Quadratic Outer Approximation for Convex Integer Programming

Christoph Buchheim and Long Trieu

Fakultät für Mathematik, Technische Universität Dortmund
{christoph.buchheim,long.trieu}@math.tu-dortmund.de

Abstract. We present a quadratic outer approximation scheme for solving general convex integer programs, where suitable quadratic approximations are used to underestimate the objective function instead of classical linear approximations. As a resulting surrogate problem we consider the problem of minimizing a function given as the maximum of finitely many convex quadratic functions having the same Hessian matrix. A fast algorithm for minimizing such functions over integer variables is presented. Our algorithm is based on a fast branch-and-bound approach for convex quadratic integer programming proposed by Buchheim, Caprara and Lodi [5]. The main feature of the latter approach consists in a fast incremental computation of continuous global minima, which are used as lower bounds. We generalize this idea to the case of k convex quadratic functions, implicitly reducing the problem to $2^k - 1$ convex quadratic integer programs. Each node of the branch-and-bound algorithm can be processed in $O(2^k n)$ time. Experimental results for a class of convex integer problems with exponential objective functions are presented. Compared with Bonmin's outer approximation algorithm B-OA and branch-and-bound algorithm B-BB, running times for both ternary and unbounded instances turn out to be very competitive.

1 Introduction

Many optimization problems arising in real world applications can be formulated as convex mixed-integer nonlinear programs (MINLP) of the form

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & g_j(x) \leq 0 \ \forall j = 1, \dots, m \\ & x_i \in \mathbb{Z} \ \forall i \in \mathcal{I} \,, \end{aligned}$$

where $f, g_1, \dots, g_m : \mathbb{R}^n \to \mathbb{R}$ are convex functions and $\mathcal{I} \subseteq \{1, \dots, n\}$ is the set of indices of integer variables. Allowing both integrality and nonlinearity makes this class of problems extremely hard. In fact, MINLP comprises the NP-hard subclasses of mixed-integer linear programming (MILP) and general nonlinear programming (NLP). The restriction to convex MINLP preserves NP-hardness, as MILP is still contained as a special case. The most important exact approaches applied to convex MINLP are branch-and-bound by Dakin [7], generalized Benders decomposition by Geoffrion [10], outer approximation by Duran

and Grossmann [8] and Fletcher and Leyffer [9], branch-and-cut by Quesada and Grossmann [12], and the extended cutting plane method by Westerlund and Pettersson [15]. A detailed survey of algorithms and software for solving convex MINLP is given by Bonami, Kilinç and Linderoth [3].

In the further course of this paper, for simplicity, we focus on the pure integer case with box-constraints, thus assuming $\mathcal{I} = \{1, \dots n\}$ and $l \leq x \leq u$ for some fixed lower and upper bound vectors $l \in (\mathbb{R} \cup \{-\infty\})^n$ and $u \in (\mathbb{R} \cup \{\infty\})^n$. Note that the resulting box-constrained convex integer nonlinear program (INLP) of the form

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & x \in \mathcal{B}\,, \end{array} \tag{1}$$

where $\mathcal{B} := \{x \in \mathbb{Z}^n \mid l \leq x \leq u\}$, still belongs to the class of NP-hard problems, since minimizing a convex quadratic function over the integers is equivalent to the Closest Vector Problem, which is known to be NP-hard [14]. The algorithm for solving convex INLP presented in the following is based on the outer approximation scheme.

1.1 Organization of the Paper

After giving a short recapitulation of the standard linear outer approximation scheme, we describe our quadratic outer approximation scheme in Section 2. Section 3 presents our approach for solving a convex piecewise quadratic integer program with constant Hessian matrix, which occurs as a surrogate problem in every iteration of our extended outer approximation scheme. In Section 4, we present computational results and compare the effectiveness of the proposed algorithm, applied to a special class of convex integer nonlinear optimization problems, with existing state-of-the-art software. Finally, we summarize our main results and give a short outlook in Section 5.

2 Outer Approximation

2.1 Linear Outer Approximation

The main idea of the classical linear outer approximation approach is to equivalently transform the original integer nonlinear problem into an integer linear problem by iteratively adding linearizations to the objective function. As soon as we obtain an iterate that has already been computed in an earlier iteration, we have reached optimality. Applied to (1), we get the simplified scheme described in Algorithm 1. The main computational effort of the presented approach lies in the computation of the integer minimizer in Step 3 of each iteration, by solving a box-constrained convex piecewise linear integer program, which can be formulated as an integer linear program (ILP). The approach is illustrated in Fig. 1.

Algorithm 1: Linear Outer Approximation Scheme

input : convex and continuously differentiable function f
output: integer minimizer $x^* \in \mathcal{B}$ of f

1. set $k := 1$ and choose any $x^1 \in \mathcal{B}$
2. compute supporting hyperplane for f in x^k:

$$f(x) \geq f(x^k) + \nabla f(x^k)^\top (x - x^k)$$

3. compute $x^{k+1} \in \mathcal{B}$ as an integer minimizer of

$$\max_{i=1,\dots,k} \{f(x^i) + \nabla f(x^i)^\top (x - x^i)\}$$

4. if $x^{k+1} \neq x^i$ for all $i \leq k$, set $k := k + 1$ and go to 2, otherwise, x^{k+1} is optimal

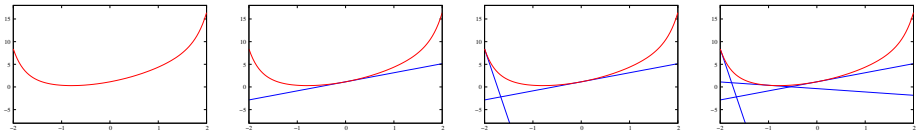

Fig. 1. Linear Outer Approximation applied to $f\colon [-2, 2] \to \mathbb{R}$, $f(x) = (x+1)^2 + e^{x^2-2}$. Iterates are $x^1 = 0$, $x^2 = -2$, $x^3 = -1$, and $x^4 = -1$

2.2 Quadratic Outer Approximation

The main drawback of Linear Outer Approximation is that in general many iterations are necessary to obtain an appropriate approximation of the original objective function. The basic idea of our approach is to modify Step 2 of Algorithm 1 by replacing the linearizations by appropriate quadratic underestimators. Unfortunately, the second-order Taylor approximation is not necessarily a global underestimator of the original function. One important challenge therefore is to find a suitable quadratic underestimator. The following observation gives a sufficient condition for an underestimator to be feasible.

Theorem 1. *Let f be twice continuously differentiable and let $Q \in \mathbb{R}^{n\times n}$ such that $Q \preccurlyeq \nabla^2 f(x)$ for all $x \in \mathbb{R}^n$. Consider the supporting quadratic function*

$$T(x) := f(x^k) + \nabla f(x^k)^\top (x - x^k) + \tfrac{1}{2} (x - x^k)^\top Q (x - x^k).$$

Then $f(x) \geq T(x)$ for all $x \in \mathbb{R}^n$.

Proof. By construction, we have $\nabla^2(f - T)(x) = \nabla^2 f(x) - Q \succcurlyeq 0$ for all $x \in \mathbb{R}^n$ and $\nabla(f - T)(x^k) = \nabla f(x^k) - \nabla f(x^k) = 0$. This implies that $f - T$ is convex with minimizer x^k, yielding $f(x) - T(x) \geq f(x^k) - T(x^k) = 0$ for all $x \in \mathbb{R}^n$. □

The entire quadratic outer approximation scheme is described in Algorithm 2.

Algorithm 2: Quadratic Outer Approximation Scheme

input : convex and twice continuously differentiable function f, matrix Q s.t. $0 \preccurlyeq Q \preccurlyeq \nabla^2 f(x)$ for all $x \in \mathbb{R}^n$
output: integer minimizer $x^* \in \mathcal{B}$ of f

1. set $k := 1$ and choose any $x^1 \in \mathcal{B}$
2. compute supporting quadratic underestimator for f in x^k:

$$f(x) \geq f(x^k) + \nabla f(x^k)^\top (x - x^k) + \tfrac{1}{2} (x - x^k)^\top Q (x - x^k)$$

3. compute $x^{k+1} \in \mathcal{B}$ as an integer minimizer of

$$\max_{i=1,\dots,k} \{f(x^i) + \nabla f(x^i)^\top (x - x^i) + \tfrac{1}{2} (x - x^i)^\top Q (x - x^i)\}$$

4. if $x^{k+1} \neq x_i$ for all $i \leq k$, set $k := k + 1$ and go to 2, otherwise, x^{k+1} is optimal

In Step 2 of Algorithm 2, the new underestimator for a given iterate x^k is computed as follows:

$$\begin{aligned} & f(x^k) + \nabla f(x^k)^\top (x - x^k) + \tfrac{1}{2} (x - x^k)^\top Q (x - x^k) \\ = & \underbrace{f(x^k) - \nabla f(x^k)^\top x^k + \tfrac{1}{2} {x^k}^\top Q x^k}_{=:c_{k+1}} + \underbrace{(\nabla f(x^k)^\top - {x^k}^\top Q)}_{=:L_{k+1}^\top} x + \tfrac{1}{2} x^\top Q x \,, \end{aligned}$$

the new quadratic underestimator

$$\tfrac{1}{2} x^\top Q x + L_{k+1}^\top x + c_{k+1}$$

is a convex quadratic function with Hessian $Q \succcurlyeq 0$ not depending on k.

Step 3 of Algorithm 2 requires to compute an integer minimizer of a quadratic program instead of a linear program, as was the case in Algorithm 1. Although the hardness of this surrogate problem increases from a practical point of view, this approach might pay off if the number of iterations decreases significantly with respect to linear approximation. In fact, we observed in our experiments that the number of iterations stays very small in general, even for problems in higher dimensions; see Section 4.

However, the surrogate problem of solving a convex box-constrained piecewise quadratic integer program, being the most expensive ingredient in Algorithm 2, requires an effective solution method to keep the whole algorithm fast. The surrogate problem can be formulated as an integer quadratic program (IQP), it is of the form

$$\min_{x \in \mathcal{B}} \max_{i=1,\dots,k} \big(\tfrac{1}{2} x^\top Q x + L_i^\top x + c_i \big), \tag{2}$$

where $Q \succcurlyeq 0$, $L_1, \dots, L_k \in \mathbb{R}^n$, and $c_1, \dots, c_k \in \mathbb{R}$. At this point it is crucial to underline that the Hessian Q of each quadratic function is the same, so that we

can rewrite (2) as

$$\min_{x\in\mathcal{B}} \left(\tfrac{1}{2}\, x^\top Q x + \max_{i=1,\dots,k} (L_i^\top x + c_i)\right) ,$$

which in turn can be reformulated as an integer quadratic program using a dummy variable $\alpha \in \mathbb{R}$:

$$\begin{aligned} \min \quad & \tfrac{1}{2}\, x^\top Q x + \alpha \\ \text{s.t.} \quad & L_i^\top x + c_i \le \alpha \quad \forall i = 1, \dots, k \\ & x \in \mathcal{B} \\ & \alpha \in \mathbb{R} \,. \end{aligned} \tag{3}$$

The quadratic outer approximation scheme is illustrated in Fig. 2. In this example, the algorithm terminates after two iterations.

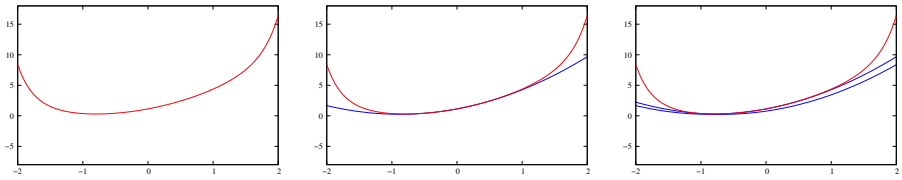

Fig. 2. Quadratic Outer Approximation for $f\colon [-2,2] \to \mathbb{R}$, $f(x) = (x+1)^2 + e^{x^2-2}$, using $Q = 2 + 2e^{-2}$. The iterates are $x^1 = 0$, $x^2 = -1$, and $x^3 = -1$

3 Convex Piecewise Quadratic Integer Programming

Solving the convex piecewise quadratic integer program (2) in each iteration is the core task of the quadratic outer approximation scheme. We implicitly reduce this problem, in iteration k, to at most $2^k - 1$ convex quadratic integer programs, which are solved by a fast branch-and-bound algorithm proposed by Buchheim, Caprara and Lodi [5]. Our computational results in Section 4 show that only few iterations of Algorithm 2 are necessary in general to solve an instance to optimality, so that the number $2^k - 1$, though being exponential, remains reasonably small in practice.

3.1 Branch-and-Bound for Convex Quadratic Integer Programming

For a better understanding of the branch-and-bound algorithm, we shortly summarize its key ingredients. For given $Q \succcurlyeq 0$, $L \in \mathbb{R}^n$, and $c \in \mathbb{R}$, consider the convex quadratic integer program

$$\min_{x\in\mathbb{Z}^n} f(x) = \tfrac{1}{2}\, x^\top Q x + L^\top x + c \,.$$

For simplicity, assume Q to be positive definite. In this case, we can easily determine the unique global minimizer $\bar{x}$ of f over $\mathbb{R}^n$ by solving a system of linear equations, as

$$\bar{x} = -Q^{-1}L \quad \text{with} \quad f(\bar{x}) = c - \tfrac{1}{2}\, L^\top Q^{-1} L,$$

which we can use as a lower bound for f over $\mathbb{Z}^n$. Simple rounding of the continuous minimizer

$$x_j := \lfloor \bar{x}_j \rceil,\ j = 1, \dots, n$$

to the next integer yields a trivial upper bound $f(x)$ for f over $\mathbb{Z}^n$.

In our branch-and-bound scheme, we branch by fixing the variables in increasing distance to their values in the continuous relaxation. By exploiting the convexity of f and its symmetry with respect to $\bar{x}$, we can cut off the current node of the tree and all its siblings as soon as we fix a variable to some value for which the resulting lower bound exceeds the current best known upper bound. Using these ingredients we get a straightforward branch-and-bound algorithm with running time $O(n^3)$ per node, mainly for computing the continuous minimizer by solving a linear system of equations. However, the running time of this computation can be improved to even linear running time per node, in two steps.

First, note that after fixing the first d variables, the problem reduces to the minimization of

$$\bar{f} \colon \mathbb{Z}^{n-d} \to \mathbb{R},\ x \mapsto \tfrac{1}{2}\, x^\top \bar{Q}_d x + \bar{L}^\top x + \bar{c}$$

where $\bar{Q}_d \succ 0$ is obtained by deleting all rows and columns of Q corresponding to fixed variables and $\bar{L}$ and $\bar{c}$ are adapted properly. The main idea is to fix the variables in a predetermined order. Following this approach, the reduced matrices $\bar{Q}_d$ only depend on the depth d, but not on specific fixings. This implies that only n different matrices $\bar{Q}_d$ appear in the entire branch-and-bound tree, so that their inverse matrices can be predetermined in a preprocessing phase. The resulting running time reduces to $O((n-d)^2)$ per node.

The second improvement is a consequence of the following observation [5]:

Theorem 2. *For each $d \in \{0, \dots, n-1\}$, there exist vectors $z^d, v^d \in \mathbb{R}^{n-d}$ and a scalar $s_d \in \mathbb{R}$, only depending on Q and the chosen order of variables, such that the following holds: if $\bar{x}^{old} \in \mathbb{R}^{n-d}$ denotes the minimizer of $\bar{f}$ after fixing variables $x_1, \dots, x_d$, and x_{d+1} is fixed to $r_{d+1} \in \mathbb{Z}$, then the resulting continuous minimizer and associated minimum can be computed incrementally by*

$$\bar{x}^{new} := \bar{x}^{old} + \alpha z^d$$

and

$$\bar{f}(\bar{x}^{new}) := \bar{f}(\bar{x}^{old}) + \alpha\big((\bar{x}^{old})^\top v^d + \bar{L}^\top z^d\big) + \alpha^2 s_d$$

where $\alpha = r_{d+1} - \bar{x}_1^{old}$.

As a conclusion, if z^d, v^d and s_d are computed in a preprocessing phase for all $d = 0, \dots, n-1$, the resulting total running time per node is $O(n-d)$.

3.2 Lower Bound Computation for the Surrogate Problem

To solve the surrogate problem (2) with a branch-and-bound algorithm, we compute a lower bound at every node by solving its continuous relaxation

$$\min_{x\in\mathbb{R}^{n-d}} \max_{i=1,\dots,k} \left(\tfrac{1}{2}\, x^T \bar{Q}_d x + \bar{L}_i^T x + \bar{c}_i\right) .$$

The main idea is to decompose this problem into subproblems, namely the minimization of several auxiliary quadratic functions defined on affine subspaces of $\mathbb{R}^{n-d}$, and finally make use of the incremental computation technique described in the last subsection. To describe this decomposition procedure, we need to introduce some definitions. First, we define

$$\bar{f}(x) := \max_{i=1,\dots,k} \bar{f}_i(x)$$

as the maximum of the reduced functions

$$\bar{f}_i(x) := \tfrac{1}{2}\, x^\top \bar{Q}_d x + \bar{L}_i^\top x + \bar{c}_i,\ i = 1,\dots,k\ .$$

For all $J \subseteq \{1,\dots,k\}$, $J \neq \emptyset$, we define

$$U_J := \{x \in \mathbb{R}^{n-d} \mid \bar{f}_i(x) = \bar{f}_j(x)\ \forall i,j \in J\}$$

and consider the auxiliary function

$$\bar{f}_J(x) : U_J \to \mathbb{R}, \quad \bar{f}_J(x) := \bar{f}_i(x),\ i \in J\ .$$

As all functions $\bar{f}_i$ have the same Hessian matrix $\bar{Q}_d$, each set U_J is an affine subspace of $\mathbb{R}^{n-d}$. In particular, we can compute the minimizers x_J^* of all $\bar{f}_J$ incrementally as described in Section 3.1.

Theorem 3. *For each $J \subseteq \{1,\dots,k\}$ with $J \neq \emptyset$, let x_J^* be a minimizer of $\bar{f}_J$ over U_J. Then the global minimum of $\bar{f}$ is*

$$\min\ \{\bar{f}_J(x_J^*) \mid \emptyset \neq J \subseteq \{1,\dots,k\},\ \bar{f}(x_J^*) = \bar{f}_J(x_J^*)\}.$$

Proof. We clearly have "$\leq$". To show "$\geq$", let $x^* \in \mathbb{R}^{n-d}$ be the global minimizer of $\bar{f}$ and define

$$J^* := \{i \mid \bar{f}_i(x^*) = \bar{f}(x^*)\} \neq \emptyset.$$

Then it follows that $x^* \in U_{J^*}$ and $\bar{f}(x^*) = \bar{f}_{J^*}(x^*)$. Moreover, x^* minimizes $\bar{f}_{J^*}$ over U_{J^*}, hence $x^* = x_{J^*}^*$ by strict convexity. In summary,

$$\bar{f}(x_{J^*}^*) = \bar{f}(x^*) = \bar{f}_{J^*}(x^*) = \bar{f}_{J^*}(x_{J^*}^*)\ ,$$

from which the result follows. □

Corollary 1. *The running time of the modified branch-and-bound algorithm for solving the surrogate problem (2) is $O(2^k \cdot (n-d))$ per node.*

Note that the index set J does not need to be considered any more in the current node and its branch-and-bound subtree as soon as the value of $\bar{f}_J(x_J^*)$ exceeds the current upper bound.

4 Experimental Results

To show the potential of our approach, we carried out two types of experiments. First, we compared our branch-and-bound algorithm for solving the surrogate problems (2), called CPQIP, to CPLEX 12.4 [1]. Second, we created a class of hard convex integer programs, to illustrate the effectiveness of our quadratic outer approximation algorithm, called QOA, and to compare its performance to that of Bonmin-OA and Bonmin-BB 1.5.1 [13] using Cbc 2.7.2 and Ipopt 3.10.

4.1 Implementation Details

We implemented our algorithm in C++. To speedup our algorithm, we used a straightforward local search heuristic to determine a good starting point. We start by taking the origin $x = (0, \dots, 0)$ and continue to increase the first variable x_1 until no further improvement can be found. In the same way we test if decreasing the variable leads to a better solution. We repeat this procedure for every consecutive variable $x_2, \dots, x_n$. The improvement in running time can be seen in Table 2. Another small improvement in running time can be achieved by using the optimal solution x_k^* of the surrogate problem in iteration k to get an improved global upper bound UB for the next iteration, i.e.

$$UB := \max_{i=1,\dots,k+1} \tfrac{1}{2}\, {x_k^*}^\top Q x_k^* + L_i^\top x_k^* + c\,.$$

4.2 Surrogate Problem

We randomly generated 160 instances for the surrogate problem (2), 10 for each combination of $n \in \{20, 30, 40, 50\}$ and $k \in \{2, \dots, 5\}$. We chose $\mathcal{B} = \mathbb{Z}^n$, i.e., we consider unbounded instances. For generating the positive semidefinite matrix Q, we chose n eigenvalues λ_i uniformly at random from $[0, 1]$ and orthonormalize n random vectors v_i, where all entries are chosen uniformly at random from $[-10, 10]$, then we set $Q = \sum_{i=1}^n \lambda_i v_i v_i^\top$. The entries of all L_i and c_i, $i = 1, \dots, k$, were chosen uniformly at random from $[-10, 10]$.

We compared our algorithm CPQIP with the Mixed-Integer Quadratic Programming (MIQP) solver of CPLEX 12.4 [1] applied to the QP model (3). For both approaches, we used a time limit of 3 hours and an absolute optimality tolerance of 10^{-6}. The relative optimality tolerance of CPLEX was set to 10^{-10}.

The results are summarized in Table 1. Running times are measured in cpu seconds and the numbers of nodes explored in the branch-and-bound trees are given in the corresponding column. First, we can observe that our algorithm could solve 158 out of 160 instances in total within the given time limit, while CPLEX 12.4 managed to solve only 154 instances. Second, for instances with a large number n of variables but small k, our approach turns out to be significantly faster. Instances of this type are the most relevant instances in a quadratic outer approximation scheme, as shown in Section 4.3.

As expected, CPQIP tends to be slower than CPLEX on most of the instances with $k = 5$, since the running time per node in our branch-and-bound algorithm is exponential in k.

		CPQIP			CPLEX 12.4		
n	k	solved	time	nodes	solved	time	nodes
20	2	10/10	0.02	1.39e+4	10/10	0.24	2.52e+3
20	3	10/10	0.03	9.84e+3	10/10	0.18	1.68e+3
20	4	10/10	0.06	9.51e+3	10/10	0.21	1.59e+3
20	5	10/10	0.05	2.91e+3	10/10	0.10	1.14e+3
30	2	10/10	0.07	4.23e+4	10/10	0.90	1.33e+4
30	3	10/10	0.40	1.09e+5	10/10	1.94	2.81e+4
30	4	10/10	4.48	4.20e+5	10/10	7.75	9.06e+4
30	5	10/10	4.57	2.63e+5	10/10	2.90	3.92e+5
40	2	10/10	10.41	5.35e+6	10/10	138.77	1.31e+6
40	3	10/10	30.70	7.18e+6	10/10	69.58	7.91e+5
40	4	10/10	66.13	7.15e+6	10/10	73.32	8.28e+5
40	5	10/10	141.53	7.21e+6	10/10	92.80	1.00e+6
50	2	10/10	320.30	1.45e+8	10/10	2337.91	1.81e+8
50	3	10/10	844.18	1.72e+8	8/10	646.15	5.41e+7
50	4	10/10	1925.06	1.79e+8	8/10	864.92	6.95e+7
50	5	8/10	1650.26	6.90e+7	8/10	931.00	7.50e+7

Table 1. Average running times, numbers of instances solved and average numbers of branch-and-bound nodes for CPQIP and CPLEX 12.4 on randomly generated unbounded instances of problem type (2)

4.3 Quadratic Outer Approximation Scheme

In order to evaluate the entire quadratic outer approximation scheme, we consider the following class of problems:

$$\min_{x\in\mathbb{Z}^n} f(x),\ \ f : D \subseteq \mathbb{R}^n \to \mathbb{R},\ \ f(x) = \sum_{i=1}^{n} \exp(q_i(x)) \tag{4}$$

where $q_i(x) = x^\top Q_i x + L_i^\top x + c_i$. We assume $Q_i \succ 0$ for all $i = 1, \dots n$, so that f is a strictly convex function.

In order to determine a feasible matrix Q according to Theorem 1, we first compute $m_i := \min_{x\in D} q_i(x)$ for all $i = 1, \dots, n$ and then choose

$$Q := \sum_{i=1}^{n} 2Q_i \exp(m_i) \succ 0\ .$$

It is easy to see that $\nabla^2 f(x) - Q \succcurlyeq 0$ for all $x \in \mathbb{R}^n$, so that Q can be used in Algorithm 2. The quality of Q and therefore of the quadratic underestimator strongly depends on D: the smaller the set D is, the larger are the m_i, and the better is the global underestimator.

To test the quadratic outer approximation scheme, we randomly generated ternary and unbounded instances of type (4), i.e., we consider both $D = [-1, 1]$ and $D = \mathbb{R}^n$. All data were generated in the same way as in Section 4.2, except

that all coefficients of Q and $L_i, c_i,\ i = 1, \dots, n$, were scaled by 10^{-5}, to avoid problems with the function evaluations.

We tested our algorithm against Bonmin-OA and Bonmin-BB [13, 2], which are state-of-the-art solvers for convex mixed-integer nonlinear programming. While Bonmin-OA is a decomposition approach based on outer approximation [8, 9], Bonmin-BB is a simple branch-and-bound algorithm based on solving a continuous nonlinear program at each node of the search tree and branching on the integer variables [11]. Again the time limit per instance was set to 3 hours.

	QOA(h)				QOA				Bonmin-OA		Bonmin-BB	
n	solved	ø it	mx	time (s)	solved	ø it	mx	time (s)	solved	time (s)	solved	time (s)
20	10/10	1.00	1	0.03	10/10	2.00	2	0.07	10/10	91.53	10/10	2.12
30	10/10	1.50	4	5.87	10/10	2.50	4	28.46	5/10	1017.99	10/10	148.85
40	10/10	1.40	3	20.19	10/10	2.40	4	65.42	0/10	—	9/10	4573.80
50	10/10	2.00	3	69.54	10/10	3.10	4	151.88	0/10	—	0/10	—
60	9/10	2.11	4	1154.66	9/10	3.00	5	1692.76	0/10	—	0/10	—
70	5/10	3.80	6	3363.11	4/10	3.75	5	2916.21	0/10	—	0/10	—

Table 2. Running times, number of instances solved, average and maximum number of iterations of QOA(h), QOA compared to Bonmin-OA and Bomin-BB for randomly generated ternary instances of problem type (4)

	QOA(h)				Bonmin-BB	
n	solved	ø it	mx	time (s)	solved	time (s)
20	10/10	2.30	3	0.12	10/10	113.13
30	8/10	3.12	5	26.38	1/10	4897.10
40	6/10	2.83	3	134.42	0/10	—
50	2/10	3.00	3	7630.12	0/10	—

Table 3. Running times, number of instances solved, average and maximum number of iterations of QOA(h) compared to Bonmin-BB for randomly generated unbounded instances of problem type (4)

Results for ternary and unbounded instances are shown in Table 2 and 3, respectively. "QOA(h)" denotes our quadratic approximation scheme using the local search heuristic, while in "QOA" the heuristic was turned off. Running times are again measured in seconds. The columns named "ø it" and "mx" show the average and maximum number of iterations in our approach, respectively.

Overall our quadratic outer approximation approach could solve more instances and seems to be considerably faster for the ternary instances as well as the unbounded instances. For the unbounded instances, we unfortunately experienced some numerical issues, because the function evaluations seem to cause

problems. While Bonmin-OA did not converge properly, the branch-and-bound algorithm Bonmin-BB, which seems to be faster than B-OA in all cases, sometimes computed solutions which were slightly worse than ours and hence not always optimal. In the ternary case, no such problems occured in any of the approaches tested.

An important observation in all our experiments is that both the average and maximum number of iterations in our outer approximation scheme tend to be small, here up to 4 for the instance sizes we could solve to optimality within the given time limit. In particular, the number of iterations does not seem to increase significantly with the number of variables n, contrary to the standard linear outer approximation approach.

5 Conclusion

We proposed a quadratic outer approximation scheme for solving convex integer nonlinear programs, based on the classical linear outer approximation scheme. From our computational results, we can conclude that quadratic underestimators have the potential to yield significantly better approximations, which might lead to considerably fewer iterations of the entire algorithm. While the standard linear outer approximation scheme requires to solve an integer linear program in each iteration, our method requires the solution of integer quadratic programs with linear constraints. Therefore we proposed an algorithm which is based on the reduction of the surrogate problems to a set of unconstrained convex quadratic integer programs, which are effectively solved by a branch-and-bound algorithm introduced by Buchheim et al. [5].

For future work it remains to study possible and good choices of Q for other classes of problems. Moreover, the running time could be further reduced by, e.g., trying to eliminate non-active underestimators or approximately solving the surrogate problem instead of solving it to optimality, in order to obtain additional quadratic underestimators quickly. Furthermore, our approach could be extended to constrained convex integer programs by using penalty functions. For this, note that a feasible Q stays feasible if adding a convex penalty function to the objective function.

Finally, one could also consider using non-convex quadratic underestimators for non-convex nonlinear integer optimization within our framework, assuming that a fast algorithm for solving non-convex quadratic integer programs is at hand; potential candidates are the algorithms proposed by Buchheim and Wiegele [4] or by Buchheim, De Santis, Palagi and Piacentini [6]. Such an extension would allow our approach to be applied to a much wider class of problems than classical linear outer approximation.

References

[1] IBM ILOG CPLEX Optimizer 12.4, 2013. URL `www.ibm.com/software/integration/optimization/cplex-optimizer`.

[2] P. Bonami, L.T. Biegler, A.R. Conn, G. Cornuéjols, I.E. Grossmann, C.D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, and A. Wächter. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization*, 5(2):186 – 204, 2008.

[3] P. Bonami, M. Kilinç, and J.T. Linderoth. *Algorithms and Software for Solving Convex Mixed Integer Nonlinear Programs*, volume 154 of *IMA Volumes in Mathematics and its Applications: Mixed Integer Nonlinear Programming*, chapter Part I: Convex MINLP, pages 1–39. Springer, 2012.

[4] C. Buchheim and A. Wiegele. Semidefinite relaxations for non-convex quadratic mixed-integer programming. *Mathematical Programming*, 2012. To appear.

[5] C. Buchheim, A. Caprara, and A. Lodi. An effective branch-and-bound algorithm for convex quadratic integer programming. *Mathematical Programming*, 135:369–395, 2012.

[6] C. Buchheim, M. De Santis, L. Palagi, and M. Piacentini. An exact algorithm for quadratic integer minimization using ellipsoidal relaxations. Technical report, Optimization Online, 2012.

[7] R.J. Dakin. A tree-search algorithm for mixed integer programming problems. *The Computer Journal*, 8:250–255, 1965.

[8] M.A. Duran and I.E. Grossmann. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming*, 36: 307–339, 1986.

[9] R. Fletcher and S. Leyffer. Solving mixed integer nonlinear programs by outer approximation. *Mathematical Programming*, 66:327–349, 1994.

[10] A. Geoffrion. Generalized Benders Decomposition. *Journal of Optimization*, 10:237–260, 1972.

[11] O.K. Gupta and A. Ravindran. Branch and bound experiments in convex nonlinear integer programming. *Management Science*, 31(12):1533–1546, 1985.

[12] I. Quesada and I.E. Grossmann. An LP/NLP based branch-and-bound algorithm for convex MINLP. *Computers and Chemical Engineering*, 16: 937–947, 1992.

[13] Bonmin 1.5.1: Basic Open source Nonlinear Mixed INteger programming, 2013. URL `www.coin-or.org/Bonmin`.

[14] P. Van Emde Boas. Another NP-complete problem and the complexity of computing short vectors in a lattice. Technical Report 81-04, University of Amsterdam, Department of Mathematics, 1981.

[15] T. Westerlund and F. Pettersson. A cutting plane method for solving convex MINLP problems. *Computers and Chemical Engineering*, 19:131–136, 1995.