graph G, the task is to determine an equicut in G.

Polyhedral investigations of the equicut polytope have been presented in [12, 13, 15]. Building upon this theoretical knowledge, an integer programming-based branch-and-cut approach was implemented in [10]. The bisection problem is the more general task of determining a cut in which the shore sizes are constrained (but not necessarily equal). Formulations using integer programming, semidefinite programming (SDP), a polyhedral study and computational results are presented in [4, 5]. The maximum cut problem as well as related other problems have been investigated in great depth [16]. SDP-models for related graph partition problems have been introduced in [18, 21] and in the recent preprints [23, 28].

In this work, we first summarize important facts about the cut and the equicut polytopes and their relationships. The main part of this paper presents an algorithm engineering approach for an exact branch-and-cut algorithm for computing equicuts. Experimentally, we find that the

bounds obtained from the SDP relaxation with several additional linear constraints are usually not stronger than those from solving the same linear relaxation without the SDP model. Furthermore, the computation of the SDP bounds often needed considerably more time. We then focus on the usage of integer linear programming (ILP) methods. We take the most important ingredients from the method of [10] and enrich it by target cuts, a variant of local cuts [11]. It turns out that within this separation routine, orthogonal projections on node-induced subgraphs together with zero-lifting of the separated inequalities works well in practice. For complete graphs, the projection and lifting approach is equivalent to the graph shrinking procedure from [20]. As target cut separation yields very strong bounds but can be time-consuming, we then design fast heuristics to separate those inequalities found by target cut separation. We finally show that our approach yields an effective method for determining optimum equicuts in graphs, i.e. the computation time needed is about two orders of magnitude less than those reported in [10] on average. Furthermore we could reduce the average number of branching nodes by a factor of approximately four; for some of the largest instances the computation time was three orders of magnitude faster than those reported in [10]; and instances that required more than 50 branching nodes are now solved at the root node. Hence we could solve some large instances that were not reported in [10] as well as instances of Coulomb glasses to optimality. We also solved instances with more than 200 nodes, doubling the size of the instances reported in [10].

2 The Equicut Problem

Let a graph $G = (V, E)$ with node set V, edge set E, and edge weights $c : E \to \mathbb{R}$ be given. We denote by n and m the cardinalities of V and E, respectively. The complete graph on n nodes is denoted by K_n. For $S \subseteq V$ a cut $\delta(S) \subseteq E$ is the set of edges with exactly one node in S. Its weight is the sum of the weights of edges in $\delta(S)$. We call S and $V \setminus S$ shores of $\delta(S)$. A cut is an equicut if $\lfloor \frac{n}{2} \rfloor \leq |S| \leq \lceil \frac{n}{2} \rceil$. We associate with every cut $\delta(S)$ its characteristic vector $x(\delta(S)) = (x_1, \ldots, x_m) \in \{0,1\}^m$ where $x_e = 1$ if $e \in \delta(S)$ and $x_e = 0$ otherwise. The cut polytope $\mathcal{C}(G)$ and the equicut polytope $\mathcal{Q}(G)$ of a graph G are the convex hulls of the characteristic vectors of all cuts and of all equicuts of G, respectively. The equicut problem consists in finding an equicut in G with minimum weight.

2.1 The Cut and the Equicut Polytope

In this section, we review known results about the equicut polytope [12, 13].

For a complete graph K_{2p}, the edge set induced by an equicut contains exactly p^2 many edges. Furthermore, every equicut in K_{2p+1} contains $p(p+1)$ many edges. Thus, an equicut on a complete graph K_n has exactly $\lfloor \frac{n}{2} \rfloor \lceil \frac{n}{2} \rceil$ edges in the cut.

For complete graphs, the equicut polytope $\mathcal{Q}(K_n)$ can be defined as

$$\mathcal{Q}(K_n) = conv \left\{ x \in \mathcal{C}(K_n) \mid \sum_e x_e = \left\lfloor \frac{n}{2} \right\rfloor \left\lceil \frac{n}{2} \right\rceil \right\}. \tag{1}$$

Given a relaxation of the cut polytope we can use (1) to obtain a relaxation of the equicut problem as all inequalities remain valid.

We use the following classes of valid inequalities, some of which are well-known from the cut polytope [16]. For a more in-depth discussion see [12, 13] and [15].

Definition 1. *Let $K_p = (V', E')$ be a complete subgraph (clique) of $G = (V, E)$. The clique inequality is given by*

$$\sum_{e \in E'} x_e \leq \left\lfloor \frac{p}{2} \right\rfloor \left\lceil \frac{p}{2} \right\rceil . \tag{2}$$

For each cut $\delta(S)$ in K_p the switched clique inequality reads

$$\sum_{e \in E' \setminus \delta(S)} x_e - \sum_{e \in \delta(S)} x_e \leq \left\lfloor \frac{p}{2} \right\rfloor \left\lceil \frac{p}{2} \right\rceil - |\delta(S)|. \tag{3}$$

For the special case $p = 3$ the clique inequalities reduce to the triangle inequalities. For any triplet i, j, k such that $(i, j), (i, k), (j, k) \in E$, they take the following forms:

$$x_{ij} + x_{ik} + x_{jk} \leq 2 \tag{4}$$

$$x_{ij} - x_{ik} - x_{jk} \leq 0 \tag{5}$$

Barahona et al. [7] proved that the clique inequalities (2) are facets of the cut polytope $\mathcal{C}(K_p)$ iff p is odd. Hence the equicut polytope $\mathcal{Q}(K_n)$ is a face of the cut polytope and it is a facet iff n is odd [12].

As the number of cut edges in a cycle is even, for every cycle C with $|C| = p + 1$ and $n = 2p$ the cycle inequalities

$$\sum_{e \in C} x_e \geq 2 \tag{6}$$

are valid for $\mathcal{Q}(K_n)$. For a formal proof and facet inducing properties of these inequalities we refer to Conforti et al. [13].

If an edge $e = st$ is fixed to be in the cut, the resulting polytope is known as s-t-(equi-)cut polytope. Given a path $P = (s = v_0, v_1, \ldots, v_k = t)$ we consider the following s-t path inequalities that make sure that at least one edge on that path is cut:

$$\sum_{e \in P} x_e \geq 1 \tag{7}$$

It is easy to see that inequalities (7) are valid for the respective s-t-cut polytopes and they can be separated efficiently by shortest path computations. For further reasons to use these inequalities we refer to Conforti et al. [13].

For complete graphs K_{2p}, any equicut is also a p-regular subgraph of K_{2p} [10], i.e. each node has degree $|\delta(v)| = p$. Therefore, $\mathcal{Q}(K_{2p})$ is contained in $\mathcal{PF}(K_{2p})$, which is the convex hull of all p-regular subgraphs of K_{2p}. Edmonds et al. [17] give the following complete description of $\mathcal{PF}(K_{2p})$:

$$\sum_{e \in \delta(v)} x_e = p \tag{8}$$

$$\sum_{e \in W \times W} x_e + \sum_{e \in T} x_e \leq \frac{p|W| + |T| - 1}{2}, \tag{9}$$

where $v \in V$, $W \subseteq V$, $T \subseteq \delta(W)$ and $p|W| + |T|$ is odd. Inequalities (9) are called blossom inequalities.

We will work with complete graphs K_{2p} in the following as the inequalities (8) may be invalid for non-complete graphs.

2.2 Inequalities outside the Template Paradigm

The classes of inequalities introduced earlier follow the template paradigm, i.e. inequalities within a class share a similar structure. A general procedure to generate valid inequalities for some polytope is given by separation of local cuts [3] or their variant called target cuts [11]. For the separation, the polytope in question and the point to be separated are projected into a low-dimensional space. A cutting plane separating the projected point from the projected polytope is generated by solving a small linear program. The size of the linear program is basically determined by the number of vertices of the projected polytope. Inequalities are then lifted to inequalities valid for the original problem.

2.3 Non-Polyhedral Model

Given a weight matrix $C = \{c_{ij}\}$ for the edges $e = ij$ of a complete graph K_n the following SDP relaxation for the equicut problem was introduced by Frieze et al. [18]:

$$\min \left\{ \frac{1}{4} \mathit{trace}(C(J_n - X)) \mid \mathit{diag}(X) = e, Xe = 0, X \succeq 0 \right\}. \tag{10}$$

The notation $\mathit{trace}(A)$ refers to the sum of all elements of the main diagonal of a matrix A and the matrix J_n is the $n \times n$ matrix of all ones. In the positive semidefinite relaxation of the cut polytope [16], we restrict ourselves to positive-semidefinite matrices X ($X \succeq 0$) where all elements of the main diagonal are equal to one ($\mathit{diag}(X) = e$). This SDP relaxation is also referred to as the elliptope $\mathcal{E}_n$. A relaxation of the equicut problem is obtained by adding the constraint $Xe = 0$.

The variables $x_{ij} \in \{0, 1\}$ from the LP formulation are in one-to-one correspondence with the elements $\mathbf{x}_{ij} \in X$ of the positive-semidefinite matrix X via the transformation $\mathbf{x}_{ij} = 1 - 2x_{ij}$.

Our basic LP relaxation includes degree and triangle inequalities:

$$\min \left\{ \sum_e c_e x_e \mid \forall i, j, k \in V : (4) - (5), \forall v \in V : (8) \right\}. \tag{11}$$

As we will see, the relaxation obtained by separating (switched) clique (3), cycle (6) and blossom (9) inequalities is already very good in practice. In fact we can solve most instances of the benchmark from Brunetta et al. [10] at the root node without branching. In some cases a better performance of our method can be obtained by forcing the algorithm to branch when there is too little improvement in the objective value (c.f. tailing-off, section 3.1). Target cut separation may improve the performance when applied before branching.

In order to evaluate the respective bounds based on SDP and LP relaxations we first applied our branch-and-cut algorithm to the benchmark from Brunetta et al. [10]. For each instance we studied the relaxation that was obtained in the root, i.e. without branching.

In order to get reasonable computation times and memory usage we chose all instances with less than 10^5 inequalities in the final LP. There were 11 such instances. Since they were already solved at the root node, the respective SDP bounds are not significantly better than the LP bounds. In order to evaluate their strengths in practice nevertheless, we compare the LP bounds with those of the SDP when adding only a fraction of the inequalities separated by branch-and-cut. All SDP relaxations were computed using CSDP 6.1.1. [9]. The LPs were solved using version 12.1 of the CPLEX callable library [1]. We show in Table 1 the bounds that are obtained by solving the LP as well as the SDP when adding all separated inequalities (All), when adding only all triangle inequalities (Triangles) and when adding all inequalities but leaving out the (switched) clique inequalities (All but cliques). The generated inequalities are very strong in practice. It turns out that the triangle inequalities are important in practice as the corresponding bound is quite close to that given by all inequalities. This is known to hold for maximum cut as well, see e.g. [27].

Furthermore, the switched clique inequalities are most significant for improving the relaxation over the triangle bound as the remaining inequalities do not improve the bound any further. Similar results are obtained for most instances. Hence we conclude that using SDP relaxations does not significantly improve the root bounds compared to those obtained from the LP approach. Furthermore, the LP relaxations were obtained within minutes whereas some of the SDP relaxations needed several days of CPU time. Although better computing times would be observed using other methods such as bundle algorithms, we conclude that for our purposes using LP relaxations is preferable to SDP.

				Triangles		All but cliques	
Instance	Size	Opt	All	LP	SDP	LP	SDP
reti/2x11	22	11	11	10.24	10.34	10.24	10.34
tori/15x2	30	11	11	9.42	9.46	9.42	9.46
reti/16x2	32	10	9.44	6.27	6.27	6.27	6.27
tori/16x2	32	11	11	9.31	9.31	9.31	9.31
tori/18x2	36	13	13	9.58	9.58	9.58	9.58
misti/2x19m	38	388	386.61	383.88	383.88	383.88	383.88
reti/19x2	38	10	8.94	6.77	6.78	6.77	6.78
reti/2x19	38	6	4.83	2.89	2.9	2.89	2.9
rand/q0.20	40	1238	1238	1152	1237.73	1152	1237.73
rand/q0.60	40	530	530	489.55	528.32	489.55	528.32
reti/20x2	40	10	9.78	5.87	5.87	5.87	5.87

Table 1. LP and SDP bounds of root relaxations.

Armbruster et al. [5] present relaxations using LP and SDP methods to solve a generalization of the equicut problem, i.e. the minimum bisection problem, where the number of nodes in both shores has to be less than some parameter $F \leq \frac{n}{2}$. Their computational results on sparse instances suggest that SDP relaxations are superior to the corresponding LP relaxations. As equicut is a special case of the minimum bisection problem this appears to contradict the above observations. However, in contrast to the relaxations used in [5], our model is defined on complete graphs and includes constraints that are not valid for the minimum bisection polytope in general. Hence it is likely that our LP relaxation for the equicut problem is stronger than more general relaxations for

the minimum bisection problem, and that it is especially well suited for dense instances such those coming from the physics application.

3 Enhanced Branch-and-Cut Algorithm

Branch-and-cut is a framework often used for solving NP-hard combinatorial optimization problems exactly. Upper and lower bounds on the objective value are iteratively improved until optimality of a known solution can be proven. The size of the branching tree is kept small by using strong relaxations for the lower bounds and primal heuristics for the upper bounds. In this section, we describe cutting-plane separation and primal heuristics for our proposed branch-and-cut algorithm.

3.1 Cutting-Plane Separation

Given a (possibly fractional) vector $x^* \in \mathbb{R}^m, 0 \leq x^* \leq 1$, the separation problem asks for either an inequality violated by x^* or a proof that all inequalities valid for the polytope in question are satisfied. An algorithm that solves the separation problem for any fractional solution is called *exact*. In contrast *heuristic* separation algorithms find violated inequalities but if none is found, they cannot prove that there are no violated inequalities for the polytope. Separation algorithms are often defined for classes of inequalities such as those introduced in Section 2.

In general some classes of inequalities are more important than others or may become more important after several iterations. Brunetta et al. [10] use the relative change in the objective value from the previous iteration to decide which class of inequalities should be separated. Therefore they introduce certain threshold values for each class of inequalities. Hence they separate triangle inequalities if the relative change is above a certain value, then clique separation is applied if the relative change is above another value and then blossom inequalities and $(p+1)$-cycle inequalities are separated if the relative change is above their respective thresholds. If the relative change is below any thresholds branching is applied. Because the thresholds are very sensitive to small changes in the separation routines, we decided to use just a single threshold to decide whether any separation is applied or we branch. We found that a factor $\alpha = 10^{-4}$ for the relative change of the objective value was a reasonable choice. Further in each iteration we stop separation whenever we found a certain number of violated inequalities. We found that 500 violated inequalities are a reasonable choice for most instances.

The degree inequalities (8) are all added from the beginning. In contrast to the heuristics for separating blossom inequalities in [10], we separate them exactly by the efficient algorithm [24].

Finally, target cut separation is applied whenever the other routines do not find any violated inequality. Target cuts were introduced in [11]. The key observation for target cuts is the following. For $k \leq m$, let π be some projection $\mathbb{R}^m \to \mathbb{R}^k$. Then the projection $\overline{P} = \pi(P) \subseteq \mathbb{R}^k$ of some polytope P in $\mathbb{R}^m$ is the convex hull of all points in $\mathbb{R}^k$ that can be extended to a point in P. Thus, $\overline{P}$ is the convex hull of all points $\pi(x_1), \ldots, \pi(x_r) \in \mathbb{R}^k$ such that $x_1, \ldots, x_r$ are the vertices of P. For $k \ll m$, many of the $\pi(x_i)$ are equal so that for small k, $\overline{P}$ can be dealt with efficiently. For details, we refer the reader to [11].

For the equicut polytope, we use orthogonal projections to (node-induced) subgraphs. More specifically, edges incident on nodes which are not in the considered subgraph are neglected. The projected polytope is then again a cut polytope. Given an inequality valid for $\overline{P}$, the corresponding inequality with coefficients set to zero for the neglected edges is then valid for P ('zero-lifting').

For the maximum cut problem, another projection is given through shrinking nodes to supernodes as introduced by Jünger et al. [20]. This projection is especially tailored for sparse graphs. For an edge $(s,t) \in E$ nodes s,t are replaced by a supernode v. Loops and multiple edges are deleted. Given a valid inequality $ax \leq b$ for $\overline{P}$ with complete graphs the corresponding lifted inequality $a'x \leq b'$ w.l.o.g. is defined as $a'_{st} = 0$, $a'_{sn} = a_{vn}$ and $a'_{tn} = 0$ for all nodes that are neighbours of s, t and v. For more details about the general procedure we refer to [8, 25].

It is easy to see that for complete graphs the shrinking procedure is equivalent to the orthogonal projection that we use. Indeed, in the notation introduced above, a supernode v may be replaced by either s or t. Thus, when lifting an inequality the coefficients of variables x_{st} and x_{tn} are zero-lifted.

For the equicut problem, we can solve the target cut linear programs for subgraphs with up to 20 nodes within reasonable time. Therefore if we solve graphs with more than $n = 40$ nodes the subgraph induced by our projection always has less than $\lceil \frac{n}{2} \rceil$ nodes. Thus, for instances of interesting sizes the projection of the equicut polytope is again a cut polytope without any restrictions on the size of the shores.

Next, we describe how we choose the nodes of the subgraphs used for projection. In order to separate a valid inequality the projected fractional solution needs to be infeasible. It is well known, that the cut polytope is a very symmetric object, i.e. the structure of inequalities valid with equality is the same at each vertex of the polytope. Furthermore the barycenter of the (equi)cut polytope is the vector $m = (0.5, 0.5, \ldots, 0.5)$ and the closer a fractional solution $\tilde{x}$ is to m, the less likely it is to be violated. Therefore given the fractional value $\tilde{x}_e$ of an edge e we assume that the value $b_e = |0.5 - \tilde{x}_e|$ correlates to the probability that the variable x_e contributes to a violated inequality, and we assume that these probabilities can be cumulated as $b_u = \sum_{e \in \delta(u)} b_e$ at node u. Unfortunately choosing the first k nodes with the largest values of b_u may as well result in a subgraph where the fractional solution $\tilde{x}$ is integral, hence given an integer feasible solution there is no violated inequality. Consequently the more fractional the solution $\tilde{x}$ is, the more likely it will yield a violated inequality. Further results on the projection we used as well as on the performance of target cut separation are described in section 4.

3.2 Primal Heuristic

Given a fractional optimum solution $x^* \in \mathbb{R}^m$ of the current LP-relaxation, primal heuristics round x^* to a feasible solution that is hopefully better than the best one known to date. For the maximum cut problem, a primal heuristic works as follows [6]. A cut is given by a spanning tree where each edge is either a cut or a non-cut edge according to the corresponding value of the solution. The 'most decided' edges are used if the spanning tree in G is minimum with respect to the weights $w_e = \min(x^*_e, 1 - x^*_e)$ on the edges $e \in E$. The LP-values on the tree are then rounded appropriately, and the corresponding cut is returned. A minimum spanning tree may be computed by Prim's algorithm in $O(|E| + |V| \log |V|)$ [26].

For the equicut problem, a cut has to additionally satisfy the cardinality constraint on the shore size. We thus adapt the above greedy approach using Prim's algorithm. In each step, several trees are combined to larger trees until a spanning tree arises. Each of these trees induces a cut in the subgraph induced by its nodes, which we will call partial cuts.

Furthermore we have to make sure that in each iteration it is possible to combine these partial cuts to an equicut. We will call such a set of partial cuts compatible, and incompatible otherwise. Let a_i and b_i denote the size of the shores induced by the partial cut $\delta_i(S)$ and $d_i = |a_i - b_i|$ the absolute difference of the shore sizes. The subset-sum problem asks whether a subset $D' \subseteq \{d_i\}$ exists such

that $\sum_{d_i \in D'} d_i = k$. Choosing $k = \frac{\sum d_i}{2}$, the partial cuts are compatible if the answer to the subset-sum problem is positive, and incompatible otherwise. In general the subset-sum problem is NP-complete. Nevertheless, we use the well-known pseudo-polynomial algorithm due to Ibarra et al. [19] for the knapsack problem which can be used to solve the subset-sum problem as well. As in our case the number of items and their weights are bounded above by $|V|$, the pseudo-polynomial algorithm yields an $O(|V|^2)$ algorithm.

Within Prim's algorithm, we make sure that whenever an edge is added to the spanning tree the partial cuts are compatible. This can be achieved by either skipping critical edges which lead to incompatible partial cuts or by repairing the partial cuts in such a way that the partial cuts become compatible. In either case we must never add edges that lead to a cycle. We also avoid using edges with weights w_e larger than a certain value r by temporarily removing them from G. From our experiments, $r = 0.25$ is a good choice.

Then according to Prim's algorithm all edges are iterated in order of increasing values w_e and edges which lead to cycles are skipped. Our method then uses three phases where edges are added and components of the graph are joined until a spanning tree is found. In the first phase we avoid repairing the partial cuts by skipping critical edges. In the second phase edges which lead to incompatible partial cuts are added. After adding an edge the partial cuts are then repaired greedily with respect to a minimum increase of the total edge weights w_e of the tree and such that no partial cut induces a shore of size greater than $\frac{|V|}{2}$. In the third phase we iteratively join the two largest components, which may occur from removing all edges with weights $w_e > r$. Again if a shore size exceeds $\frac{|V|}{2}$ nodes we have to repair that shore as in the second phase. Finally, we apply the Kernighan-Lin heuristic [22] to the solution to further improve the objective value.

We applied the above heuristic in our branch-and-cut algorithm. For all instances that were computed, whenever the root relaxation was strong enough to avoid branching, the optimal solution had been found by our primal heuristic.

4 Computational Results

In this section we evaluate our proposed branch-and-cut algorithm which is based upon a reimplementation of the algorithm presented in [10] using state-of-the-art tools. We used C++ and version 12.1 of CPLEX callable library as the branch-and-cut framework. For our experiments we used machines with Intel Xeon CPUs E5410 at 2.33 GHz. We use instances from [10] and instances from the physics application to evaluate the performance of our algorithm. The data for the instances we used and complete tables of computational results are available at [2].

In the determination of ground states in Coulomb glasses, we need to compute the minimum of the energy function:

$$H(q) = \sum_{i<j} p_{ij} q_i q_j + \sum_i c_i q_i. \tag{12}$$

The values $q_i \in \{-1, 1\}$ represent the positive or negative charges of sites i and are to be optimized. We are interested in charge-neutral systems, i.e. $\sum_i q_i = 0$. Those sites are located on a lattice and the values p_{ij} represent the pairwise interaction of sites i and j. We can use the same variable transformation as in Section 2.3 to obtain a quadratic unconstrained binary optimization (QUBO) problem. It was proved by de Simone [14] that QUBO is equivalent to the

maximum cut problem. For this transformation the quadratic terms in the objective function are represented by edges in a graph. The linear terms are represented by edges connected to an artificial node s that is added to the graph. The Coulomb glass instances are generally defined on an even number of sites, hence the above transformation would yield a graph with an odd number of nodes. A complete graph K_{2n} is then obtained by adding another artificial node t. Further the constant term from the transformation to QUBO is represented by the weight of the edge st. Since we restrict to charge-neutral systems we have to find a minimum s-t equicut (c.f. section 2.1).

In order to improve the quality of the LP relaxation we used target cut separation. We found that a significant number of violated inequalities found by target cut separation were switched clique inequalities (3).

We use a greedy heuristic to separate switched clique inequalities that extends the algorithm described in [10] in a straight forward way. We start with the most violated triangle inequality which is also a switched clique inequality. Given a switched clique inequality for a clique $K_p = (V', E')$ and a cut $\delta(S)$ with $|S| \leq |\bar{S}|$, we iteratively compute switched clique inequalities for a clique K_{p+1} unless $p+1$ exceeds a certain node limit k. Considering the switching operation we further improve the violation of the inequality in each iteration by iteratively switching pairs of nodes i and j if the violation of the switched clique inequality is increased.

In Table 2 we give the results for instances reported in [10] with our branch-and-cut algorithm including switched clique separation and target cut separation with different projections to subgraphs with 15 nodes. Further we give the computation times and number of branching nodes reported in [10] as a reference.

The results suggest that target cut separation improves the LP relaxation for all projections and no choice of projection dominates the others in terms of branching nodes. Considering CPU times for most instances the overhead of target cut separation is moderate for the given size of projections but helps to reduce the number of branching nodes. Comparing our results with those reported in [10] is very difficult since their reported computation times are for experiments carried out in the mid-1990s. Nevertheless we point out that our computation times are two orders of magnitude smaller on average, and more importantly, we need fewer branching nodes.

In Table 3 we give the bounds at the root node, the number of branching nodes and the CPU time needed by our branch-and-cut algorithm with and without switched clique separation for instances from [10], including some instances that were not reported in [10]. Further we did not apply target cut separation. The results suggest that switched clique separation improves the bounds significantly.

Furthermore, as switched clique inequalities and target cut separation significantly improve the LP relaxations, we were able to solve larger instances than those reported in [10]. We illustrate this by presenting our results on Coulomb glass instances with up to 258 nodes. Table 4 gives the number of branching nodes and CPU time required. For some instances that could not be solved, we report the gaps after four days of computation.

5 Conclusions

Our experimental results support the conclusion that the proposed branch-and-cut algorithm based on a linear relaxation with additional switched clique inequalities and target cut separation is able to efficiently solve medium-sized instances of equicut and larger instances of Coulomb glasses. This new algorithm thus contributes to the practical solution of equicut problems.

			Root Bound			Branching				Time (sec)			
Instance	Nodes	Opt	N	R	G	N	R	G	B	N	R	G	B
rand/q0.90	40	63	*	*	*	1	1	1	1	4	5	5	179
rand/q0.80	40	199	*	*	*	1	1	1	1	73	86	93	581
rand/q0.70	40	354	*	*	*	1	1	1	5	17	20	21	3942
rand/q0.30	40	1056	1052.01	*	*	9	1	1	3	161	226	256	3303
rand/q0.10	40	1425	1420.38	*	*	9	1	1	7	99	173	169	3246
rand/q0.00	40	1606	*	*	*	1	1	1	1	44	48	54	1152
rand/c0.90	50	122	*	*	*	1	1	1	7	6	7	7	958
rand/c0.80	50	368	359.71	*	*	5	1	1	11	478	1150	1212	7225
rand/c0.70	50	603	585.54	*	*	3	1	1	1	1203	2864	3159	10223
rand/c0.30	50	1658	1510.82	*	*	3	1	1	9	815	1611	1554	20742
rand/c0.10	50	2226	2090.9	*	*	3	1	1	7	674	986	870	15602
rand/c0.00	50	2520	2510.78	*	*	7	1	1	5	420	552	549	10261
rand/c2.90	52	123	*	*	*	1	1	1	7	8	10	10	2042
rand/c4.90	54	160	*	*	*	1	1	1	17	32	36	40	2955
rand/c6.90	56	177	*	*	*	1	1	1	17	35	38	45	3143
rand/s0.90	60	238	235.97	*	*	3	1	1	7	240	341	451	10488
reti/5x8	40	18	*	*	*	1	1	1	7	2	2	3	851
reti/3x14	42	10	*	*	*	1	1	1	5	12	23	30	1256
reti/5x10	50	22	*	*	*	1	1	1	3	10	23	34	2843
reti/6x10	60	28	*	*	*	1	1	1	31	101	142	151	17994
tori/8x5	40	33	*	*	*	1	1	1	7	3	3	3	494
tori/21x2	42	9	*	*	*	1	1	1	3	3	18	24	1061
tori/23x2	46	9	*	*	*	1	1	1	3	60	63	69	2788
tori/4x12	48	24	*	*	*	1	1	1	5	4	18	24	3012
tori/5x10	50	33	*	*	*	1	1	1	13	12	28	37	2031
tori/10x6	60	42	41.81	*	*	3	1	1	3	97	123	100	6517
tori/7x10	70	45	*	*	*	1	1	1	33	17	19	23	28297
misti/10x4m	40	436	*	*	*	1	1	1	1	3	13	28	315
misti/5x10m	50	670	*	*	*	1	1	1	5	6	8	7	3212
misti/4x13m	52	721	*	*	*	1	1	1	7	27	28	35	5702
misti/13x4m	52	721	*	*	*	1	1	1	5	21	25	26	5105
misti/10x6m	60	954	*	*	*	1	1	1	9	22	38	46	12920
misti/10x7m	70	1288	*	*	*	1	1	1	13	60	84	110	127297
negative/q0.n.00	40	-471	-474.02	*	*	3	1	1	1	192	254	298	3173
negative/q0.n.40	40	-450	*	*	*	1	1	1	1	9	10	11	478
negative/q0.n.50	40	-389	*	*	*	1	1	1	1	59	64	77	1221
negative/q0.n.70	40	-298	*	*	*	1	1	1	1	29	29	35	688
negative/c0.n.00	50	-829	-1069.94	-954.81	*	3	2	1	5	2591	4717	4027	96847
negative/s0.n.80	60	-465	*	*	*	1	1	1	1	45	52	53	4892
real/ma.i	54	2	*	*	*	1	1	1	29	6	7	8	3218
real/me.i	60	3	*	*	*	1	1	1	37	7	7	9	6505
real/m6.i	70	7	*	*	*	1	1	1	55	215	251	242	5737
real/mb.i	74	4	*	*	*	1	1	1	33	889	1119	1153	3772
real/mc.i	74	6	*	*	*	1	1	1	53	92	129	149	27712
real/md.i	80	4	*	*	*	1	1	1	57	762	781	884	86140
real/mf.i	90	4	3.58	*	3.58	2	1	4	47	470	803	687	14659
real/m1.i	100	4	*	*	*	1	1	1	101	153	171	196	97271

Table 2. Root bounds, number of branching nodes and CPU time to solve instances reported in [10] without target cut separation (N), with random (R) and with greedy (G) projections. Number of branching nodes and computation times reported in [10] are given in columns (B). Results for instances with less than 40 nodes are omitted. Bounds are given as "*" if optimum was found in the root node.

			Root Bound		Branching		Time (sec)	
Instance	Nodes	Opt	No Switched	Switched	No Switched	Switched	No Switched	Switched
rand/c2.90	52	123	*	*	1	1	9	8
rand/c4.90	54	160	156.68	*	3	1	30	33
rand/c6.90	56	177	176.33	*	2	1	28	30
rand/c8.90	58	226	205.1	215.05	108	3	234	3200
rand/s0.90	60	238	229.45	*	8	1	72	276
reti/13x4	52	20	15.7	*	7	1	29	45
reti/6x10	60	28	26.43	*	6	1	80	101
reti/10x6	60	19	*	*	1	1	20	19
reti/7x10	70	23	21.58	*	15	1	252	198
tori/13x4	52	20	15.7	*	7	1	32	45
tori/6x10	60	35	*	*	1	1	7	7
tori/10x6	60	42	41.81	41.81	3	3	107	97
tori/7x10	70	45	*	*	1	1	18	17
tori/10x8t	80	43	42.99	42.99	2	2	450	429
misti/4x13m	52	721	*	*	1	1	28	26
misti/13x4m	52	721	*	*	1	1	25	21
misti/9x6m	54	792	*	*	1	1	17	16
misti/10x6m	60	954	*	*	1	1	22	23
misti/10x7m	70	1288	*	*	1	1	63	58
negative/s0.n.80	60	-465	*	*	1	1	53	45
negative/tt0.n.80	70	-550	-579.08	-579.08	56	3	1534	22170
negative/o0.n.80	80	-690	-725.88	-725.88	86	3	5232	31124
real/ma.i	54	2	*	*	1	1	6	6
real/me.i	60	3	*	*	1	1	8	7
real/m6.i	70	7	*	*	1	1	216	198
real/mc.i	74	6	5.87	*	3	1	116	89
real/mb.i	74	4	3.26	*	6	1	243	752
real/md.i	80	4	3.46	3.71	3	2	347	508
real/mf.i	90	4	3.58	3.58	4	2	470	423
real/m1.i	100	4	*	*	1	1	163	140
real/m8.i	148	7	*	*	1	1	551	604

Table 3. Root bounds, number of branching nodes and CPU time to solve instances from [10] with and without switched clique separation. Results for instances with less than 52 nodes are omitted. Bounds are given as "*" if optimum was found in the root node. No target cut separation was applied.

Instance	Nodes	Root Gap (%)	Branching	Time (h)
L14_dim2_seed1	198	0	1	23.37
L14_dim2_seed2	198	0	1	23.55
L14_dim2_seed3	198	0.05	1	25.3
L14_dim2_seed4	198	0	1	24.37
L14_dim2_seed5	198	0	1	27.47
L6_dim3_seed1	218	0.32	15	60.99
L6_dim3_seed2	218	1.58	19	61.93
L6_dim3_seed3	218	1.54	19	69.69
L6_dim3_seed4	218	0.01	1	36.54
L6_dim3_seed5	218	1.04	19	65.15
L16_dim2_seed1	258	3.84	-	> 72
L16_dim2_seed2	258	4.96	-	> 72
L16_dim2_seed3	258	3.69	-	> 72
L16_dim2_seed4	258	4.06	-	> 72
L16_dim2_seed5	258	3.73	-	> 72

Table 4. Relative gap at the root node, number of branching nodes and computation times for large Coulomb glass instances with two- and three-dimensional grid graphs. Instance names are given by the length of the grids L, its dimension and a random seed that is used to generate the values c_i of the local field. The number of nodes is given as $L^{dim} + 2$.

Most inequalities separated by target cut separation are hypermetric inequalities which is a very general class of inequalities [16]. It would be interesting to find heuristics to separate more specific hypermetric inequalities. Therefore target cut separation could be used to classify important inequalities. Further improving the performance of target cut separation would allow the use of larger subgraphs for the projection. Finally it would be interesting to improve the projections to find violated inequalities more efficiently.

Acknowledgements

Financial support from the German Science Foundation is acknowledged under contract Li 1675/1. The first author acknowledges financial support by the Alexander von Humboldt Foundation and by the Natural Science and Engineering Research Council of Canada. We thank Helmut G. Katzgraber, Creighton Thomas and Juan Carlos Andersen for providing us with instances for the physics application.

Bibliography

[1] CPLEX® Callable Library version 12.1 – C API Reference Manual. `ftp://ftp.software.ibm.com/software/websphere/ilog/docs/optimization/cplex/refcallablelibrary.pdf`, 2009.

[2] `http://cophy.informatik.uni-koeln.de/eng_eq_ref.html`, 2012.

[3] D. Applegate, R. E. Bixby, V. Chvátal, W. J. Cook, and et al. TSP cuts which do not conform to the template paradigm. *Computational Combinatorial Optimization: Optimal or Provably Near-Optimal Solutions*, 2241:261—303, 2001.

[4] M. Armbruster, M. Fügenschuh, C. Helmberg, and A. Martin. A Comparative Study of Linear and Semidefinite Branch-and-Cut Methods for Solving the Minimum Graph Bisection Problem. In *Integer Programming and Combinatorial Optimization*, IPCO'08, pages 112–124, 2008.

[5] M. Armbruster, M. Fügenschuh, C. Helmberg, and A. Martin. LP and SDP Branch-and-Cut Algorithms for the Minimum Graph Bisection Problem: A Computational Comparison. Optimization Online, Dec 2011.

[6] F. Barahona, M. Grötschel, M. Jünger, and G. Reinelt. An application of combinatorial optimization to statistical physics and circuit layout design. *Operations Research*, 36(3):493–513, 1988.

[7] F. Barahona, M. Grötschel, and A. R. Mahjoub. Facets of the Bipartite Subgraph Polytope. *Math. of Operations Research*, 10(2):340–358, 1985.

[8] T. Bonato. *Contraction-based Separation and Lifting for Solving the Max-Cut Problem.* PhD thesis, Universität Heidelberg, 2011.

[9] B. Borchers. CSDP, A C library for semidefinite programming. *Optimization Methods and Software*, 11(1–4):613–623, 1999. Special Issue: Interior Point Methods.

[10] L. Brunetta, M. Conforti, and G. Rinaldi. A Branch-And-Cut Algorithm for the Equicut Problem. *Math. Program. B*, 78(2):243–263, 1997.

[11] C. Buchheim, F. Liers, and M. Oswald. Local Cuts Revisited. *Operations Research Letters*, 36(4):430–433, Jul. 2008.

[12] M. Conforti, M. R. Rao, and A. Sassano. The Equipartition Polytope. I: Formulations, Dimension and Basic Facets. *Math. Program. A*, 49:49–70, 1990.

[13] M. Conforti, M. R. Rao, and A. Sassano. The Equipartition Polytope. II: Valid Inequalities and Facets. *Math. Program. A*, 49:71–90, 1990.

[14] C. de Simone. The Cut Polytope and the Boolean Quadric Polytope. *Discrete Mathematics*, 79(1):71–75, Jan. 1990.

[15] C. C. de Souza and M. Laurent. Some New Classes of Facets for the Equicut Polytope. *Discrete Applied Mathematics*, 62(1-3):167–191, 1995.

[16] M. M. Deza and M. Laurent. *Geometry of Cuts and Metrics.* 1st edition, 1997.

[17] J. Edmonds and E. Johnson. Matching: A Well-Solved Class of Integer Linear Programs. In *Combinatorial Optimization Eureka, You Shrink!*, pages 27–30. 2003.

[18] A. M. Frieze and M. Jerrum. Improved approximation algorithms for max k-cut and max bisection. In *Proceedings of the 4th International IPCO Conference on Integer Programming and Combinatorial Optimization*, pages 1–13, London, UK, 1995. Springer-Verlag.

[19] O. H. Ibarra and C. E. Kim. Fast Approximation Algorithms for the Knapsack and Sum of Subset Problems. *J. ACM*, 22:463–468, 1975.

[20] M. Jünger, G. Reinelt, and G. Rinaldi. Lifting and Separation Procedures for the Cut Polytope. Technical Report, in preparation.
[21] S. E. Karisch and F. Rendl. Semidefinite programming and graph equipartition. In *In Topics in Semidefinite and Interior-Point Methods*, pages 77–95. AMS, 1998.
[22] B.W. Kernighan and S. Lin. An Efficient Heuristic Procedure for Partitioning Graphs. *Bell System Technical Journal*, 49:291–307, 1970.
[23] E. De Klerk, C. Dobre, D.Pasechnik, and R. Sotirov. On semidefinite programming relaxations of maximum k-section. Technical report, 2011.
[24] A. N. Letchford, G. Reinelt, and D. O. Theis. A Faster Exact Separation Algorithm for Blossom Inequalities. In *IPCO'04*, pages 196–205, 2004.
[25] F. Liers. *Contributions to Determining Exact Ground-States of Ising Spin-Glasses and to their Physics*. PhD thesis, Universität zu Köln, 2004.
[26] R. C. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36:1389–1401, Nov. 1957.
[27] F. Rendl, G. Rinaldi, and A. Wiegele. Solving max-cut to optimality by intersecting semidefinite and polyhedral relaxations. *Mathematical Programming*, 212:307–335, 2010.
[28] R. Sotirov. Tractable semidefinite programming relaxation for the graph partition problem. Technical report, 2011.