Speeding up IP-based Algorithms for Constrained Quadratic 0–1 Optimization*

Christoph Buchheim[1], Frauke Liers[1], and Marcus Oswald[2]

[1]Universität zu Köln, Institut für Informatik, Pohligstraße 1, 50969 Köln, Germany
[2]Universität Heidelberg, Institut für Informatik, INF 368, 69120 Heidelberg, Germany

Abstract. In many practical applications, the task is to optimize a non-linear objective function over the vertices of a well-studied polytope as, e.g., the matching polytope or the travelling salesman polytope (TSP). Prominent examples are the quadratic assignment problem and the quadratic knapsack problem; further applications occur in various areas such as production planning or automatic graph drawing. In order to apply branch-and-cut methods for the exact solution of such problems, the objective function has to be linearized. However, the standard linearization usually leads to very weak relaxations. On the other hand, problem-specific polyhedral studies are often time-consuming. Our goal is the design of general separation routines that can replace detailed polyhedral studies of the resulting polytope and that can be used as a black box.

As unconstrained binary quadratic optimization is equivalent to the maximum cut problem, knowledge about cut polytopes can be used in our setting. Other separation routines are inspired by the local cuts that have been developed by Applegate, Bixby, Chvátal and Cook for faster solution of large-scale traveling salesman instances. Finally, we apply quadratic reformulations of the linear constraints as proposed by Helmberg, Rendl and Weismantel for the quadratic knapsack problem.

By extensive experiments, we show that a suitable combination of these methods leads to a drastical speedup in the solution of constrained quadratic 0–1 problems. We also discuss possible generalizations of these methods to arbitrary non-linear objective functions.

Key words: quadratic programming, maximum cut problem, local cuts, quadratic knapsack, crossing minimization, similar subgraphs

* Financial support from the German Science Foundation is acknowledged under contracts Bu 2313/1-1 and Li 1675/1-1. Partially supported by the Marie Curie RTN Adonet 504438 funded by the EU. A preliminary version of this paper appeared in the proceedings of WEA 2008 [3].

1 Introduction

Optimizing a linear objective function over binary variables under additional linear constraints is NP-hard in general. One of the most successful frameworks for solving such problems is branch-and-cut. In order to develop fast branch-and-cut algorithms, it is crucial to determine good outer descriptions of the polytope P consisting of the convex hull of all feasible solutions of the problem at hand. The branch-and-cut approach is well developed, and the facial description of many polytopes corresponding to classical combinatorial optimization problems is well understood. For several problems practically efficient implementations exist.

Instead of a linear objective function, we often desire to optimize a non-linear objective function over the vertices of P. We consider problems where the non-linearities are locally defined, i.e., where every non-linear term in the objective function depends on few variables. In this paper, we focus on quadratic functions, so that every non-linear term is a product of only two variables. However, the proposed methods can be adapted to general non-linear functions, as discussed later. The easiest example of a binary quadratic optimization problem is the maximum cut problem, which is equivalent to optimizing a degree-two polynomial over the hyper cube [8].

Many practical applications lead to non-linear objective functions in a natural way. Several crossing minimization problems that arise in automatic graph drawing and VLSI design can be modeled as quadratic optimization problems over linear ordering type polytopes. To give another example, the tool switching problem arising in production planning can be solved by minimizing a polynomial of degree three over a polytope that is closely related to the TSP.

In most integer programming based approaches to such non-linear 0–1 problems, the first step is to linearize the objective function by introducing artificial variables that represent the non-linear terms. This leads to the problem of optimizing the linearized objective function over some polytope Q defined in a higher-dimensional space, instead of optimizing a non-linear objective function over the original polytope P.

Clearly, all valid linear inequalities of P yield valid linear inequalities for Q. A naive branch-and-cut approach for the optimization over Q would use the separation routines known for P, in combination with simple linear constraints modeling the connection between original and new variables. According to our experience, the performance of such an approach is very weak. Often, facet-inducing inequalities for P do not induce facets of Q, and the variables modeling non-linear terms change the polyhedral structure significantly. This can even happen if only one product is introduced and linearized.

In view of this, one could decide to undertake a polyhedral investigation of Q and try to develop specialized separation routines. Doing this will – very probably – be time consuming. Instead, much (human and computer) time could be saved by having some effective black-box routines at hand that speed up the solution algorithms but need only very limited knowledge about the problem structure. For quadratic problems, we provide such black-box routines and show that they drastically improve the running time of the solution algorithms.

Assuming that P is well understood, we ask the following question: How can we exploit the knowledge of P for optimizing over Q, without detailed polyhedral studies of Q? Even if the user is willing to invest some specific knowledge of Q, he/she can still combine her own separation strategies with our general methods outlined below. Moreover, the constraints produced by our methods might give the user some more insight into the polyhedral structure of Q and point at important classes of cutting planes, which could be separated right from the start, using tailored separation algorithms.

We address the general separation problem from two complementary directions. First assume that the objective function is quadratic. In case the problem is unconstrained, one can formulate it as a maximum cut problem on an associated graph [8]. Even in the presence of constraints, valid inequalities for the cut polytope remain valid for Q after transformation, and can be separated using the same transformation. In several applications, the transformed constraints of P induce a face of the corresponding cut polytope, which gives some theoretical evidence that the inequalities derived from the cut polytope can be helpful. This approach can be generalized to arbitrary non-linear objective functions using the constructions proposed in [5, 6].

On the other hand, we want to exploit the knowledge of the structure of the feasible solutions in P. Our proposed separation routine is inspired by the local cuts that have been developed by Applegate, Bixby, Chvátal and Cook (ABC^2) [1]. With the help of local cuts, they could solve big TSP instances being unsolved before. Recently, we proposed a variant of the local cut generation procedure that has some advantageous features [4]. We call our cutting planes *target cuts*. The main difference to the local cuts lies in a modified LP formulation that makes it possible to avoid the time-consuming tilting steps, as always a facet of the projected polytope is determined that can immediately be lifted to a valid inequality for Q.

For non-linear problems, the local or target cut approach is well-suited, as every non-linear term is determined by the original variables, so that the number of vertices does not change from P to Q. In particular, going from linear to non-linear objective functions does not slow down the cut generation significantly. Another advantage of this approach is that the separation can be implemented as a general framework that applies to all problems in this class. The user only needs to input some information about the structure of the feasible solutions, which is much easier than understanding the structure of the corresponding polytope. This approach can be applied to arbitrary non-linear problems in which the non-linearities are locally defined.

Another general technique to strengthen a given LP relaxation corresponding to a quadratic problem with linear constraints consists of a quadratic reformulation of the constraints. The idea is to replace each linear constraint by an equivalent quadratic one. The latter can be linearized as usual, using the present product variables or introducing new ones if necessary. Moreover, since we assume variables to be binary, we can replace each square term x^2 by the original variable x. The latter replacement usually leads to a stronger relaxation than the one given by the original linear constraints. Such quadratic replacements have been examined systematically by Helmberg et al. [10] for the quadratic knapsack problem. They show experimentally that, in an SDP setting, the reformulations lead to much stronger dual bounds. It turns out that the same holds in our IP setting, and that these stronger bounds often lead to faster running times.

Our main contribution is to show that these three approaches are very easy to use and lead to much better performance of general branch-and-cut approaches. By extensive computational experiments we show that not only the number of nodes in the enumeration tree but also the running time decreases dramatically, when compared to an algorithm that only uses the standard separation routines for the well-studied polytope P.

For some classical quadratic 0–1 problems, such as the quadratic knapsack problem or the quadratic assignment problem, special-purpose algorithms and implementations exist that exploit the problem structure and lead to efficient algorithms. Clearly, we cannot compete with such problem-specific approaches. In this work, we aim at designing general-purpose methods that help improve the solution algorithms for quadratic problems for which not much is known about their structure. In particular, the reference point for our evaluation is the basic approach using standard linearization and separation for P.

The outline of this paper is as follows. We fix notation in Section 2. In Section 3, we discuss cutting planes derived from the cut polytope. In Section 4, we introduce target cuts and their usage in the context of quadratic problems. Subsequently, we discuss methods to replace linear constraints by stronger quadratic ones in Section 5. In Section 6, we explain some motivating applications: the quadratic matching problem in Section 6.1 and the quadratic linear ordering and the linear arrangement problem in Section 6.2 and 6.3. In Section 7, we present experimental results for these applications. Finally, in Section 8, we discuss possible generalizations of our techniques to higher-degree polynomials and general non-linear objective functions.

A preliminary version of this paper appeared in the proceedings of WEA 2008 [3].

2 Definitions

Consider a combinatorial optimization problem on a finite set E with feasible solutions $\mathcal{I} \subseteq 2^E$ and with a linear objective function $c(I) = \sum_{e \in I} c_e$, where $c_e \in \mathbb{R}$ for all $e \in E$. Without loss of generality, we desire to minimize $c(I)$ over all $I \in \mathcal{I}$. Let the polytope $P \subseteq \mathbb{R}^E$ denote the convex

hull of all incidence vectors of feasible solutions. The corresponding integer linear program reads

$$\begin{array}{rll} & \min & \sum_{e\in E} c_e x_e \\ \text{(P)} & \text{s.t.} & x \in P \\ & & x \in \{0;1\}^E \end{array}$$

In the following, we focus on objective functions that are quadratic in the variables x, i.e., we consider problems of the form

$$\begin{array}{rll} & \min & \sum_{e\in E} c_e x_e + \sum_{e,f\in E; e\neq f} c_{ef} x_e x_f \\ \text{(QP)} & \text{s.t.} & x \in P \\ & & x \in \{0;1\}^E, \end{array}$$

For problems defined on a graph $G = (V, E)$ with variables corresponding to edges, and for two edges $e = (i, j)$ and $f = (k, l)$, we will use the notations c_{ef}, $c_{(i,j)(k,l)}$, and c_{ijkl} interchangeably. In order to address (QP) by integer programming techniques, we apply the standard linearization: for each pair $\{e, f\}$ with $c_{ef} \neq 0$, we introduce a binary variable y_{ef} modeling $x_e x_f$, along with the constraints $y_{ef} \leq x_e$, $y_{ef} \leq x_f$, and $y_{ef} \geq x_e + x_f - 1$. The linearized problem then reads

$$\begin{array}{rll} & \min & \sum_{e\in E} c_e x_e + \sum_{e,f\in E; e\neq f} c_{ef} y_{ef} \\ \text{(LQP)} & \text{s.t.} & x \in P \\ & & y_{ef} \leq x_e, x_f \qquad \text{for all } \{e,f\} \text{ with } c_{ef} \neq 0 \\ & & y_{ef} \geq x_e + x_f - 1 \quad \text{for all } \{e,f\} \text{ with } c_{ef} \neq 0 \\ & & y_{ef} \in \{0;1\} \qquad \text{for all } \{e,f\} \text{ with } c_{ef} \neq 0 \\ & & x \in \{0;1\}^E . \end{array}$$

We are interested in the polytope Q spanned by all feasible solutions of (LQP).

Note that other methods for linearizing (QP) have been proposed in the literature. Nevertheless, we focus on the standard linearization, as it is the most natural and popular way to linearize (QP) and as it can easily be implemented.

3 Cutting Planes from Max-Cut

Consider a graph $G = (V, E)$ with edge weights w_e. For $W \subseteq V$, the cut $\delta(W)$ is defined as

$$\delta(W) = \{(u, v) \in E \mid u \in W, v \notin W\} .$$

Its weight is $\sum_{e\in\delta(W)} w_e$. The maximum cut problem asks for a cut of maximum weight and is NP-hard for general graphs. The corresponding cut polytope, i.e., the convex hull of incidence vectors of cuts, is well studied [2, 9], and practically efficient branch-and-cut implementations exist for its solution [12, 13].

It is a well-known result that the problem of optimizing a quadratic function over binary variables without further constraints is equivalent to determining a maximum cut in an auxiliary graph $G_{\text{lin}} = (V_{\text{lin}}, E_{\text{lin}})$ [8]. The latter contains a node for each variable x_e. For each quadratic term $x_e x_f$ occuring in the objective function with $c_{ef} \neq 0$, the edge set E_{lin} contains an edge between the nodes corresponding to x_e and x_f. Furthermore, an additional root node and edges from this node to all nodes in V_{lin} are introduced. Now there exists a simple linear transformation between the edge variables of G_{lin} in the maximum cut setting and the linear variables and their products in the unconstrained quadratic optimization setting. Under this transformation, P is isomorphic to the cut polytope of G_{lin} [8].

If P is the unit hypercube, solving (LQP) thus amounts to determining a maximum cut in G_{lin}, i.e., to optimizing over a cut polytope defined in the E_{lin}-dimensional space. If P is a strict subset of the unit hypercube, i.e., if additional constraints are present, these constraints can be transformed as well and we derive that P is isomorphic to a cut polytope with further linear constraints. In

particular, all inequalities valid for the cut polytope still yield inequalities valid for (LQP) and can be used in a cutting plane approach.

Clearly, intersecting the cut polytope with arbitrary hyperplanes in general yields a non-integer polytope. The structure of the resulting polytope can be very different from a cut polytope. In this case it is not clear whether the knowledge about the cut polytope can help solving the constrained optimization problem. However, several relevant applications exist in which the intersection of the cut polytope with a set of hyperplanes cuts out a face of a cut polytope, at least if certain product variables are present, e.g., for quadratic assignment and quadratic matching. The proof for the quadratic matching polytope is a slight modification of the proof for the quadratic assignment polytope. For the quadratic linear ordering problem, the same result can be obtained, see [7].

In any case, we obtain a correct separation algorithm for (LQP) based on cut separation. Within a branch-and-cut framework, we can always work in the original model and apply other separation algorithms as desired. When it comes to the cut separation, we build the graph G_{lin}, transform the fractional point, and separate the inequalities known for the cut polytope. Found cutting planes are transformed back to yield cutting planes for (LQP).

4 Target Cuts for Quadratic 0–1 Problems

Usually, separation routines aim at generating faces or facets of some polytope in question that share similar structure. They are said to follow the *template paradigm*. Recently, ABC^2 proposed some general separation routine yielding so-called *local cuts* that are inequalities outside the template paradigm for which the structure is not known [1]. The size of the problem is first reduced by projecting the incidence vectors of feasible solutions onto a small-dimensional space; ABC^2 do this by shrinking nodes into supernodes.

For $r \leq m$, let π denote a projection $\mathbb{R}^m \to \mathbb{R}^r$ and let $\overline{Q} = \pi(Q) \subseteq \mathbb{R}^r$ denote the convex hull of the projected feasible solutions. Let $x^* \in \mathbb{R}^m$ be the point to be separated and $\overline{x}^* = \pi(x^*)$ be its projection to $\mathbb{R}^r$. A face-inducing inequality that separates some projected fractional point from $\overline{Q}$ can be obtained by solving an appropriately chosen linear program. Its size is basically determined by the number of its vertices. Thus, if the dimension of $\overline{Q}$ is not too big, this is fast in practice. Furthermore, the size of the linear program can be reduced by several considerations, and by delayed column generation only necessary feasible solutions are enumerated. A found local cut is then sequentially lifted and tilted until it becomes a facet for $\overline{Q}$ and then lifted to become feasible for the original TSP polytope.

Recently, we proposed a variant of the local cuts that we call *target cuts* [4]. The local cut framework can easily be adapted to target cuts, however the time-consuming tilting steps can be omitted. The reason for this is that we propose a different cut-generating linear program that generates a facet of $\overline{Q}$ right away. Furthermore, the volume of the generated facet is expected to be big. In the following, we briefly explain the target cuts separation. Details can be found in [4]. Subsequently, we will show that their use is favorable in the context of quadratic problems.

Assuming for now that $\overline{Q}$ is full-dimensional, we choose a point $\overline{q}$ in the interior of $\overline{Q}$. In case the projected non-feasible point $\overline{x}^*$ is not contained in $\overline{Q}$, we want to return a cutting plane that separates $\overline{x}^*$ from $\overline{Q}$. We argue in [4] that a facet from $\overline{Q}$ can be obtained by solving the following linear program:

$$\begin{aligned} \max \quad & a^\top(\overline{x}^* - \overline{q}) \\ \text{s.t.} \quad & a^\top(\overline{x}_i - \overline{q}) \leq 1 \quad \text{for all } i = 1, \ldots, s \\ & a \in \mathbb{R}^r \end{aligned} \tag{1}$$

Here, $x_1, \ldots, x_s$ are the vertices of $\overline{Q}$. A facet for $\overline{Q}$ violated by $\overline{x}^*$ can be read off the optimum solution of (1) as follows. If the optimum value of (1) is greater than 1, the corresponding inequality $a^\top(x - \overline{q}) \leq 1$ is violated by $\overline{x}^*$. Otherwise, $\overline{x}^*$ is contained in $\overline{Q}$.

In case the dimension of $\overline{Q}$ is smaller than r, the linear program (1) can be unbounded. In this case, $a^\top(x - \overline{q}) = 0$ is a valid equation for $\overline{Q}$ violated by $\overline{x}^*$, if a is an unbounded ray in (1).

In order to reduce the size of LP (1), we adapted the delayed column generation procedure proposed for local cuts to the target cut case. The procedure requires an oracle for maximizing any linear function over $\overline{Q}$. Having this at hand, one starts with a small, possibly empty, set of vertices $\overline{x}_1, \ldots, \overline{x}_h$. Then a target cut $a^\top(x - \overline{q}) \leq 1$ is produced for the polytope $\overline{Q}_h = \text{conv}\{\overline{x}_1, \ldots, \overline{x}_h\}$, by solving the corresponding linear program. Then, the oracle is called to maximize the left-hand side of the inequality. In case the maximum is bigger than 1, we add the maximal solution as a new $\overline{x}_{h+1}$ to (1). Otherwise we stop the procedure, having found a valid target cut. This process is iterated until the generated inequality is found to be valid. The number of columns added in this procedure is usually much smaller than the number of vertices of $\overline{Q}$.

In order to use target cuts for quadratic problems, we need to specify which projection to choose. In general, there is no easy answer to this question, and the user might have to test the performance of different projections in order to find which one gives best results. The projection needs to allow fast recognition or enumeration of the points in $\mathbb{R}^r$ that can be extended to feasible solutions in the original space. For several applications this is possible with the trivial, i.e., orthogonal, projection onto some sub-graph or sub-space, or the projection through shrinking of nodes into supernodes. For a given (linear) projection, lifting of a found inequality is trivial.

For some problems, certain projections seem to be favorable to others. For example, in a problem in which the global structure is important, as is the case for the TSP, a projection through shrinking should be prefered in case it allows to characterize the points in $\mathbb{R}^r$ having a preimage in $\mathbb{R}^m$ under π. On the other hand, there are problems in which the local structure seems to be characteristic of the problem, as, e.g., for the matching problem. In the latter, trivial projections can be used.

The usage of target cuts allows the implementation of a general framework in which only the projection and the oracle need to be specified for the particular application; everything else is problem-independent. Moreover, target cuts are well-suited for quadratic 0–1 problems: the size of the cut generating program (1) remains moderate, as there is a bijection between the vertices of the polytope P and those of Q. Therefore, the projected linearized polytope $\overline{Q}$ has the same number of vertices as $\pi(P)$, so that the number of rows of (1) does not grow with the introduction of product variables. In other words, the additional product variables do not affect the performance considerably, which allows to deal with non-trivial chunk sizes.

5 Quadratic Reformulations of Linear Inequalities

It is well-known that LP relaxations can often be improved by replacing the linear constraints by equivalent quadratic ones. The latter can then be linearized as usual by variables representing the resulting products of original variables. Since all variables are binary in our context, every square term x_e^2 can be replaced by the original variable x_e. Due to this latter replacement, the new relaxation is often stronger than the original one, in particular in combination with a tight polyhedral description of the quadratic structure as described in Section 3.

However, there is a trade-off between the stronger bounds obtained by the reformulation and the increase in the number of variables implied by the linearization. In the case of a quadratic objective function, the situation is favorable, since all (or many) product variables are already present in the LP, so that using linearized quadratic reformulations usually does not increase complexity too much.

This reformulation idea was used, e.g., by Johnson [11] in his integer linear programming model for the quadratic assignment problem. A systematic investigation of such reformulations was given by Helmberg et al. [10] for the case of a single linear inequality, i.e., for the quadratic knapsack problem. They evaluate different reformulations in an SDP framework, giving theoretical results comparing their strength and experimental results on the improvement of dual bounds.

More precisely, they propose two ways to reformulate a given linear inequality $a^\top x \leq b$. First note that we may assume $a \geq 0$, otherwise we can consider complement variables $1 - x_e$. The first reformulation, called (SQK2), is given by the quadratic inequality $(a^\top x)^2 \leq b^2$. The second reformulation replaces the single linear inequality by a set of new quadratic inequalities, obtained

by multiplying both sides of $a^\top x \leq b$ with x_e, for each $e \in E$, and with $(1 - x_f)$ for some $f \in E$ chosen arbitrarily. The reformulation given by these $|E| + 1$ constraints is called (SQK3).

As shown in [10], the reformulation (SQK3) is at least as strong as (SQK2), which in turn is at least as strong as the original formulation, called (SQK1). These relations are reflected in the experimental results given in [10], showing that the quality of the dual bounds increases from (SQK1) to (SQK3). However, it is not obvious that this improvement is reflected in the running times. In particular, it is not clear whether it pays off to replace a single constraint by $|E| + 1$ other constraints in order to improve the relaxation.

In the following, we investigate this experimentally in our IP setting. Additionally, we consider combinations of (SQK1) to (SQK3) with the techniques explained in the previous sections. Since it turned out that (SQK3) yields far too many constraints to be competitive in practice, we decided to separate the new constraints dynamically, i.e., we add the constraints only when violated. This approach is already suggested in [10].

One has to take care when separating quadratic constraints dynamically in the case where not all product variables are present. In this case, when adding some linearized quadratic constraint, we might also have to add the corresponding linearization variables and the linear constraints linking these new variables to the corresponding original variables. This rises the following question: what LP-value should we assume for an absent variable y_{ef} corresponding to a product $x_e x_f$? In the next LP solution, the new variable will be restricted only by the linear inequalities

$$y_{ef} \leq x_e, \quad y_{ef} \leq x_f, \quad y_{ef} \geq 0, \quad y_{ef} \geq x_e + x_f - 1.$$

Hence we separate an inequality only if it is violated for *all* values of y_{ef} in this range. We can thus choose the LP-value for y_{ef} as follows: if the coefficient of y_{ef} in the inequality to be added is positive, we may set y_{ef} to its lower bound $\max\{0,\ x_e + x_f - 1\}$. Otherwise, we set it to the upper bound $\min\{x_e,\ x_f\}$.

6 Applications

Applications of constrained quadratic 0–1 optimization problems abound. One of the classical examples is the quadratic assignment problem; more recently the quadratic knapsack problem has attracted some interest. In the following, we consider two other problems: the quadratic matching and the quadratic linear ordering problem. More precisely, we consider applications that are naturally modeled as such problems. In Section 6.1, we discuss the problem of finding highly similar subgraphs, which can be modeled as a quadratic (bipartite) matching problem. In Sections 6.2 and 6.3, we deal with two applications of quadratic linear ordering: the bipartite crossing minimization problem and the linear arrangement problem.

6.1 Finding Highly Similar Subgraphs – Quadratic Matching

Assume we are given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, and we want to get insight into how similar the two graphs are. This problem occurs in several practical applications, e.g., in automatic graph drawing and computational biology. The task is to determine a matching of a subset or all nodes of G_1 to those of G_2 such that as many edges as possible in the two graphs are mapped onto each other. This problem is a generalization of the graph isomorphism problem.

In our situation, we also allow but penalize the case in which $u_1 \in V_1$ is matched with $u_2 \in V_2$ and $v_1 \in V_1$ with $v_2 \in V_2$, but exactly one of the edges (u_1, v_1) or (u_2, v_2) exists. A straight-forward model for this problem is the following quadratic matching formulation

$$\begin{array}{lll} & \max & \sum_{i \in V_1, j \in V_2} x_{ij} + \sum_{i,k \in V_1, j,l \in V_2} c_{ijkl} x_{ij} x_{kl} \\ \text{(QMP)} & \text{s.t.} & \sum_{i \in V_1} x_{ij} \leq 1 \qquad \forall j \in V_2 \\ & & \sum_{j \in V_2} x_{ij} \leq 1 \qquad \forall i \in V_1 \\ & & x_{ij} \in \{0;1\} \ \forall i \in V_1, j \in V_2 \end{array}$$

with costs $c_{ijkl} < 0$ if either $(i,k) \in E_1$ or $(j,l) \in E_2$, but not both. Otherwise $c_{ijkl} \geq 0$. In this model, $x_{ij} = 1$ means that node $i \in V_1$ is matched with node $j \in V_2$.

6.2 Bipartite Crossing Minimization – Quadratic Linear Ordering I

Consider a bipartite graph $G = (V_1 \cup V_2, E)$. We want to draw G in the plane so that the nodes of V_1 and V_2 are placed on two parallel horizontal lines. The task is to minimize the number of crossings between edges, assuming that all edges are drawn as straight lines. Several applications exist in the area of automatic graph drawing. Clearly, the number of crossings only depends on the orders of vertices on the two lines.

First, we assume that the nodes V_1 on the upper level are layouted in some fixed order, whereas the nodes on the lower level are allowed to permute within the layer. The permutation of the nodes in V_2 has to be chosen such that the number of edge crossings is minimal. Let $i, j \in V_1$, $k, l \in V_2$ and edges $(i,k), (j,l)$ be present. Assume i is before j in the fixed order. No crossing exists in case k is before l on the second level, otherwise there is a crossing.

Hence the bipartite crossing minimization problem with one fixed layer can easily be formulated as a linear ordering problem. Now let us formulate the problem with two free layers as a quadratic optimization problem over the linear ordering polytope. For i, j, k, l chosen as above, there is no crossing in case i is before j and k is before l, or j is before i and l is before k. Let us introduce variables x_{uv} that take value 1 if u is drawn before v, and 0 otherwise. Then we have to solve the problem

$$\begin{array}{lll} & \max & \sum_{(i,k),(j,l)\in E} x_{ij}x_{kl} \\ (\text{QLO}_1) & \text{s.t.} & x \in P_{LO} \\ & & x_{ij} \in \{0;1\}\ \forall i,j \in V_1 \text{ or } i,j \in V_2\ , \end{array}$$

where P_{LO} is the linear ordering polytope.

6.3 Linear Arrangement – Quadratic Linear Ordering II

The linear arrangement problems is given as follows. We are looking for a permutation of n objects in such a way that a linear function c on the differences of positions of the objects is minimized. More precisely, we desire to determine a permutation π of $\{1, \ldots, n\}$ minimizing

$$\sum_{1 \le i,j \le n} c_{ij} |\pi(i) - \pi(j)|\ .$$

To this end we use the fact that the distances of the positions of two elements i and j with respect to a permutation π can be expressed in terms of betweenness variables. This distance equals 1 plus the number of elements lying between i and j, i.e., $|\pi(i) - \pi(j)| = 1 + \sum_k x_{ik}x_{kj}$ where x_{ij} is the usual linear odering variable modeling whether $\pi(i) < \pi(j)$ or not. Therefore, up to a constant, the linear arrangement problem can be rewritten as

$$\begin{array}{lll} & \max & \sum_{i\neq j\neq k\neq i} c_{ij}x_{ik}x_{kj} \\ (\text{QLO}_2) & \text{s.t.} & x \in P_{LO} \\ & & x_{ij} \in \{0;1\}\ \forall i,j \in \{1 \ldots n\}, i \neq j. \end{array}$$

Note that for this application only products of linear ordering variables are required that are of the type $x_{ik}x_{kj}$, which are only $O(n^3)$ many.

7 Experiments

We implemented the three separation approaches discussed in Sections 3, 4 and 5 within the branch-and-cut framework ABACUS, using CPLEX 11. All test runs were performed on Xeon machines with 2.66 GHz.

For each application we addressed, we start a branch-and-cut algorithm with the linear programming relaxation of the linearized problem (LQP). Separation routines for the polytopes P are assumed to be readily available. We compare the performance of this basic approach with the same approach extended by appropriately used maximum cut separation as described in Section 3,

quadratic reformulations as explained in Section 5, and the target cut separation as introduced in Section 4. For the tested applications, we used trivial projections onto subsets of variables, called *chunks*.

The chunks were chosen randomly in the sense that we first generate a subgraph randomly and then project onto all those linear and product variables that are completely determined by the subgraph. For the maximum cut separation, we separate the cycle inequalities [2]. For (SQK2), the starting relaxation consists of all reformulated inequalities. As the number of constraints in (SQK3) multiplies with the number of linear variables, we separate the reformulated inequalities on the fly. As in the beginning of the optimization process we introduce only the set of product variables having non-zero coefficient in the objective, it can happen that a violated reformulated inequality has a non-zero coefficient on a product that was not yet introduced in the problem formulation. In this case, we also introduce the needed product variables together with the corresponding linearization constraints to the model, as explained in Section 5.

We aimed at developing one relatively abstract implementation that can easily be used for all quadratic problems of type (QP) without having to incorporate many changes. Only the target-cut oracle and the test whether some vector represents a feasible solution are specific to the problem and have to be implemented separately for each application. We tested our approaches mainly on randomly generated instances. We put an upper limit of 10^4 on the number of sub problems. Instances that could not be solved within this limit are appropriately marked in the tables.

7.1 The Quadratic Matching Problem

For the quadratic matching problem, we studied instances defined on complete graphs. Note that a product $x_{ij}x_{kl}$ is necessarily zero if i, j, k, l are not pairwise distinct. We create random instances where for given pairwise distinct i, j, k, l the weight c_{ijkl} is non-zero with a given probability p. In this case, the weight is randomly chosen from $\{-1000, \ldots, 1000\}$. All linear weights c_{ij} are also chosen randomly from $\{-1000, \ldots, 1000\}$. An instance is thus defined by the number of nodes n, the percentage p of products with non-zero coefficient, and a random seed r for the weights.

Our implementations either determine a maximum quadratic matching or a minimum quadratic perfect matching. In the basic branch-and-bound approach, we separate the blossom inequalities that are known to be the only non-trivial facets of the matching polytope. We compare this basic approach with a branch-and-cut algorithm that uses separation of cutting planes derived from the cut polytope, from the reformulation (SQK3), and of target cuts on varying chunk sizes, as explained in Sections 3, 4 and 5. Reformulation (SQK2) yields only trivial inequalities, and therefore is always turned off. We also test an implementation in which all three features are turned on.

It turns out that better performance can be achieved if the maximum-cut separation procedure is only called in the root node of the branch-and-bound tree, and not after branching has been done. For the target cuts, the extendable solutions under a trivial projection are the incidence vectors of quadratic (not necessarily perfect) matchings. For their generation, two methods are implemented: For big chunk sizes, a heuristic greedy oracle first tries to identify fast necessary incidence vectors of quadratic matchings. In case it is successful, the delayed column generation procedure continues. In case it is not successful, we test whether a violating vector exists by calling an exact oracle. In the latter, the integer programming formulation for the quadratic matching problem on the small chunk is solved exactly. The column generation procedure is iterated until no more violating vector is found by the exact oracle. If the chunk sizes are not too big, one can also do a brute-force enumeration of all characteristic vectors of feasible solutions instead of the oracle calls. For the instance and oracle sizes studied here, the enumerative approach was faster than the one using oracles.

We show some running times for instances of the maximum quadratic matching problem in Table 1. Results for quadratic perfect matchings are comparable. We report the cpu time in seconds and the number of subproblems needed to solve the instance to optimality. IP refers to the basic algorithm, MxSyTz means that we apply cut generation if $x = 1$, reformulation (SQK3) if $y = 3$ and target cut separation with chunk size z.

$\|V\|$	p	IP		M0S3T0		M0S3T5		M0S3T6		M1S0T0		M1S0T5		M1S0T6		M1S3T0		M1S3T5		M1S3T6	
10	1.0	0:16	191	0:07	31	0:08	19	0:07	1	0:18	135	0:08	39	0:12	1	0:07	25	0:08	11	0:08	1
10	1.0	0:05	81	0:02	5	0:01	1	0:02	1	0:08	31	0:03	1	0:02	1	0:02	3	0:01	1	0:03	1
10	1.0	0:07	107	0:04	19	0:03	1	0:07	7	0:10	65	0:05	13	0:04	1	0:03	9	0:03	1	0:06	1
10	1.0	0:14	169	0:06	23	0:05	13	0:06	3	0:12	103	0:06	27	0:06	1	0:06	17	0:08	17	0:07	1
10	1.0	0:08	113	0:02	7	0:02	3	0:03	1	0:10	57	0:04	9	0:02	1	0:02	5	0:02	1	0:03	1
12	0.1	0:00	3	0:00	1	0:00	1	0:00	1	0:00	3	0:00	3	0:00	3	0:00	1	0:00	1	0:00	1
12	0.1	0:00	13	0:00	3	0:00	3	0:00	3	0:00	15	0:00	13	0:00	11	0:00	3	0:00	3	0:00	3
12	0.1	0:00	5	0:00	3	0:00	3	0:00	3	0:00	5	0:00	5	0:00	3	0:00	3	0:00	3	0:00	3
12	0.1	0:00	5	0:00	9	0:00	7	0:00	3	0:00	5	0:00	5	0:00	5	0:00	3	0:00	3	0:00	3
12	0.1	0:00	9	0:00	5	0:00	5	0:01	7	0:00	9	0:00	9	0:00	9	0:00	5	0:00	5	0:00	5
12	0.2	0:00	25	0:01	7	0:00	5	0:01	5	0:00	19	0:01	23	0:01	15	0:01	5	0:01	3	0:01	1
12	0.2	0:00	15	0:00	5	0:00	3	0:00	3	0:00	13	0:00	13	0:00	7	0:00	1	0:00	1	0:00	1
12	0.2	0:00	19	0:00	7	0:00	3	0:00	3	0:00	11	0:00	11	0:00	7	0:01	5	0:01	5	0:01	1
12	0.2	0:01	27	0:03	21	0:03	11	0:03	5	0:01	33	0:01	23	0:01	23	0:02	13	0:02	11	0:03	3
12	0.2	0:01	35	0:01	11	0:01	11	0:01	7	0:01	39	0:01	31	0:01	31	0:01	11	0:01	9	0:02	7
12	0.2	0:11	169	0:15	37	0:15	39	0:12	17	0:11	89	0:07	41	0:03	9	0:16	29	0:25	41	0:21	21
14	0.1	0:01	37	0:05	23	0:02	19	0:02	23	0:01	37	0:01	31	0:01	27	0:01	21	0:02	25	0:02	17
14	0.1	0:00	21	0:00	7	0:00	7	0:00	7	0:00	15	0:00	15	0:00	13	0:00	7	0:00	7	0:00	7
14	0.1	0:00	19	0:01	17	0:02	19	0:06	11	0:00	17	0:00	15	0:01	19	0:01	7	0:01	7	0:01	7
14	0.1	0:01	25	0:01	13	0:01	13	0:01	13	0:00	17	0:01	17	0:01	23	0:01	9	0:01	13	0:01	9
14	0.1	0:00	13	0:00	7	0:00	7	0:00	7	0:00	11	0:00	11	0:00	7	0:01	9	0:01	9	0:01	7
16	0.1	0:04	71	0:23	25	0:24	29	0:19	25	0:04	61	0:05	59	0:05	59	0:14	19	0:18	19	0:16	17
16	0.1	0:03	49	0:04	9	0:04	11	0:05	15	0:02	49	0:03	35	0:03	25	0:09	13	0:08	13	0:06	13
16	0.1	0:02	41	0:03	9	0:03	9	0:08	13	0:02	43	0:02	43	0:02	33	0:06	9	0:05	9	0:05	9
16	0.1	0:02	49	0:05	9	0:04	11	0:03	11	0:03	43	0:03	39	0:02	27	0:06	9	0:07	7	0:06	11
16	0.1	0:06	117	0:12	21	0:11	21	0:14	19	0:06	97	0:05	77	0:05	69	0:12	21	0:12	21	0:11	13
18	0.1	0:35	257	1:54	93	1:37	57	1:47	73	0:25	133	0:28	143	0:19	97	1:40	43	1:56	69	1:34	63
18	0.1	0:46	275	1:02	49	1:03	45	0:53	35	0:16	71	0:16	61	0:15	41	1:31	47	1:37	41	1:16	27
18	0.1	0:36	227	2:14	85	2:13	83	1:56	67	0:29	131	0:27	133	0:21	97	3:02	93	2:52	97	1:27	31

Table 1. Results for the maximum quadratic matching problem.

As can be expected, in practice the number of found blossom inequalities is very small, often none of them is violated, and so the basic implementation in column IP solves the problem basically via branching. The quadratic reformulation (SQK3) reduces the number of sub problems and can be further improved by target cuts. The optimal size of the chunks depends on the size of the instances. Clearly, using too large chunks can increase the total runtime, since the effect of having to solve less subproblems is foiled by the long running time needed to compute the target cuts; the latter increases exponentially in the size of the chunk. For the larger instances we considered, the best results where obtained with chunks of 5 to 6 nodes. Compared to the pure IP solution or the settings without quadratic reformulation, cut separation strongly improves the performance. In case the quadratic reformulation is additionally turned on, the effect of the cutting planes of max-cut is smaller. Usually, the number of sub problems reduces further, and sometimes the also running time improves. However, instances exist in which the running time increase, so that cut separation does not always pay off in case the quadratic reformulation is also used.

7.2 The Quadratic Linear Ordering Problem

According to our experience, separating inequalities known to be valid for the polytope P does not speed up the optimization over Q considerably, and so our basic branch-and-cut algorithm for the solution of (QLO_1) and (QLO_2) only separates the 3-dicycle inequalities. The latter are known to be facets for the linear ordering polytope. In contrast to the quadratic matching case, max-cut separation turns out to be very efficient for the quadratic linear ordering problem, and so it is called in every node of the branch-and-bound tree. The target cut separation is again performed on randomly chosen chunks that are generated via trivial projection. The vectors that are extendable under the trivial projection are again linear orderings on the chunks. We test both reformulations (SQK2) and (SQK3). Unlike for the Quadratic Matching problem, here both reformulations yield new inequalities.

We studied instances defined on complete graphs. Again, weights of linear and product variables are chosen randomly in $\{-1000, \ldots, 1000\}$ with probability p. An instance is defined by the number of nodes n of the complete graph, p, and a random seed r for the randomly chosen weights.

In Tables 2 we show running times for instances of the quadratic linear ordering problem. As above, we report the cpu time and the number of subproblems needed to solve the instance to optimality. We do not show numbers for the options M0SxTy with $x, y > 0$ in which no max-cut separation is done but a reformulation and target cut separation, as according to our experience cut separation is fast and favorable.

Moreover, we created linear arrangement instances defined by random graphs, see Section 6.3, and show the performance of the solution algorithms in Table 3. In order to show the effectiveness of our methods, we report for the bigger instances only the results with reformulation, max-cut separation and target cuts turned on.

As for the quadratic matching case, the basic implementation solves the problem essentially via branching. Only small instances can be solved to optimality. In case of failure, the gap between upper and lower bounds is often quite large. The reformulation (SQK2) reduces the number of sub problems however is usually outperformed by (SQK3) which is also reflected in our computational results. Again the separation of inequalities from the cut polytope considerably improves the running time. Also the target cut separation further reduces the number of subproblems and the running time. The best chunk sizes are 5 to 6. Chunk size 7 reduces the number of sub problem however often increases the running time. Best performance is usually achieved with all cutting plane generation methods turned on, with suitably chosen chunk sizes for the target cut separation. This is especially true for the linear arrangement problem.

It is not easy to compare the algorithmic performance for instances in which the pure IP solver reached the sub problem limit of 10000. As even small instances might require large numbers of sub problems in this basic approach, it may happen that the limit is reached after less running time than the other approaches need to solve the instance to optimality. However, as can be seen in the tables, there are many instances for which reaching the limit in the IP context already

$\|V\|$	p	IP		M0S2T0		M0S3T0		M1S0T0		M1S0T5		M1S0T6		M1S2T0		M1S2T5		M1S2T6		M1S3T0		M1S3T5		M1S3T6	
10	0.4	2:09	2541	3:21	1675	0:47	381	1:11	247	0:13	1	1:49	1	2:00	111	0:10	1	3:02	1	2:16	71	0:37	1	4:43	1
10	0.4	0:47	1181	0:60	611	0:21	351	0:21	147	0:05	3	0:45	1	0:41	73	0:06	1	1:53	1	1:32	83	0:15	1	1:21	1
10	0.4	2:02	2729	3:14	1917	0:53	999	0:59	297	0:11	1	1:57	1	1:42	143	0:11	1	3:13	1	4:42	117	0:24	1	4:09	1
10	0.4	3:35	5027	4:57	3163	0:23	205	0:22	117	0:08	1	1:19	1	0:50	65	0:10	1	2:18	1	3:53	95	0:31	1	3:05	1
10	0.4	1:57	3117	2:29	1685	0:37	579	0:55	363	0:07	1	1:25	1	1:48	159	0:08	1	2:13	1	1:52	89	0:24	1	2:49	1
10	0.7	17:58	9379	16:19	6057	6:49	2455	6:52	431	0:16	1	5:18	1	7:10	277	0:19	1	5:50	1	3:19	85	0:40	1	6:12	1
10	0.7	14:44	7031	11:07	3759	9:06	3087	16:53	645	0:19	1	6:05	1	16:52	393	0:19	1	6:15	1	5:28	123	0:40	1	7:09	1
10	0.7	> 15:33	> 10001	19:33	6493	3:13	1045	5:58	471	0:42	1	14:39	1	8:45	293	0:42	1	15:24	1	4:26	135	1:15	1	14:13	1
10	0.7	> 13:49	> 10001	20:44	8773	2:21	697	7:50	471	0:24	1	7:03	1	10:18	399	0:22	1	7:24	1	2:59	61	0:48	1	7:35	1
10	0.7	6:11	3357	5:36	2177	1:17	409	5:44	291	0:16	1	4:28	1	5:50	159	0:16	1	4:56	1	1:09	25	0:36	1	4:56	1
10	1.0	> 30:48	> 10001	25:29	6333	2:19	433	20:38	327	0:24	1	8:00	1	6:08	87	0:25	1	8:04	1	1:08	21	0:24	1	7:54	1
10	1.0	> 31:07	> 10001	28:37	7291	2:08	519	18:09	361	0:32	1	10:44	1	10:46	171	0:32	1	10:49	1	3:01	51	0:30	1	9:41	1
10	1.0	> 28:15	> 10001	25:44	7187	1:06	249	21:23	497	0:21	1	7:20	1	5:07	91	0:24	1	7:28	1	1:06	19	0:22	1	7:26	1
10	1.0	25:50	6205	18:35	4191	2:18	545	9:13	267	0:29	1	9:22	1	8:41	135	0:30	1	8:52	1	1:44	29	0:29	1	9:13	1
10	1.0	32:14	8271	21:45	5467	1:57	389	27:58	479	0:29	1	11:08	1	8:10	111	0:30	1	10:53	1	2:11	29	0:29	1	10:53	1
12	0.1	0:51	2531	3:12	1403	0:36	803	0:31	675	0:13	83	1:06	15	4:59	247	0:28	1	5:48	1	3:46	249	8:19	25	2:34:17	7
12	0.1	0:11	517	0:52	307	0:12	441	0:08	177	0:07	29	0:19	1	1:55	93	0:11	1	2:11	1	1:46	119	1:46	3	2:00	1
12	0.1	0:12	745	1:09	423	0:12	211	0:10	193	0:04	11	0:14	3	1:50	99	0:09	1	1:42	1	0:33	31	0:54	7	0:47	1
12	0.1	0:08	377	0:46	291	0:08	141	0:07	161	0:05	21	0:16	1	1:13	63	0:11	1	2:04	1	1:43	119	1:54	15	1:05	1
12	0.1	0:10	395	0:10	207	1:53	657	0:10	235	0:07	49	0:19	9	1:34	91	0:14	1	3:26	1	1:46	103	2:23	11	45:43	29
12	0.4	> 16:38	> 10001	> 37:11	> 10001	14:48	2405	30:16	1305	01:03	1	6:37	1	46:02	457	0:55	1	8:33	1	1:24:31	675	3:40	1	9:42	1
12	0.4	> 16:52	> 10001	0:46	93	10:04	1977	4:48	173	0:24	1	2:56	1	34:48	325	0:32	1	4:38	1	18:50	673	2:22	1	3:49	1
12	0.4	> 17:41	> 10001	0:37	83	25:53	5045	10:35	441	0:43	1	3:53	1	24:21	243	0:40	1	5:41	1	45:34	349	5:10	3	6:28	1
12	0.4	> 18:00	> 10001	> 23:40	> 10001	1:32	181	38:01	1529	4:28	9	12:19	1	1:15:12	771	1:13	1	11:25	1	1:01:15	505	9:36	13	14:02	1
12	0.4	> 17:19	> 10001	0:46	89	34:15	9225	36:39	1569	3:44	7	20:05	1	1:16:20	931	2:09	1	16:51	1	2:26:26	3383	10:09	5	25:35	1
14	0.1	7:54	7067	9:52	1183	7:19	6159	0:40	187	0:21	13	0:28	1	6:21	59	0:31	1	3:18	1	6:49	249	6:26	27	5:04	1
14	0.1	> 7:03	> 10001	> 54:14	> 10001	11:16	6961	7:40	1951	1:09	61	3:35	11	46:39	471	1:17	1	8:28	1	35:26	1205	22:42	59	3:31:18	13
14	0.1	> 6:45	> 10001	47:17	7311	2:05	1413	1:57	529	0:26	25	1:26	7	18:12	191	0:41	1	4:02	1	15:54	555	4:08	15	16:23	5

Table 2. Results for the quadratic linear ordering problem.

$\|V\|$	$\|E\|$	IP	M0S2T0	M0S3T0	M1S0T0	M1S0T5	M1S0T6	M1S2T0	M1S2T5	M1S2T6	M1S3T0	M1S3T5	M1S3T6	M1S3T7
15	10	0:00 3	0:14 37	0:00 3	0:00 1	0:00 1	0:00 1	0:53 5	0:53 1	5:20 3	0:00 1	0:00 1	0:00 1	0:00 1
15	13	0:00 11	0:17 23	0:00 11	0:00 1	0:00 3	0:00 1	0:18 1	1:30 1	10:39 1	0:00 1	0:00 1	0:01 1	0:06 1
15	16	0:03 67	0:24 59	0:03 67	0:00 1	0:00 1	0:01 1	0:24 3	0:51 1	5:02 1	0:00 1	0:01 1	0:01 1	0:20 1
15	19	0:09 239	1:49 225	0:08 239	0:01 7	0:01 7	0:05 7	0:38 3	1:27 1	10:25 1	0:02 7	0:02 7	0:06 7	0:54 1
15	22	0:05 101	0:57 101	0:05 101	0:01 7	0:01 3	0:02 3	0:21 3	0:27 1	3:47 1	0:02 9	0:01 3	0:03 3	0:32 3
15	25	1:16 1581	22:23 2867	1:11 1581	0:02 1	0:02 1	0:06 1	0:23 3	0:30 1	5:32 1	0:04 1	0:03 1	0:07 1	3:07 1
15	28	2:19 1677	19:10 1701	2:10 1677	0:06 11	0:02 1	0:10 1	3:12 15	0:44 1	8:55 1	0:09 7	0:05 1	0:16 1	4:37 1
15	31	> 8:20 > 10001	> 57:44 > 10001	> 7:49 > 10001	0:46 3	0:17 1	0:36 1	3:25 3	1:43 1	13:40 1	22:00 19	2:20 1	0:36 1	19:08 1
15	34	4:59 3293	31:20 3319	4:40 3293	0:07 1	0:04 1	0:18 1	0:52 3	0:39 1	5:56 1	0:12 1	0:08 1	0:21 1	7:59 1
15	37	> 10:09 > 10001	> 59:14 > 10001	> 9:27 > 10001	0:32 7	0:15 1	0:52 1	2:41 7	1:59 1	13:05 1	9:54 17	3:22 1	1:27 1	20:08 1
15	40	21:04 9239	1:44:40 9435	19:38 9239	0:29 1	0:07 1	0:36 1	1:48 3	0:45 1	6:03 1	0:48 1	0:16 1	0:51 1	13:43 1
15	43	14:14 6463	1:05:51 6619	13:17 6463	0:08 3	0:04 1	0:22 1	0:41 3	0:27 1	3:11 1	0:17 3	0:11 1	0:28 1	11:39 1
15	46	> 13:41 > 10001	> 56:55 > 10001	> 12:44 > 10001	0:41 19	0:13 1	1:00 1	4:56 15	1:05 1	7:09 1	1:05 7	0:55 1	2:54 1	31:21 1
15	49	> 14:36 > 10001	> 56:23 > 10001	> 13:33 > 10001	2:43 11	0:22 3	1:30 1	5:39 19	1:08 1	5:50 1	15:35 19	1:10 1	3:16 1	29:17 1
20	19	0:23 139	10:36 249	0:22 139	0:02 9	0:01 3	0:03 3	25:16 61	22:00 3	2:09:22 7	0:04 9	0:04 3	0:06 3	0:27 5
20	22	0:15 83	7:36 203	0:15 83	0:01 1	0:01 1	0:02 1	18:25 21	7:56 3	44:10 1	0:02 1	0:03 1	0:03 1	0:14 1
20	25	0:33 141	20:12 431	0:32 141	0:02 1	0:02 1	0:03 1	4:29 5	16:21 3	1:14:49 3	0:05 1	0:05 1	0:07 1	0:44 1
20	28	5:04 1563		4:48 1563							0:08 1	0:06 1	0:11 1	1:06 1
20	31	8:16 2187		0:38 3								0:29 1	0:29 1	3:22 1
20	34	26:30 7503		26:37 7503								0:22 3	0:38 1	3:43 1
20	37	> 30:59 > 10001		> 31:07 > 10001								2:07 1	1:08 3	10:01 1
20	40	> 31:21 > 10001		> 31:36 > 10001								1:20 1	1:19 1	10:27 1
20	43	> 32:01 > 10001		> 32:00 > 10001								15:38 1	11:15 1	44:31 1
20	46	> 34:38 > 10001		> 34:30 > 10001								57:38 1	15:23 1	49:42 1
20	49	> 36:28 > 10001		> 36:33 > 10001								26:33 1	19:28 1	1:15:09 1
20	52	> 39:19 > 10001		> 39:26 > 10001								19:27 1	10:49 1	44:43 1
20	55	> 41:50 > 10001		> 42:22 > 10001								1:41:42 33	1:31:26 11	6:58:30 3
20	58	> 45:13 > 10001		> 45:30 > 10001								1:42:03 3	29:53 1	3:53:27 1
20	61	> 47:22 > 10001		> 47:07 > 10001								27:18 1	23:57 1	1:34:28 1
20	64	> 50:40 > 10001		> 51:05 > 10001								2:07:28 9	2:14:23 3	11:09:2 2 3
25	25			2:35 475								0:29 1	0:17 1	1:00 1
25	28			7:18 1219								0:30 1	0:25 1	1:03 1
25	31			19:39 2921								0:41 3	0:45 3	1:42 3
25	34			9:31 1097								0:33 1	0:32 1	1:54 1
25	37			> 1:11:21 > 10001								2:32 1	3:56 1	8:48 1
25	40			> 1:16:36 > 10001								1:42:44 9	1:03:53 11	3:42:43 3
25	43			> 1:20:59 > 10001								12:04 1	7:00 1	12:37 1
25	46			> 1:26:12 > 10001								47:01 3	5:59 1	29:43 1
25	49			> 1:30:46 > 10001								7:45 3	7:45 3	21:54 1
25	52			> 1:36:08 > 10001								3:19:29 9	2:21:00 3	8:44:41 1

Table 3. Results for the linear arrangement problem.

takes longer than solving the instance to optimality when using our methods. This shows that the proposed methods considerably speed up the solution time.

7.3 The Quadratic Assignment Problem

As mentioned in the introduction, also well-studied quadratic problems like the QAP can be addressed by our basic toolbox. Clearly, using such general methods we cannot compete with problem-specific approaches to such problems. Nevertheless, it is interesting to study whether these methods can improve basic IP-based solution algorithms for the QAP. We generated instances on $|V|$ nodes in which the cost of a product is again randomly chosen between $\{-1000, \ldots, 1000\}$. All product variables are modeled. We show the corresponding results in Table 4.

Clearly, if no reformulation is used, target cuts considerably improve the performance of the solution algorithms. Furthermore, reformulation (SQK3) is very strong and drastically reduces the number of sub problems and improves the running time when compared to the basic IP approach. If the target cut separation is turned on additionally, the effect is not very strong. For some instances, the performance improves even further, for some other instances the running time increases slightly.

8 Generalization

In this paper, we focused on quadratic objective functions. However, the proposed techniques can be generalized to polynomial or even arbitrary non-linear objective functions. In practice, this is feasible if each non-linear expression in the objective function is defined locally, i.e., determined by few original variables. Here we give a short description of these generalizations.

First, consider the separation of cutting planes derived from max-cut, as described in Section 3. To generalize this approach, we propose to use the results presented in [6]; for the polynomial case, see also [5]. For this, a few additional variables representing partial non-linear terms have to be introduced. More precisely, for a non-linear term involving d original variables, e.g., a monomial of degree d, we have to introduce up to $d-1$ new variables. This means that the number of additional variables remains small if each term is defined locally. After this extension of the model, the separation of cutting planes concerning the non-linear structure of the objective function reduces to the separation problem for an auxiliary max-cut problem. For details of the construction, we refer to [5, 6]. In the case of a quadratic objective function, this approach agrees with the separation methods proposed in Section 3.

Second, the target cut separation discussed in Section 4 can be easily generalized to arbitrary non-linear terms. Having chosen a chunk defined by a set of original variables, we can include any non-linear term that is determined by these variables. Hence, the more locally a term is defined, the more likely it is that all necessary original variables are contained in the chunk, so that the term can be included. Observe that a non-linear term included in this way does not increase the number of feasible solutions on the chunk, i.e., the number of vertices of $\overline{Q}$. Moreover, note that if some non-linear term is defined on a given chunk, then the same is true for all variables corresponding to its partial terms. In other words, the new variables necessary for applying the results of [5, 6] do not increase the complexity for target cut separation.

For the quadratic reformulation discussed in Section 5, it is not clear whether a generalization is useful, since in general only few variables representing linearized products of two original variables might be present in the model. We expect that the answer to this question is problem-specific. An implementation and experimental evaluation of all three generalizations proposed in this section is left as future work.

9 Conclusion

We present and evaluate two methods for improving the performance of branch-and-cut approaches to quadratic 0–1 optimization problems, addressing the problem from two different directions. The

$\|V\|$	IP	M0S3T0	M1S0T0	M1S0T5	M1S0T6	M1S0T7	M1S3T0	M1S3T5	M1S3T6	M1S3T7
6	0:09 369	0:00 1	0:09 263	0:10 355	0:17 7	0:11 5	0:00 1	0:00 1	0:00 1	0:00 1
6	0:09 261	0:00 1	0:09 261	0:09 255	0:19 9	0:14 9	0:00 1	0:00 1	0:01 1	0:00 1
6	0:05 191	0:00 1	0:06 191	0:06 191	0:12 7	0:12 11	0:00 1	0:00 1	0:00 1	0:00 1
6	0:06 197	0:00 1	0:06 197	0:06 209	0:13 5	0:13 5	0:00 1	0:00 1	0:01 1	0:00 1
6	0:08 233	0:00 1	0:08 229	0:08 239	0:12 5	0:11 7	0:00 1	0:00 1	0:01 1	0:00 1
8	23:10 3833	0:41 19	23:11 3833	24:11 3679	11:03 1319	11:02 1319	0:41 15	0:41 15	0:56 9	0:56 9
8	29:52 4343	0:37 17	31:07 4353	30:58 4221	19:21 2137	19:19 2137	0:38 15	0:34 13	0:48 9	0:48 9
8	25:17 3605	0:32 13	25:20 3605	24:51 4001	13:08 1673	13:07 1673	0:34 13	0:50 19	0:46 7	0:46 7
8	10:39 2171	0:05 1	10:50 2181	10:52 2011	4:59 575	4:56 575	0:05 1	0:05 1	0:06 1	0:06 1
8	27:20 4205	0:34 15	27:45 4251	28:01 3951	14:57 1753	14:55 1753	0:31 11	0:30 11	0:48 7	0:48 7
10		24:52 83	> 261:14 > 10001	> 264:08 > 10001	> 280:14 > 10001		37:32 117	34:04 107	29:24 75	34:18 85
							15:19 43	22:09 77	20:39 63	18:55 49
							31:17 85	36:08 105	29:31 71	30:39 61
							21:54 71	17:47 47	21:08 49	23:25 75
							16:39 65	17:07 65	13:35 33	15:04 43

Table 4. Results for quadratic assignment instances.

first method addresses the quadratic structure, exploiting separation routines for cut polytopes, while the second implicitly takes into account the specific structure of the underlying polytope, applying a technique similar to local cut generation. Furthermore, we evaluate the quadratic reformulation procedures introduced by Helmberg, Rendl, and Weismantel for several different applications. Our results show that the total running time can be decreased significantly by a suitable combination of these techniques.

References

1. A. Applegate, R. Bixby, V. Chvátal, and W. Cook. TSP cuts which do not conform to the template paradigm. In *Computational Combinatorial Optimization: Optimal or Provably Near-Optimal Solutions*, volume 2241 of *Lecture Notes in Computer Science*, pages 261–304. Springer Verlag, 2001.
2. F. Barahona and A.R. Mahjoub. On the cut polytope. *Mathematical Programming*, 36:157–173, 1986.
3. C. Buchheim, F. Liers, and M. Oswald. A basic toolbox for constrained quadratic 0/1 optimization. In C. C. McGeoch, editor, *WEA 2008: Workshop on Experimental Algorithms*, volume 5038 of *LNCS*, pages 249–262. Springer Verlag, 2008.
4. C. Buchheim, F. Liers, and M. Oswald. Local cuts revisited. *Operations Research Letters*, 36:430–433, 2008.
5. C. Buchheim and G. Rinaldi. Efficient reduction of polynomial zero-one optimization to the quadratic case. *SIAM Journal on Optimization*, 18(4):1398–1413, 2007.
6. C. Buchheim and G. Rinaldi. Terse integer linear programs for boolean optimization. Technical Report zaik2007-558, Computer Science Department, University of Cologne, 2007.
7. C. Buchheim, A. Wiegele, and L. Zheng. Exact algorithms for the quadratic linear ordering problem. Technical Report zaik2007-560, Computer Science Department, University of Cologne, 2007.
8. C. De Simone. The cut polytope and the Boolean quadric polytope. *Discrete Mathematics*, 79:71–75, 1990.
9. M. Deza and M. Laurent. *Geometry of Cuts and Metrics*, volume 15 of *Algorithms and Combinatorics*. Springer Verlag, 1997.
10. C. Helmberg, F. Rendl, and R. Weismantel. A semidefinite programming approach to the quadratic knapsack problem. *Journal of Combinatorial Optimization*, 4(2):197–215, 2000.
11. T. A. Johnson. *New Linear-Programming Based Solution Procedures for the Quadratic Assignment Problem*. PhD thesis, Graduate School of Clemson University, 1992.
12. F. Liers, M. Jünger, G. Reinelt, and G. Rinaldi. *Computing Exact Ground States of Hard Ising Spin Glass Problems by Branch-and-Cut*, pages 47–68. New Optimization Algorithms in Physics. Wiley-VCH, 2004.
13. F. Rendl, G. Rinaldi, and A. Wiegele. A branch and bound algorithm for Max-Cut based on combining semidefinite and polyhedral relaxations. In *Integer programming and combinatorial optimization*, volume 4513 of *Lecture Notes in Computer Science*, pages 295–309. Springer Verlag, Berlin, 2007.