COLORADOSCHOOLOFMINES
EARTH • ENERGY • ENVIRONMENT

Division of Economics and Business
Working Paper Series

Continuous Piecewise Linear δ-Approximations for MINLP Problems. II. Bivariate and Multivariate Functions

Steffen Rebennack
Josef Kallrath

Working Paper 2012-13
http://econbus.mines.edu/working-papers/wp201213.pdf

Colorado School of Mines
Division of Economics and Business
1500 Illinois Street
Golden, CO 80401

October 2012

Colorado School of Mines
Division of Economics and Business
Working Paper No. **2012-13**
October 2012

Title:
Continuous Piecewise Linear δ-Approximations for MINLP Problems. II. Bivariate and Multivariate Functions

Author(s):
Steffen Rebennack
Division of Economics and Business
Colorado School of Mines
Golden, CO 80401-1887
srebenna@mines.edu

Josef Kallrath
Department of Astronomy
University of Florida
kallrath@astro.ufl.edu

ABSTRACT

Following up on Rebennack & Kallrath (2012), in this paper, for functions depending on two variables, using refinement heuristics, we automatically construct triangulations subject to the condition that the continuous, piecewise linear approximation, under- or overestimation never deviates more than a given δ-tolerance from the original function over a given domain. This tolerance is proven by solving subproblems over each triangle to global optimality. The continuous, piecewise linear approximators, under- and overestimators involve shift variables at the vertices of the triangles leading to a small number of triangles while still ensuring continuity over the full domain. On a set of test functions, we demonstrate the numerical behavior of our approach.

For functions depending on more than two variables we provide appropriate transformations and substitutions which allow to use one- or two-dimensional δ-approximators. We address the problem of error propagation when using these dimensionality reduction routines. The automatic refinement triangulation provides an alternative to separation or transformation techniques applied to bivariate functions followed by one-dimensional piecewise linear approximation. We discuss and analyze the tradeoff between one-dimensional and two-dimensional approaches.

To demonstrate the methodology we apply it to a cutting stock problem in which we compute minimal area rectangles hosting a given number of circles; we prove optimality for one literature problem which so far had been solved only with finite gap.

Keywords: global optimization, NLP, nonconvex, overestimator, underestimator, inner approximation, outer approximation.

1 Introduction

We consider the following nonlinear (and nonconvex) optimization problem:

$$\begin{aligned} \text{NOP}: \min \quad & f(x) && (1)\\ \text{s.t.} \quad & g(x) = 0 && (2)\\ & h(x) \leq 0 && (3)\\ & x \in \mathbb{D} && (4) \end{aligned}$$

with lower and upper bounds, X_- and X_+, on x and

$$\begin{aligned} D_1: \quad & \mathbb{D} := [X_-, X_+]^n \subset \mathbb{R}^n, \text{ or}\\ D_2: \quad & \mathbb{D} := [X_-, X_+]^n \subset \mathbb{R}^{n_1} \times \{0,1\}^{n_2}, \text{ or}\\ D_3: \quad & \mathbb{D} := [X_-, X_+] \subset \mathbb{R}, \text{ or}\\ D_4: \quad & \mathbb{D} := [X_-, X_+]^2 \subset \mathbb{R}^2 \end{aligned}$$

as well as continuous functions $f : \mathbb{D} \to \mathbb{R}$, $g : \mathbb{D} \to \mathbb{R}^{m_1}$, $h : \mathbb{D} \to \mathbb{R}^{m_2}$ with dimensions $n_1 + n_2 = n$.

Just like in the univariate case (Rebennack & Kallrath, 2012, [15]), we are *not* interested in solving problem (1)-(4) to global optimality. Rather, we assume that there are global solvers available, which can solve optimization problems over each single function to global optimality. Our focus is on computing linear ε-approximations of the optimization problem (1)-(4) in the following sense:

Definition 1 (ε-approximated problem, [15]) Let x^* be a globally optimal solution to (1)-(4) and $|\cdot|$ denote the absolute value function. We call an optimization problem an *ε-approximated problem*, if for any optimal solution y^* of the ε-approximated problem, y^* satisfies the following properties:

$$\begin{aligned} P_1: \quad & |g_i(y^*)| \leq \varepsilon \quad , \qquad i = 1, \ldots, m_1\\ P_2: \quad & h_j(y^*) \leq \varepsilon \quad , \qquad j = 1, \ldots, m_2\\ P_3: \quad & |f(x^*) - f(y^*)| \leq \varepsilon \quad . \end{aligned}$$

We propose the derivation of an ε-approximated problem by careful construction of piecewise linear δ-approximators, δ-underestimator, and δ-overestimators; *i.e.*, δ-approximators never deviate more than δ from the nonlinear function (*cf.* Rebennack & Kallrath, 2012, [15, Definitions 2 & 3]). When choosing the δ values carefully, one can replace NOP by a MILP problem and obtain valid lower bounds (for minimization problems); for the choice of δ please see Rebennack & Kallrath (2012, [15, Section 3.3]). The size of the resulting MILP problem depends crucially on the number of segments introduced by the piecewise linear δ-approximators. Thus, two aspects are important: (1) the computed δ-approximators should contain few segments and (2) a given tolerance δ should not be exceeded.

Detecting infeasibilites in NOP might also be of great practical interest, we refer the reader again to Rebennack & Kallrath (2012, [15]).

In this paper, we extend the ideas for univariate functions, as discussed in Rebennack & Kallrath (2012, [15]), to higher dimensional problems (with a focus on D_4).

In particular, we construct good δ-approximations to nonlinear functions by piecewise linear functions. While in the case of univariate functions this involves appropriate systems of breakpoints, their convex combination and connecting lines between points of the function graphs, in higher dimensions the lines are triangles, tetrahedra, or in general simplices. The principle of convex combination remains the same.
The contributions of this paper are threefold:

1. We develop algorithms to automatically compute triangulations and the construction of continuous, piecewise linear functions over such systems of triangles which approximate nonlinear, convex or nonconvex, functions in two variables to δ accuracy.
2. We classify a rich class of *n*-dimensional functions which can be separated into lower dimensional functions. This enables us to apply approximation techniques developed for univariate and bivariate functions. In addition, the approximation error of the *n*-dimensional functions is expressed in the lower dimensional transformations.
3. We demonstrate both the one- and two-dimensional approximation techniques with a nonlinear cutting stock problem.

In this paper we follow up on the setting of paper I (Rebennack & Kallrath, 2012, [15]) to construct δ-accurate piecewise linear approximators, over- and underestimators for two-dimensional functions (Section 3) based on automatic triangulations. Higher dimensions and higher dimensional functions are treated in Section 4, where we also outline the approximation of NLPs and MINLPs. Section 5 provides numerical results. We illustrate the usage of estimator functions on a two-dimensional circle cutting problem in Section 6. A series of future research directions is presented in Section 7. In Section 8 we conclude the paper and discuss the methodology and its limits.

2 Literature Review

A recent publication by Geißler et al. (2012, [6]) and slightly earlier the dissertation by Geißler (2011,[5]) come in some parts close to our ideas but differ in the following aspects. Their automatic, incremental triangulation produces Delaunay triangulations but does not involve shift variables at the vertices of the triangles. Our approach is more general in this aspect because it can handle arbitrary, indefinite functions regardless of their curvature. Our only requirement is that the functions have a finite number of discontinuities over a compactum (*e.g.*, no singularities).

Instead of reviewing a rich body of literature related to piecewise linear approximation of functions in one, two or more dimensions, we refer the reader to the literature reviews contained in recent publications by Misener & Floudas (2010, [13]) who presented explicit, piecewise linear formulations of two- or three-dimensional functions based on simplices, Linderoth (2005,[12]) who uses triangulations in the solution process of quadratically constrained problems, Vielma & Nemhauser (2011, [19]) who developed a formulation which provides an efficient alternative to standard special ordered sets of type 2 (SOS-2) as their models grow only logarithmically in the number of binary variables, D'Ambrosio et al. (2010, [3]) who compare

different formulations (one-dimensional, rectangle, triangle) to approximate for two-dimensional functions and discuss the notion of special ordered sets of type 3 (SOS-3), and Rebennack & Kallrath (2012, [15]) who are the first to compute optimal breakpoint systems for univariate functions based on global optimization techniques.

The recent invention of modified SOS-2 conditions or formulations using significantly fewer binary variables growing only logarithmically in the number of support areas (breakpoints, triangles, or simplices) by Vielma & Nemhauser (2011, [19]) relieves somewhat the pressure to seek for a minimum number of support areas involved in the linear approximation of functions. Their approach has been used, for instance, by Misener & Floudas (2010, [13]) or Geißler et al. (2012, [6]). We have also applied the Vielma & Nemhauser formulation in Section 6.5 but it is only a side track of the numerical experiments; the focus in this paper is rather on the construction of δ-approximators, over- and underestimators.

3 Bivariate Functions

In the one-dimensional case, we constructed convex linear combinations of support areas which were breakpoint-limited disjunct intervals covering the region (master intervals) we were interested in. In the two-dimensional case, we consider rectangular regions to start with (*e.g.*, resulting from lower and upper bounds on the two decision variables). We are seeking support areas which cover the rectangle, which can be easily made larger or smaller reflecting the curvature of the function we want to approximate. Similar to the one-dimensional case, we use interpolation techniques to construct the δ-approximators. In two dimensions, we require (at least) three points to construct such interpolation planes.

While functions depending on two or more variables are in the most general case treated by equal-sized simplices leading to direct SOS-2 representation, *cf.* Misener and Floudas (2010, [13]), our approach is somewhat different: We use different-sized triangles, select the triangle hosting a point (x_1, x_2), and represent (x_1, x_2) and its function value as convex linear combinations of the argument and function value at the vertices of the triangle. We have selected triangles for their simplicity, and we have chosen different-sized objects to adjust better to the function and its variations.

In the following, we discuss the piecewise linear approximation of a function f over a triangle in Section 3.1. We present in Section 3.2 efficient algorithms to compute δ-approximators via triangulations while we discuss the case for δ-under- and overestimators in Section 3.3. MILP formulations derived from triangulations to approximate f are outlined in Section 3.4.

3.1 Function Approximation over Triangles

Consider a triangle $\mathscr{T}_1 \subset \mathbb{R}^2$ in the x_1-x_2-plane established by three points (vertices) $P_j = (X_{1j}, X_{2j}) \in \mathbb{R}^2$, $j = 1,2,3$. We assume that at most two of them are colinear, *i.e.*, all three of them never lie on the same line. Each point $p = (x_1, x_2) \in \mathscr{T}_1$ can be

represented as a convex combination of these three points, *i.e.*,

$$p = \sum_{j=1}^{3} P_j \lambda_j \qquad \text{with } \lambda_j \geq 0 \text{ and } \qquad \sum_{j=1}^{3} \lambda_j = 1 \quad .$$

Let $f(p) = f(x_1, x_2)$ be a real-valued function in two arguments x_1 and x_2 defined over a rectangle $D_4 := [X_1^-, X_1^+] \times [X_2^-, X_2^+] \supset \mathscr{T}_1$. We can construct a linear approximation $\ell(p)$ of f over $\mathscr{T}_1$ by a convex combination of the function values, $f_j = f(P_j) = f(x_{1j}, x_{2j})$, at the points p:

$$\ell(p) = \sum_{j=1}^{3} \lambda_j f_j \quad .$$

3.2 Constructing the Triangulation

Our goal is now to construct a triangulation of D_4 by a set $\mathscr{T}$ of triangles $\mathscr{T}_t$, with $\bigcup_{\mathscr{T}_t \in \mathscr{T}} \mathscr{T}_t \supseteq D_4$, where the triangles should have maximal size (area) leading to a minimal number of triangles subject to the constraint

$$\Delta_t := \max_{p \in \mathscr{T}_t} |\ell(p) - f(p)| \leq \delta \quad , \quad \forall t \in \mathscr{T} \quad . \tag{5}$$

Note that for a given triangle, Δ_t is computed by solving problem (5) to global optimality.

Therefore, let us discuss whether we can transfer some ideas used in the univariate case to the bivariate case. The construction of triangulations with a minimum number of triangles is not easy and straightforward. The *direct approach*, in which we would proceed similarly as in the one-dimensional case by allowing a fixed number of breakpoints $p_b := (x_b, y_b)$ distributed in plane D_4 raises severe problems:

1. How to ensure that the breakpoints p_b generate non-overlapping triangles fully covering the rectangle?
2. How to ensure continuity in the vertices? If we know that two triangles $\mathscr{T}_1$ and $\mathscr{T}_2$ share edge e_{12}, then we have to apply the continuity constraints to the two shared vertices $\mathbf{v}^v_{e_{12}\mathscr{T}_1} = \mathbf{v}^v_{e_{12}\mathscr{T}_2}$ with $v = 1,2$ belonging to edge e_{12}, *i.e.*, $f(\mathbf{v}^v_{e_{12}\mathscr{T}_1}) + s(\mathbf{v}^v_{e_{12}\mathscr{T}_1}) = f(\mathbf{v}^v_{e_{12}\mathscr{T}_2}) + s(\mathbf{v}^v_{e_{12}\mathscr{T}_2})$ for $v = 1,2$, where $s(\mathbf{v})$ is the shift at vertex $\mathbf{v}$.

Using a *marching scheme* (the analogue situation to the moving breakpoint approach, *cf.* Rebennack & Kallrath (2012, [15], Section 4.3)) as in the one-dimensional case would lead to complications in constructing an irregular grid of triangles, or to a regular grid with too many triangles: in the case of an irregular grid, similar to the *direct approach*, the difficulty lies in generating non-overlapping triangles fully covering the rectangular, while a regular grid does not exploit the fact that in regions with small function curvature we do not need so many triangles. Therefore, we proceed differently and use a triangle refinement approach.

To begin with, we divide D_4 by one of the diagonals into two triangles $\mathscr{T}_1$ and $\mathscr{T}_2$. Then, for a given triangle $\mathscr{T}_t$ with vertices $[v_{t1}, v_{t2}, v_{t3}]$ and fixed shift variables $[s_{t1}, s_{t2}, s_{t3}]$, we solve the following (potentially nonconvex) NLP:

$$\begin{aligned}
\Delta_t := \ & \max |\ell(p_t) - f(p_t)| && (6)\\
\text{s.t.} \ & p_t = \textstyle\sum_{j=1}^{3} \lambda_j v_{tj} && (7)\\
& \textstyle\sum_{j=1}^{3} \lambda_j = 1 && (8)\\
& \ell(p_t) := \textstyle\sum_{j=1}^{3} \lambda_j \phi(v_{ti}) && (9)\\
& \lambda_j \in [0,1], \quad p_t \in \mathscr{T}_t, \quad j = 1,2,3 \quad , && (10)
\end{aligned}$$

with $\phi(\cdot)$ defined as

$$\phi(v_{tj}) = f(v_{tj}) + s_{tj} \quad , \tag{11}$$

for vertices v_{tj} contained in triangle $\mathscr{T}_t$; equation (11) is the analogon to equation (24) in Rebennack & Kallrath (2012, [15]).

If $\Delta_t \leq \frac{\delta}{2}$, then we keep the triangle $\mathscr{T}_t$ along with the shift variables s_{tj}; *i.e.*, we add $\mathscr{T}_t$ to $\mathscr{T}$. Otherwise, we try a different value for the shift variables s_{tj} as follows

$$s_{tjd} := \left(\frac{2d}{D+1} - 1 \right) \frac{\delta}{2} \tag{12}$$

for some $D \in \mathbb{N}$. Care has to be taken to identify shift variables which have been fixed at vertices of other triangles which equal one of the vertices of triangle $\mathscr{T}_t$. Thus, for each triangle $\mathscr{T}_t$, we solve between 1 and D^3 NLP problems (6)-(10).

If for none of the shift variable combinations the $\Delta_t \leq \frac{\delta}{2}$, we use a so-called *sub-division rule* to sub-divide triangle $\mathscr{T}_t$ into smaller triangles. Given is $\mathscr{T}_t$ with vertices $[v_{t1}, v_{t2}, v_{t3}]$ and their three center points p_{ti} of each side of the triangles, *i.e.*, $p_{t1} := (v_{t1} + v_{t2})/2$, $p_{t2} := (v_{t2} + v_{t3})/2$ and $p_{t3} := (v_{t1} + v_{t3})/2$. Now, we divide $\mathscr{T}_t$ into four triangles as follows:

$$\begin{aligned}
& [v_{t1}, p_{t1}, p_{t3}] \quad , && (13)\\
& [v_{t2}, p_{t1}, p_{t2}] \quad , && (14)\\
& [v_{t3}, p_{t2}, p_{t3}] \quad , && (15)\\
& [p_{t1}, p_{t2}, p_{t3}] \quad . && (16)
\end{aligned}$$

Once all triangles in $\mathscr{T}$ yield a piecewise linear $\frac{\delta}{2}$-approximator for f, we need to remove potential discontinuities at the boundary of the triangles. The idea is to sub-divide $\mathscr{T}_t$ into smaller triangles which do not introduce any new vertices at the boundary of $\mathscr{T}_t$. The smaller triangles deviate then at most δ from f. One simple rule of sub-division is to connect each of the vertices on the boundary of $\mathscr{T}_t$ (excluding the vertices of $\mathscr{T}_t$) with at least one other vertex at the boundary of $\mathscr{T}_t$ (excluding the vertices of $\mathscr{T}_t$), but not with a vertex at the same side of the triangle.

The described procedure is summarized in Algorithm 3.1. Set $\mathbb{S}$ is a set of ordered pairs $\{s_{tj}, v_{tj}\}$ which assigns each vertex v_{tj} of a triangle a shift variable s_{tj}. With a triangulation $\mathscr{T}$ and the shift variables corresponding to each vertex of the triangles, the construction of the δ-approximator ℓ is well defined.

Using Algorithm 3.1, we obtain a triangulation for the domain D_4 leading to a continuous, piecewise linear approximation of function f with the desired discretization error δ. Furthermore, the algorithm terminates in finitely many iterations:

Corollary 1 *Algorithm 3.1 terminates after a finite number of iterations for any continuous function $f : \mathbb{D} = D_4 \subseteq \mathbb{R}^2 \to \mathbb{R}$ over a compactum $\mathbb{D}$, any $\delta > 0$ and $D \in \mathbb{N}$. The calculated triangles together with the shift variables yield a piecewise linear, continuous δ-approximator for f.*

Proof The outline of the proof is as follows. First, we show that the minimal side length among all triangles computed in steps 8-27 by the Algorithm 3.1 in order to meet the $\frac{\delta}{2}$-approximation for f is bounded below, *i.e.*, the area of each of the triangles does not get arbitrarily close to zero (follows from the continuity of f and the compactness of $\mathbb{D}$). Second, Algorithm 3.1 is able to compute triangles of such smallest size in a finite number of iterations (follows from the triangle division rule). Third, the obtained triangulation is finite (follows from the first part of the proof). Fourth, the triangulation yields a continuous function (follows from construction of the triangulation); piecewise linearity is immediate.

1st part: The first part is motivated by the proof of Theorem 1 in Rebennack & Kallrath (2012, [15]), in the following just referred to as Theorem 1. Recall the definition of an open ball with center z using metric $\|\cdot\|$ as

$$B_\mu(z) := \{x; \|x-z\| < \mu\},$$

with $\mu > 0$. For the purpose of this proof, $B_\mu(z) \subset \mathbb{R}^2$.

Use the construction scheme for the open cover of $\mathbb{D}$ with finitely many balls $B_{\gamma_l}(x_0^l)$, $l = 1,\ldots,L$ as described in the proof of Theorem 1 by choosing

$$\eta := \left(1 - \frac{D-1}{D+1}\right)\frac{\delta}{4}$$

instead of $\frac{\delta}{2}$. Define

$$l^* = \text{argmin}\{\gamma_l, l = 1,\ldots,L\}$$

as the index of the ball with the smallest radius among all balls. The ball $B_{\gamma_{l^*}}$ allows us to bound the size of any triangle $\mathscr{T}_t$ from below: the smallest (in terms of area as well as minimal side length) among all triangles $\mathscr{T}_t$ computed in steps 8-27 of Algorithm 3.1 has just to be small enough to fit into the ball $B_{\gamma_{l^*}} \cap \mathbb{D}$. By construction of $B_{\gamma_{l^*}}$ (this is where we need the choice for η), if there is a triangle $\mathscr{T}_{\tilde{t}} \subset B_{\gamma_{l^*}} \cap \mathbb{D}$, then there exists a piecewise linear function for triangle $\mathscr{T}_{\tilde{t}}$ which deviates at most δ from f, regardless if the shift variables for the vertices of $\mathscr{T}_{\tilde{t}}$ are fixed or not (a shift of zero suffices for all non-fixed shifts of the vertices).

2nd part: The triangle division rule allows for the construction of triangles with arbitrarily small side length. This can be seen as follows: if triangle $\mathscr{T}_t$ is divided into sub-triangles, then each of the side lengths of the four sub-triangles is smaller than one of the side lengths of the triangle $\mathscr{T}_t$.

3rd part: In the first part of the proof, we showed that the area of the smallest triangle $\mathscr{T}_{\tilde{t}}$ included in any computed triangulation in steps 8-27 by Algorithm 3.1 is

bounded below by a positive number. Because a finite number of identical triangles $\mathscr{T}_t$ suffice to cover $\mathbb{D}$ (overlap on the boundary of $\mathbb{D}$ is allowed to derive an upper bound on $|\mathscr{T}|$), any triangulation $\mathscr{T}$ computed in steps 8-27 by Algorithm 3.1 contains a finite number of triangles. Steps 29-33 of Algorithm 3.1 add a finite number of additional triangles because there can only be a finite number of vertices $v_{\tilde{t}j}$ included on any side of triangle $\mathscr{T}_t$.

4th part: Continuity is ensured at the boundary of each triangle $\mathscr{T}_t$ by ensuring that no vertex of any other triangle lies on the boundary and by keeping track of fixed shift variables (steps 12 & 31). □

Algorithm 3.1 Heuristic to Compute Triangulation and δ-Approximator

1: // **INPUT:** Continuous function f, scalar $\delta > 0$, and shift variable discretization size D
2: // **OUTPUT:** Set of triangles $\mathscr{T}$ and shift values $\mathbb{S}$
3: // Initialize
4: $x_0 := X_{-}$, $\mathscr{T} := \emptyset$, $\mathbb{S} := \emptyset$
5: // rectangle $[X_{1-}, X_{1+}] \times [X_{2-}, X_{2+}]$ is divided into two triangles
6: $\overline{\mathscr{T}} = \{[(X_{1-}, X_{2-}), (X_{1+}, X_{2-}), (X_{1-}, X_{2+})], [(X_{1-}, X_{2+}), (X_{1+}, X_{2-}), (X_{1+}, X_{2+})]\}$
7: // **Divide triangles until all triangles satisfy the $\frac{\delta}{2}$-criteria**
8: **repeat**
9: // obtain and remove triangulation
10: choose $\mathscr{T}_t \in \overline{\mathscr{T}}$ with vertices $[v_{t1}, v_{t2}, v_{t3}]$ and update $\overline{\mathscr{T}} \leftarrow \overline{\mathscr{T}} \setminus \mathscr{T}_t$
11: // obtain fixed variables
12: if $\{\cdot, v_{tj}\} \in \mathbb{S}$, then obtain s_{tj} and fix this variable for formulation (6)-(10); for all $j = 1,2,3$
13: **repeat**
14: for all vertices v_{tj} with un-fixed shift s_{tj}, obtain a discretized value via (12)
15: // optimize
16: solve (6)-(10) with fixed shift variables to obtain Δ_b
17: **until** $\Delta_t \leq \frac{\delta}{2}$ or all discretize values for the shift variables have been checked
18: // check $\frac{\delta}{2}$-criteria
19: **if** $\Delta_t \leq \frac{\delta}{2}$ **then**
20: // update the set of triangles...
21: $\mathscr{T} \leftarrow \mathscr{T} \cup \{\mathscr{T}_t\}$
22: // ...and the shift variables
23: $\mathscr{S} \leftarrow \mathscr{S} \cup \{\{s_{tj}, v_{tj}\}, j = 1,2,3\}$
24: **else**
25: construct new triangles via (13)-(16) and add them to set $\overline{\mathscr{T}}$
26: **end if**
27: **until** $\overline{\mathscr{T}} = \emptyset$
28: // **Remove discontinuities**
29: **for all** $\mathscr{T}_t \in \mathscr{T}$ **do**
30: **if** $\exists \mathscr{T}_{\tilde{t}} \in \mathscr{T} \in \mathscr{S}$ where $v_{\tilde{t}j}$ lies on one of the sides of $\mathscr{T}_t$ for some j **then**
31: sub-divide triangle $\mathscr{T}_t$
32: **end if**
33: **end for**

The proof provides us with the following three insights:

(1) If continuity for an approximator is not necessary, then Algorithm 3.1 can be modified as follows: steps 29-33 can be removed and the criteria for Δ_t in steps 17 and 19 can be relaxed to $\Delta_t \leq \delta$. This yields a finitely convergent algorithm computing a piecewise linear δ-approximator for f on D_4.

(2) Computationally, it is an advantage to terminate loop 13-17 if the error $\Delta_t > \delta$. In this case, there exists no combination of shift variables which can ensure that $\Delta_t \leq \frac{\delta}{2}$ for the particular triangle $\mathscr{T}_t$.

(3) The only requirement for any sub-division rule is that after finitely many iterations, triangles of arbitrarily small side lengths (*e.g.*, of side length less than or equal to a fixed $\gamma > 0$) can be computed. Thus, we suggest an alternative sub-division rule using a point of maximal deviation for triangle $\mathscr{T}_t$ to obtain a refined triangulation: using the point p_t^* of maximal deviation of ℓ and f over $\mathscr{T}_t$, obtained by solving (6)-(10), we divide $\mathscr{T}_t$ with vertices $[v_{t1}, v_{t2}, v_{t3}]$ into three triangles as follows (we found fixing the free shift variables to zero for computing point p_t^* as computationally most efficient):

$$[v_{t1}, v_{t2}, p_t^*] \quad , \tag{17}$$

$$[v_{t2}, v_{t3}, p_t^*] \quad , \tag{18}$$

$$[v_{t3}, v_{t1}, p_t^*] \quad . \tag{19}$$

The case that p_t^* happens to lie on any of the three sides of the triangle $\mathscr{T}_t$ needs special care. First, we remove the triangle with zero area. Second, we need to ensure that the calculated function approximation is continuous at this particular side of the triangle $\mathscr{T}_t$. The continuity can be ensured by a simple trick: divide the one neighboring triangle which contains point p_t^* into three triangles using (17)-(19) (ignoring the triangle with zero area). This sub-division rule has one drawback: one could imagine a case where the computed triangles get arbitrarily small in area, but not in the side length (*e.g.*, a point of maximal deviation is always in the interior of the computed series of triangles). To avoid this issue, one could sub-divide the triangles into smaller triangles using the subdivision rule used by Algorithm 3.1 every fixed iteration count.

The subdivision rule using a point of maximal deviation has the advantage over the other sub-division rule that the number of triangles is expected to increase slower.

3.3 Deriving Good δ-Underestimators and δ-Overestimators

The easiest way to construct δ-under- and overestimators $\ell_\pm(x)$ in the bivariate case is to exploit the interpolation-based approximation $\ell(x)$ of $f(x)$ accurate to $\frac{\delta}{2}$ and to set $\ell_\pm(x) := \ell(x) \pm \frac{\delta}{2}$. However, if the $\frac{\delta}{2}$-approximator for f does not possess a minimal number of triangles, then the computed δ-under- and overestimators are not minimal in the number of triangles used in the triangulation, *cf.* Rebennack & Kallrath (2012, [15], Corollary 1). Therefore, we briefly describe a specific method to compute δ-underestimators or δ-overestimators.

Our specific calculation of δ-underestimators or δ-overestimators follows very closely the idea of δ-approximators. As in the one-dimensional case, we focus our discussions on δ-underestimators. Instead of using (5), we use for the underestimator $\ell_-(p)|_{p\in\mathscr{T}_t}$

$$\Delta_t^+ := \max_{p\in\mathscr{T}_t} \big(f(p) - \ell_-(p)\big) \leq \delta$$

$$\text{s.t.} \quad \ell_-(p) \le f(p) \quad , \quad \forall p \in \mathscr{T}_t \quad . \tag{20}$$

Analogously to the one-dimensional case, the continuum conditions (20) are discretized for a given triangle into I grid points p_{ti}. This is achieved by choosing λ_{1i} and λ_{2i} with $i \in \mathbb{I}$, yielding to $\lambda_{3i} = 1 - \lambda_{1i} - \lambda_{2i}$, ensuring that $\lambda_{1i}, \lambda_{2i}, \lambda_{3i} \in [0,1]$. This generates a system of grid points p_{ti},

$$p_{ti} = \sum_{j=1}^{3} \lambda_{ji} v_{tj} \quad , \quad \forall t \quad , \quad \forall i \in \mathscr{I} \tag{21}$$

contained in triangle t.

Let $\mathscr{T}_t$ be a triangle with vertices $[v_{t1}, v_{t2}, v_{t3}]$. For underestimators, the NLP (6)-(10) is replaced by:

$$\begin{aligned}
\Delta_t^{D+} := \ & \min \ \eta && (22)\\
& \eta \ge f(p_{ti}) - \ell_-(p_{ti}) \quad , \quad \forall i \in \mathscr{I} && (23)\\
\text{s.t.} \ & \ell_-(p_{ti}) \le f(p_{ti}) \quad , \quad \forall i \in \mathscr{I} && (24)\\
& \ell_-(p_{ti}) := \textstyle\sum_{j=1}^{3} \lambda_{ji} \phi(v_{tj}) \quad , \quad \forall i \in \mathscr{I} && (25)\\
& \eta \ge 0, \quad s_{tj} \in \left[-\frac{1}{3}\delta, 0\right], \quad j = 1,2,3 \quad , && (26)
\end{aligned}$$

with $\phi(\cdot)$ as given by (11); the λ_{ji} are fixed and obtained by (21). Notice that the shift variables are not discretized, in contrast to the approach described in Section 3.2. Dependent on how many shift variables s_{tj} are fixed, problem (22)-(26) may be trivially solved.

If $\Delta_t^{D+} > \frac{1}{3}\delta$, one can proceed with a sub-division rule as in the case for δ-approximators to further divide the triangle $\mathscr{T}_t$. However, if $\Delta_t^{D+} \le \frac{1}{3}\delta$, we need to ensure that the derived ℓ_- is indeed an underestimator for f. Therefore, we need to check whether

$$z_{\pm}^{\max *} := \max_{p \in \mathscr{T}_t} f(p) - \ell_-(p) \le \frac{1}{3}\delta \quad , \tag{27}$$

and

$$z_{\pm}^{\min *} := \min_{p \in \mathscr{T}_t} f(p) - \ell_-(p) \ge 0 \quad . \tag{28}$$

If both conditions are met, then the computed ℓ is an underestimator for f on triangle $\mathscr{T}_t$. Thus, we can keep $\mathscr{T}_t$ as well as the shift variables s_{ti}^*. Otherwise, we have to divide the triangle $\mathscr{T}_t$ further. To ensure continuity at the boundary of the triangles $\mathscr{T}_t$ we proceed as in the case for δ-approximators (steps 29-33 of Algorithm 3.1). Shifting the obtained approximator by $-\frac{1}{3}\delta$ ensures a piecewise linear, continuous δ-underestimator for f.

3.4 Convex-linear Combinations of Triangles and SOS-Formulations

Let us discuss how to proceed once we have a good triangulation at hand. A special focus is on efficiency aspects.

We start with a straight forward approach: We exploit $T = |\mathscr{T}|$ binary variables δ_t indicating whether triangle $\mathscr{T}_t$ and its vertices are selected to compute the convex combination of f. These binary variables control which ones of the non-negative λ_{ti} variables can take positive values: Namely only those associated with $\mathscr{T}_t$, *i.e.*,

$$\sum_{i=1}^{3} \lambda_{ti} = \delta_t \quad , \quad \forall t \in \mathscr{T} \quad .$$

If p_{tij} denotes the j-th component of the vector p_{ti}, the function arguments x_1 and x_2 are represented as

$$x_j = \sum_{t\in\mathscr{T}} \sum_{i=1}^{3} \lambda_{ti} p_{tij} \quad , \quad \forall j \in \mathscr{J} = \{1,2\} \quad ,$$

and with $f_{ti} = f(p_{ti})$ we get the linear approximation of the function as

$$f = \sum_{t\in\mathscr{T}} \sum_{i=1}^{3} \lambda_{ti} f_{ti} \quad .$$

The condition that only one triangle can be selected is modeled as

$$\sum_{t\in\mathscr{T}} \delta_t = 1 \quad .$$

Alternatively, to declaring the δ_t variables binary, we could also declare them as a SOS-1. However, there is not much structure which this SOS-1 representation could exploit; at best, the average of the function values at the vertices could serve as a reference row. Note that is important to get the δ_t variables fixed at an early stage of the branching to get accurate approximations of the functions. It is worthwhile to try using priorities related to these variables.

The formulation could be enhanced by constructing an SOS-2 containing $3T$ variables u_τ with

$$\begin{array}{lll} u_\tau = \lambda_{t1} & , & \tau = 1+3(t-1) \\ u_\tau = \lambda_{t2} & , & \tau = 2+3(t-1) \\ u_\tau = d_t & , & \tau = 3t \end{array} \quad ,$$

where d_t are dummy variables set to zero. That way, the SOS-2 looks like

$$\lambda_{11},\lambda_{12},0,\lambda_{21},\lambda_{22},0,\lambda_{31},\lambda_{32},0,\ldots$$

That way, only pairs $(\lambda_{t1},\lambda_{t2})$ can be selected. However, to compute λ_{t3} we need the additional information provided by δ_t, *i.e.*, $\lambda_{t3} = 1-\lambda_{t1}-\lambda_{t2}-(1-\delta_t)$. If we use both, the SOS-1 for the δ_t and the SOS-2 for $(\lambda_{t1},\lambda_{t2})$, we obtain a somewhat tighter formulation connecting the argument p and $f(p)$.

We are not aware of any algebraic modeling language which would allow us to construct the SOS-2 described above; therefore, we have not followed up on this approach.

The SOS-2 formulation used by Misener & Floudas (2010, [13]) involving simplices exploits already the recent idea of modified SOS-2 conditions or formulations using significantly fewer binary variables growing only logarithmically in the number of support areas (breakpoints, triangles, or simplices) by Vielma & Nemhauser (2011, [19]). It could also be applied in this context.

Especially, for two-dimensional function approximation D'Ambrosio et al. (2010, [3]) discuss the notion of special ordered sets of type 3 (SOS-3). Unfortunately, SOS-3 are not supported by current MILP solvers.

4 Multivariate Functions and their Linear Approximations

When approximating functions of n variables by piecewise linear functions the ideas and concepts developed for univariate and bivariate functions can be extended or modified to these higher dimensions. However, as, for instance, pointed out by Geißler et al. (2012, [6]), the number of support areas, usually, simplices increases exponentially.

An unsolved, probably problem-specific question to analyze and to answer is whether it is more efficient to approximate separate functions such as $f(x,y) = x^2 + y^2$ in two dimensions or to approximate them in x and y separately. The same questions arises when we encounter functions of more than two variables in NLP or MINLP problems. Is it worthwhile to exploit special properties, *e.g.*, separability, of the functions to reduce the dimensionality, or is it more efficient to approximate the function directly in its dimensionality? Intuitively, one might argue that the reduction of dimensionality always pays out, but this is not so obvious and may depend both on the problem and also on the branching strategies used by the selected MILP solver, *cf.* Section 7.

In the next subsection we provide some transformation for functions with special structure. In Section 4.2, we discuss required function shifts for the introduced transformations. We elaborate on the validity of the transformations in the context of approximator systems in Section 4.3. We close this section with a discussion of possible application areas of the developed approximator systems in Section 4.4.

4.1 Transformation for Special Nonlinear Expressions

Transformations for special nonlinear expressions in n dimensions may in principle enable us to utilize the one- and two-dimensional techniques developed in this paper to construct good δ-approximators for n-dimensional functions. We summarize three types of functions and their transformation tricks in Table 1. A rich class of n-dimensional functions can be obtained by appropriately applying these four function types. Nested transformations are possible for type I and IV.

Table 1: Transformations for n-dimensional functions.

If not otherwise stated, then $f_i(x_i) : [X_{i-}, X_{i+}] \subset \mathbb{R} \to \mathbb{R}$ are continuous functions for all $i = 1, \ldots, n$ and $f(x) : [X_-, X_+] \subset \mathbb{R}^n \to \mathbb{R}$.

	Function $f(x)$	**Transformation**	**Approx. Error for** $f(x)$	**Comment**
I	$\sum_{i=1}^n \pm f_i(x_i)$	treat each term $f_i(x_i)$ individually	$\sum_{i=1}^n \delta_i$ of $f_i(x_i)$	δ_i is approximation error
II	$\prod_{i=1}^n f_i(x_i)$	$\ln(f(x)) = \sum_{i=1}^n \ln(f_i(x_i))$	$f(x)\left(e^{\sum_{i=1}^n \delta_i} - 1\right)$	$f_i(x_i) > 0$ for all i; δ_i is approximation error of $\ln(f_i(x_i))$
III	$f_1(x)^{f_2(x)}$	$\ln(\ln(f(x))) = \ln(f_1(x)) + \ln(\ln(f_2(x)))$	$f(x)\left(e^{e^{(\delta_1+\delta_2)}} - 1\right)$	$f_1(x), f_2(x) > 1$; δ_1 and δ_2 is approximation error for $\ln(f_1(x))$ and $\ln(\ln(f_2(x)))$, respectively
IV	$f_1(f_2(x))$	$f_1(u)$ and $f_2(x)$	$\delta_1 + \gamma(\delta_2)$	δ_1 is approximation error for $f(u)$ and δ_2 is approximation error for $f_2(x)$

$\gamma(\delta_2) := \max_{x \in \mathbb{D},\, x-\delta_2 \le y \le x+\delta_2} |f_1(x) - f_1(y)|$

I: Separable functions. We can use the one-dimensional δ-approximators for each of the n one-dimensional, continuous functions $f_i(x_i)$ separately. The obtained approximation error for the separable function $f(x)$ is then the sum of the individual errors δ_i for each expression $f_i(x_i)$.

II: Positive function products. For products of functions, we require that all functions are positive. Otherwise, assume without loss of generality that exactly one function, $f_j(x_j)$, is non-positive. As f_j is continuous on the compactum $[X_{j-}, X_{j+}]$, f_j is bounded. Therefore, $L_j := \min_{x \in [X_{j-}, X_{j+}]} f_j(x)$ is finite. Now, substitute

$$\prod_{i=1}^n f_i(x_i) = (f(x_j) + D_j) \prod_{i=1, i \neq j}^n f_i(x_i) - D_j \prod_{i=1, i \neq j}^n f_i(x_i)$$

with $D_j = L_j + k$ and some positive number k, *e.g.*, $k = 1$. As

$$(f(x_j) + D_j) \prod_{i=1, i \neq j}^n f_i(x_i) := \tilde{f}(x) > 0 \quad \text{, and}$$
$$D_j \prod_{i=1, i \neq j}^n f_i(x_i) := \bar{f}(x) > 0 \quad ,$$

we can apply the transformation for positive function products to both positive functions $\tilde{f}(x)$ and $\bar{f}(x)$ separately.

Note that the error obtained by the transformation of the product of positive functions depends on $f(x)$. If $\ln(f(x))$ has error $\delta = \sum_{i=1} \delta_i$, then $\Delta f(x)$ follows from

$$f(x) + \Delta f(x) = e^{\ln(f(x))+\delta} = f(x) \cdot e^{\delta}$$

and $\Delta f(x) = f(x)\left(e^{\delta} - 1\right)$, which for small values of δ reduces to $\Delta f(x) \approx f(x) \cdot \delta$. This means in consequence, that we loose the separation property between the x_i variables regarding the discretization error, *i.e.*, although the discretization errors of x_i and x_j are separated for $i \neq j$, the discretization error of the product $f_i(x_i) \cdot f_j(x_j)$ depends on both x_i and x_j (as well as on $f_i(x_i)$ and $f_j(x_j)$). However, if "good" bounds on $f(x)$ are available, then this approach may still be computationally feasible, *e.g.*, $0 < f(x) \leq 1$ is desirable as this guarantees an approximation error for $f(x)$ of at most $e^{\delta} - 1$, or δ for small values of δ, respectively.

III: Exponentials. Chains of exponentials $f_1(x)^{f_2(x)}$ for n-dimensional functions $f_1(x)$ and $f_2(x)$ with $x \in \mathbb{R}^n$ require some care related to the arguments. The transformation works only for $f_1(x) > 0$ and $f_2(x) > 1$.

IV: Substitutions. Complicated terms with more variables appearing as arguments of functions can always be replaced by substitutions. Let $f(x) = f_1\big(f_2(x)\big)$ be a nested function with $x \in \mathbb{D} \subseteq \mathbb{R}^n$. Define $u := f_2(x)$ and $\tilde{\mathbb{D}} := \big\{u \mid u = f_2(x), x \in \mathbb{D}\big\} \subseteq \mathbb{R}$. Then, $f_2 : \mathbb{D} \to \tilde{\mathbb{D}}$. If function $f_2(x)$ is approximated with an absolute error of δ_2, then this leads to a maximal error of $\gamma(\delta_2)$ for f_1 (if f_1 is represented exactly). The function $\gamma(\delta_2)$ is the maximal deviation of function f_1 in its domain over a small variation with magnitude δ_2. Note that $\gamma(\delta_2)$ can be overestimated using the derivative of f_1 as follows (f_1 is a one dimension function):

$$\gamma(\delta_2) \leq f'_* \delta_2, \tag{29}$$

where $f'_* = \max_{u \in \tilde{\mathbb{D}}} \frac{\partial f_1(u)}{\partial u}$. The errors of an approximation of f_1 and $\gamma(\delta_2)$ are then additive for function $f(x)$. If the derivative or slope of the outer function f_1 is reasonably bounded, the discretization error of $f(x)$ is remarkably small and well controlled.

Note: As the sum of finitely many continuous piecewise linear functions is a continuous piecewise linear function, approximating each individual function in Table 1 leads to piecewise linear, continuous functions.

$\tilde{\delta}$-underestimators and $\tilde{\delta}$-overestimators for function $f(x)$ can be obtained by computing a $\delta = \frac{\tilde{\delta}}{2}$ approximator for $f(x)$, *e.g.*, by applying any of the transformations of Table 1, *cf.* Rebennack & Kallrath (2012, [15], Corollary 1). Alternatively, one can compute $\tilde{\delta}$-underestimators and $\tilde{\delta}$-overestimators directly by computing appropriate under- and overestimators for the individual functions.

4.2 Function Shifts

Transformations of type II and III (*cf.* Table 1) require the univariate functions $f_i(x_i)$ to be strictly greater than zero. Furthermore, triangulation approaches for bivariate

functions tend to be numerically troublesome for values of $f(x)$ close to 0, *e.g.*, for function $f(x_1,x_2) = x_1 \cdot x_2$. Therefore, it is recommended to transform the variables appropriately. We discuss this approach for two functions.

$f(x_1,x_2) = x_1 \cdot x_2$: Assume that $X_1^- \le x_1 \le X_1^+$ and $X_2^- \le x_2 \le X_2^+$. Let us construct a linear transformation in such a way that the transformed variables $\tilde{x}_1$ and $\tilde{x}_2$ fall into the interval $[1, 1+\Delta]$ with $\Delta > 0$, implying that $\tilde{x}_1 \cdot \tilde{x}_2 \in [1,(1+\Delta)^2]$. The transformed bilinear product $\tilde{x}_1 \cdot \tilde{x}_2$ is well behaved, if, for instance, $0.25 \le \Delta \le 0.75$. The transformations

$$\tilde{x}_1 := 1 + \frac{x_1 - X_1^-}{E_{x_1}} \quad , \qquad E_{x_1} := \frac{1}{\Delta}\left(X_1^+ - X_1^-\right)$$

and

$$\tilde{x}_2 := 1 + \frac{x_2 - X_2^-}{E_{x_2}} \quad , \qquad E_{x_2} := \frac{1}{\Delta}\left(X_2^+ - X_2^-\right)$$

have the desired properties for any $X_1^- < X_1^+$ and $X_2^- < X_2^+$ values. In order to achieve an approximation error of δ for $x_1 \cdot x_2$, we need to request the approximation error $\tilde{\delta} = \frac{1}{E_{x_1} E_{x_2}}\delta$ for $\tilde{x}_1 \cdot \tilde{x}_2$. This error relation follows direct from the transformation. The inverse transformation is, for i=1,2 given by

$$x_i = -X_i^- + E_{x_i}\left(\tilde{x}_i - 1\right).$$

$f(x_1,x_2) = x_1 \cdot \exp(-x_1^2 - x_2^2)$: We use a type II transformation after re-writing the function as follows

$$f(x_1,x_2) = f_1(x_1) \cdot f_2(x_2) = x_1 \cdot \exp(-x_1^2) \cdot \exp(-x_2^2) \quad .$$

If we want to separate the terms, negative values of x_1 or values close to zero cause problems. Given the bounds $X_1^- \le x_1 \le X_1^+$,

$$\tilde{x}_1 := x_1 + 1 - X_1^-$$

shifts $\tilde{x}_1$ to the interval $[1, 1 + X_1^+ - X_1^-]$.

4.3 Approximating NLPs and MINLPs

Recall that we are interested in approximating the nonlinear optimization problem

$$\begin{aligned} z^* = \min \quad & f(x) && (30)\\ \text{s.t.} \quad & g(x) = 0 && (31)\\ & h(x) \le 0 && (32)\\ & x \in \mathbb{D} && (33) \end{aligned}$$

with domain

$$\begin{aligned} D_1: \quad & \mathbb{D} := [X_-, X_+]^n \subset \mathbb{R}^n, \text{ or}\\ D_2: \quad & \mathbb{D} := [X_-, X_+]^n \subset \mathbb{R}^{n_1} \times \{0,1\}^{n_2} \end{aligned}$$

and $f : \mathbb{D} \to \mathbb{R}$, $g : \mathbb{D} \to \mathbb{R}^{m_1}$, $h : \mathbb{D} \to \mathbb{R}^{m_2}$ being continuous functions as well as $n_1 + n_2 = n$.

For $n > 1$, consider those functions f, g, h which are "separable" in the sense of Table 1. Depending on the anticipated properties of an optimal solution to the approximator system, use either δ-approximators, δ-underestimators or δ-overestimators (*cf.* Rebennack & Kallrath, 2012, [15], Section 3.3). We distinguish three cases:

$f(x)$ **is non-linear:** approximate the objective function $f(x)$ by using any of the four transformations by choosing appropriate approximation errors for each individual term. The logarithm function is a monotone transformation and as such, does not alter the optimal solution values. The optimal objective function value approximating z^* can be obtained by applying the exponent.

$g(x)$ **is non-linear:** re-write this equality constraint into two inequality constraints and apply the case of $h(x)$.

$h(x)$ **is non-linear:** if function $h(x)$ can be expressed as the sum of univariate functions or obtained via substitution (type I or IV in Table 1), then apply the stated transformation to $h(x)$ by using the appropriate, individual approximation errors. For the other two transformation types (*i.e.*, type II and III), $h(x) \leq 0$ needs to be re-written as $\tilde{h}(x) \leq b$ with vector $b > 0$ and $b > 1$, respectively. We then apply transformation II or III as stated in Table 1 to $\tilde{h}(x)$ (if the requirements as listed in the column "comment" still hold). The right-hand-side of the constant $\tilde{h}(x) \leq b$ is then replaced by $\ln(b)$ and $\ln(\ln(b))$, respectively. The monotone transformations do not alter the optimal solutions of the approximate problems.

Note: As the sum of finitely many continuous piecewise linear functions is a continuous piecewise linear function, approximating each individual function in Table 1 leads to piecewise linear, continuous functions.

So far, we have not explicitly discussed discrete variables involved in our MINLP. However, when deriving approximator systems correctly, *e.g.*, as suggested in Section 3.3 of Rebennack & Kallrath (2012, [15]), all the properties regarding ε accuracy of the obtained solutions hold in case that the discrete variables take discrete values; compare also to Geißler et al. (2012, [6]) on the issue of MIP relaxations.

4.4 Possible Application Areas

The literature reviews by Misener & Floudas (2010, [13]) and Geißler et al. (2012, [6]) contain several examples: Water supply network optimization and transient technical optimization of gas networks, generalized pooling and integrated water systems problems, gas lifting and well scheduling for enhanced oil recovery, and electrical networks. Our own application area motivation comes from large scale production planning problems (*cf.* Timpe & Kallrath (2000, [18]), Kallrath (2005, [8]), Kallrath & Maindl (2006, [11], Chap. 8), or Kallrath (2007, [9])), *e.g.*, in chemical industry, where nonlinear cost or yield functions bring in a modest number of nonlinear terms into the model. The size of the problems, several hundred thousand variables and

constraints among them tenthousands of binary variables is prohibitive for current MINLP solvers. Another area of problems comes from the energy sector again with nonlinear, usually, concave costs functions; *cf.* Rebennack et al. (2010, [16]).

5 Computational Results

We have implemented Algorithm 3.1 in the modeling language GAMS version 23.6 [*cf.* Brooke et al. (1992, [1]) or Bussieck and Meeraus (2004, [2])], employing the global optimization solver LindoGlobal version 23.6.5 ([17]). We run the computational tests on an Intel(R) i7 (single core) with 2.93 GHz and 12.0 GB RAM on a 64-bit Windows 7 operating system.

The nine different functions tested are summarized in Table 2. The columns $\mathbf{X}_-$ and $\mathbf{X}_+$ define the lower and upper bounds, respectively, on both decision variables x_1 and x_2. The column "Comment" summarizes some relevant characteristics of the tested functions. The functions are plotted in Figure 1.

Table 2: Two-dimensional functions tested.

#	$\mathbf{f}(x)$	$\mathbf{X}_-$	$\mathbf{X}_+$	**Comment**
1	$x_1^2 - x_2^2$	[0.5,0.5]	[7.5,3.5]	D.C. function (Horst et al., 2000, [7])
2	$x_1^2 + x_2^2$	[0.5,0.5]	[7.5,3.5]	convex function
3	$x_1 \cdot x_2$	[2.0,2.0]	[8.0,4.0]	bilinear, convex function on domain used in Section 6
4	$x_1 \cdot \exp(-x_1^2 - x_2^2)$	[0.5,0.5]	[2.0,2.0]	maximum function value: ≈ 0.334
5	$x_1 \sin(x_2)$	[1.0,0.05]	[4.0,3.1]	concave function on domain
6	$\frac{\sin(x_1)}{x_1} x_2^2$	[1.0,1.0]	[3.0,2.0]	–
7	$x_1 \sin(x_1) \sin(x_2)$	[0.05,0.05]	[3.1,3.1]	–
8	$\left(x_1^2 - x_2^2\right)^2$	[1.0,2.0]	[1.0,2.0]	–
9	$\exp\left(-10(x_1^2 - x_2^2)^2\right)$	[1.0,1.0]	[2.0,2.0]	steep peak at $x_1 = x_2$

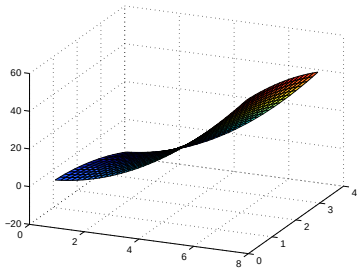
Table 3 summarizes the transformations applied towards functions 1 though 7 of Table 2. The column "Type" indicates which type of transformation, as defined in Table 1, has been applied. For all computations, we choose both δ_1 and δ_2 to be equal. For type I transformations, this leads to $\delta_1 = \delta_2 = \frac{\delta}{2}$ (*cf.* Table 1 column "approx. error"). The individual approximation errors for type II transformations are given by

$$\delta_1 = \delta_2 = \frac{1}{2} \ln\left(\frac{\delta}{m^*} + 1\right),$$

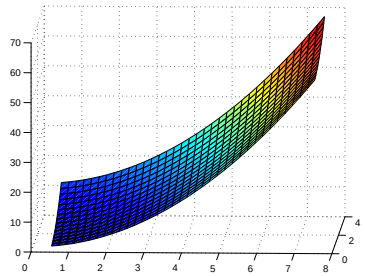
with

$$m^* := \max_{x \in [\mathbf{X}_-, \mathbf{X}_+]} |f(x)|.$$

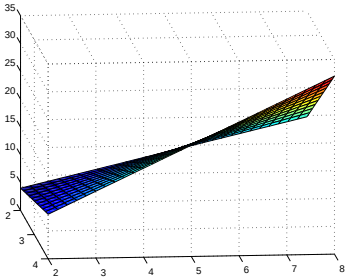
If the exact value of m^* is missing, then we use an overestimator m^+ for m^* (*i.e.*, $m^+ \geq m^*$). The values for m^+ and/or m^* are given in column "Comment" of Table 3.



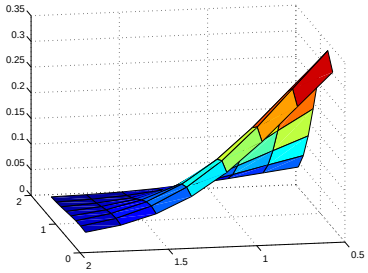
(a) Function 1



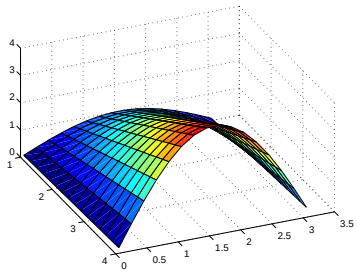
(b) Function 2



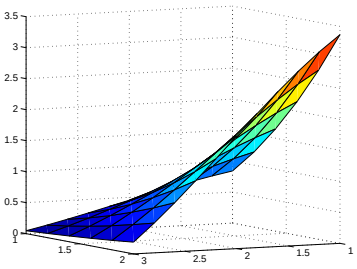
(c) Function 3



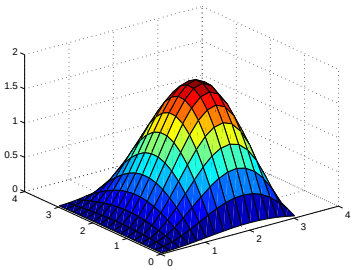
(d) Function 4



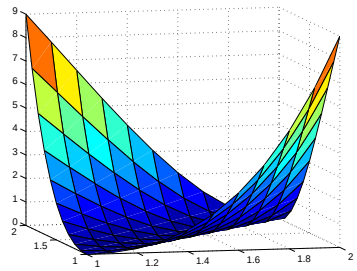
(e) Function 5



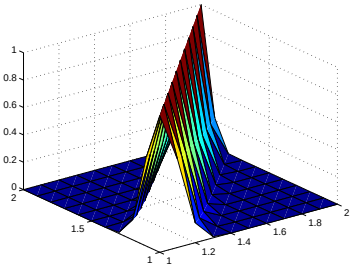
(f) Function 6



(g) Function 7



(h) Function 8



(i) Function 9

Fig. 1: Plots for functions of Table 2.

Table 3: Transformations to one-dimensional functions for functions 1 to 7 of Table 2.

#	$\mathbf{f_1}(x_1)$	$\mathbf{f_2}(x_2)$	Type	Comment
1	x_1^2	$-x_2^2$	I	–
2	x_1^2	x_2^2	I	–
3	$\ln(x_1)$	$\ln(x_2)$	II	$m^+ = m^* = 32$
4	$\ln(x_1) - x_1^2$	$-x_2^2$	II	$m^+ = 0.3341$
5	$\ln(x_1)$	$\ln\big(\sin(x_2)\big)$	II	$m^+ = m^* = 4$
6	$\ln\big(\sin(x_1)\big) - \ln(x_1)$	$2\ln(x_2)$	II	$m^+ = 3.37$
7	$\ln\big(\sin(x_1)\big) + \ln(x_1)$	$\ln\big(\sin(x_2)\big)$	II	$m^+ = 1.82$

For functions 8 and 9 of Table 2, we apply the substitution rule: case IV of Table 1. The resulting one-dimensional function $f_1(u) : \tilde{\mathbb{D}} \to \mathbb{R}$ is stated along with the two-dimensional, nested function $f_2(x_1, x_2)$ (the domain of f_2 is stated in Table 2). The choice for the approximation errors δ_1 and δ_2 for f_1 and f_2, respectively, are stated in the last two columns of the table. The two dimensional function $f_2(x_1, x_2) = x_1^2 - x_2^2$ can be approximated by applying a type I transformation, choosing an individual approximation error of $\frac{\delta_2}{2}$, for instance. In order to compute δ_2 for function 9, we have used the maximal derivative of $2\sqrt{\frac{5}{e}}$ in order to overestimate $\gamma(\delta_2)$, see equation (29).

For our computations via Algorithm 3.1, we use the maximal deviation point in each triangle as the sub-division rule (as described in Section 3.2). We always observed good convergence behavior of the algorithm. Steps 29-32 of Algorithm 3.1 were never entered (e.g., the computed piecewise linear approximators were always continuous after step 27). Thus, we allowed a deviation of δ instead of $\frac{\delta}{2}$ up front. Empirically, we observed that a discretization of the shift variables of

$$-\frac{\delta}{2}, -\frac{\delta}{4}, 0, \frac{\delta}{4}, \frac{\delta}{2}$$

is a good trade-off between computational time and number of triangles computed.

Table 4: Substitutions for function 8 and 9 of Table 2.

#	$\mathbf{f_1}(u)$	$\tilde{\mathbb{D}}$	$\mathbf{f_2}(x_1, x_2)$	δ_1	δ_2
8	u^2	[0,4]	$x_1^2 - x_2^2$	$\frac{\delta}{2}$	$\delta_2 = 4 - \sqrt{4 + \frac{\delta}{2}}$
9	$\exp(-10u^2)$	[0.4]	$x_1^2 - x_2^2$	$\frac{\delta}{2}$	$\delta_2 = \frac{\delta}{4}\sqrt{\frac{e}{5}}$

The computational results for function 1 through 9 of Table 2 are summarized in Table 5. For each function $f(x)$, we choose five consecutive values for the approximation error δ among the set $\{1.50, 1.00, 0.50, 0.25, 0.10, 0.05, 0.03, 0.01, 0.001\}$, dependent on the scaling of the function. The results for the 2-D approach are computed by Algorithm 3.1. The column $|T|$ states the number of triangles used. For

the 1-D approach, we use the Algorithm 4.1 as developed by Rebennack & Kallrath (2012, [15]). δ_i is the approximation error applied to both functions $f_1(x_1)$ and $f_2(x_2)$, except for function 8 and 9 where δ_2 is stated. B_1 and B_2 are the computed number of breakpoints for function f_1 and f_2, respectively. Column "$|R|$" reports on the number of rectangles resulting from the obtained breakpoint systems; again, functions 8 and 9 are different. There, we report the number of rectangular prisms leading to feasible values for x_1, x_2 and u. For both the 1-D and the 2-D, "dev." summarizes the maximal deviation of the obtained piecewise linear, continuous function over the triangulation compared to the approximated function $f(x)$. These values have been obtained by solving a series of global optimization problems after the approximations have been computed (the computational times are not reported). The columns "CPU (sec.)" provide the computational times in seconds.

From the numerical results presented in Table 5, we derive two main conclusions: (1) At a first glance, the advantage of applying approximations schemes, *e.g.*, piecewise linear approximations by triangulation to bivariate functions, seems not as striking as expected because separate one-dimensional piecewise linear approximations seem to require less breakpoints (particularly for functions which separate well, *e.g.*, functions 1 and 2). However, whether this is really an advantage depends on the behavior of a MILP solver when both the triangles and the one-dimensional breakpoints systems are implemented; see further discussions in Section 7. (2) An additional limitation of one-dimensional separable approaches is the numerical accuracy required. For instance, the numerical errors when using logarithmic separations approaches involve the function values themselves. This may request very small errors of the order of 0.001 or smaller, which in turn can cause numerical problems when computing optimal one-dimensional breakpoints systems.

Triangulations calculated by Algorithm 3.1 are shown in Figure 2 for different values of δ.

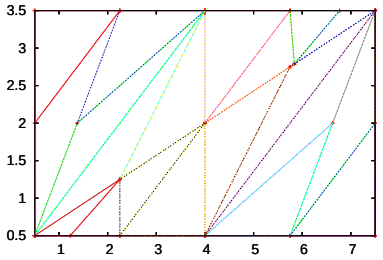
6 A Two-Dimensional Problem: Cutting Circles from One Rectangle

We apply the developed 2D-methods in this paper and the 1D-approach developed by Rebennack and Kallrath (2012,[15]) to a circle cutting stock problem which was part of a larger production planning problem in the metals industry. A few hundred orders with due dates ranging over 4 weeks should be grouped into appropriate subgroups of circles to be produced. It is possible to produce orders to stock. Addressing the production planning and cutting stock problem in this real world problem simultaneously, leads to a MILP part for the production planning aspects and complicating NLP or MINLP cutting stock extensions. We do not present the overall production planning problem but rather focus on the cutting stock part. This view is obtained after separating the production planning and cutting stock problem which is often still the preferred solution in the paper and metals industry.

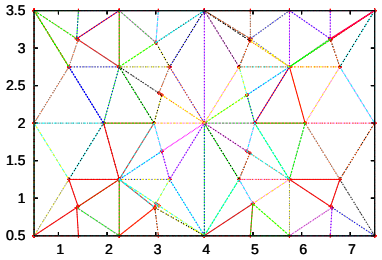
Given is a set of circles characterized by their radii, and bounds on the length and width of a design rectangle. The task is to find a rectangle, hosting all circles, of minimal area; *i.e.*, one needs to find a placement of the circle on a rectangle such that none of the circles overlap, such that all circles are complectly contained in the

Table 5: Computation results for triangulations and one-dimensional transformations.

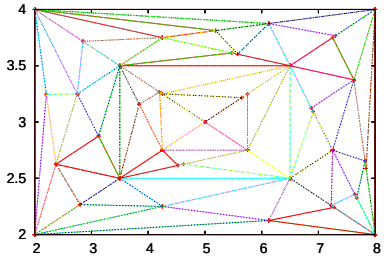
		2-D			1-D					
#	δ	$\|T\|$	dev.	CPU (sec.)	δ_i	B_1	B_2	$\|R\|$	dev.	CPU (sec.)
1	1.50	16	1.4844	30.8	0.7500	4	3	6	1.4764	0.5
	1.00	20	0.9844	84.4	0.5000	5	3	8	0.9967	0.4
	0.50	48	0.5000	150.4	0.2500	6	4	15	0.4990	0.5
	0.25	80	0.2461	272.6	0.1250	9	5	32	0.2499	1.2
	0.10	224	0.1000	380.6	0.0500	13	6	60	0.1000	1.6
2	1.50	24	1.5000	26.8	0.7500	4	3	6	1.5000	0.5
	1.00	28	0.9712	7.4	0.5000	5	3	8	1.0000	0.4
	0.50	84	0.4554	38.0	0.2500	6	4	15	0.5000	0.5
	0.25	121	0.2428	35.8	0.1250	9	5	32	0.2500	1.2
	0.10	351	0.0949	171.7	0.0500	13	6	60	0.1000	1.6
3	1.00	4	0.7500	0.8	0.0153	4	3	6	0.5446	0.4
	0.50	12	0.4444	72.4	0.0077	5	3	8	0.4966	0.5
	0.25	20	0.2344	4.7	0.0038	7	4	18	0.1697	0.6
	0.10	59	0.0968	59.3	0.0015	10	6	45	0.0889	1.0
	0.05	94	0.0490	45.3	0.0007	15	8	98	0.0413	1.3
4	0.10	2	0.0976	0.3	0.1309	3	3	4	0.0908	0.4
	0.05	6	0.0346	18.7	0.0697	4	4	9	0.0454	0.6
	0.03	10	0.0288	12.7	0.0429	5	4	12	0.0279	0.7
	0.01	31	0.0097	54.6	0.0147	7	6	30	0.0100	0.9
	0.001	350	0.0010	652.6	0.0014	19	16	270	0.0009	2.7
5	1.00	5	0.9542	1.0	0.1115	3	7	12	0.8911	0.6
	0.50	8	0.4803	13.1	0.0588	3	9	16	0.3219	1.2
	0.25	16	0.2442	30.0	0.0303	3	13	24	0.2441	1.3
	0.10	44	0.0975	74.6	0.0123	5	19	72	0.0924	1.7
	0.05	85	0.0483	141.9	0.0062	6	26	125	0.0434	2.4
6	0.50	2	0.4461	1.8	0.0691	4	2	3	0.4988	0.5
	0.25	4	0.2104	1.0	0.0357	6	3	10	0.1813	0.6
	0.10	9	0.0976	25.8	0.0146	8	4	28	0.0971	1.4
	0.05	23	0.0451	14.4	0.0073	10	4	27	0.0495	1.0
	0.03	40	0.0297	161.4	0.0044	13	5	48	0.0228	2.6
7	1.00	6	0.4885	7.7	0.2189	5	6	20	0.9764	1.2
	0.50	6	0.4885	1.3	0.1213	7	8	42	0.4280	1.4
	0.25	21	0.2351	30.8	0.0643	9	11	80	0.2089	1.3
	0.10	96	0.0980	73.0	0.0267	13	15	168	0.0944	2.6
	0.05	274	0.0498	305.5	0.0135	18	21	340	0.0497	3.4
8	1.00	6	0.8204	22.8	0.0310	4	4	6	0.6117	0.5
	0.50	9	0.4340	15.6	0.0155	4	4	7	0.2852	0.5
	0.25	12	0.2439	22.9	0.0077	6	6	18	0.1575	0.7
	0.10	40	0.0959	202.8	0.0031	8	8	40	0.7312	0.8
	0.05	87	0.0500	174.1	0.0015	11	11	83	0.0384	1.0
9	1.00	2	1.0000	0.8	0.1100	2	3	3	0.5000	0.3
	0.50	4	0.4909	66.6	0.0460	3	3	4	0.2507	0.3
	0.25	6	0.1744	4.4	0.0230	3	4	7	0.1359	0.4
	0.10	84	0.0945	231.5	0.0092	5	5	18	0.0737	0.6
	0.05	86	0.0480	57.8	0.0046	5	7	34	0.0412	0.7



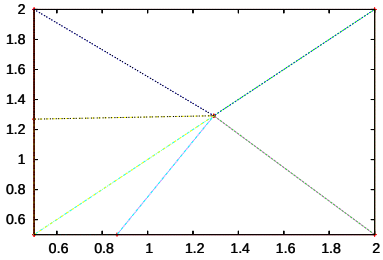
(a) Func. 1: $x_1^2 - x_2^2$, $\delta = 1.00$



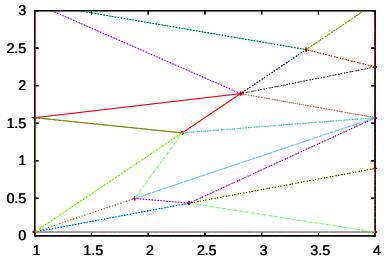
(b) Func. 2: $x_1^2 + x_2^2$, $\delta = 0.50$



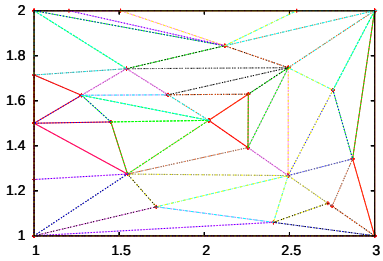
(c) Func. 3: $x_1 \cdot x_2$, $\delta = 0.10$



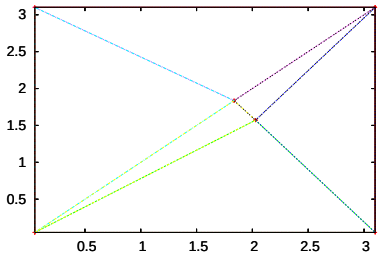
(d) Func. 4: $x_1 \cdot \exp(-x_1^2 - x_2^2)$, $\delta = 0.05$



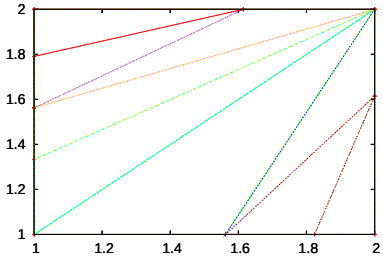
(e) Func. 5: $x_1 \sin(x_2)$, $\delta = 0.25$



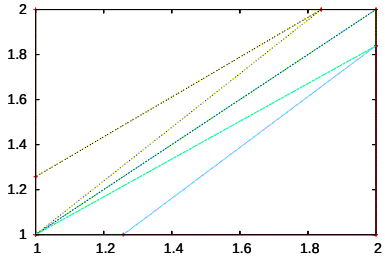
(f) Func. 6: $\frac{\sin(x_1)}{x_1} x_2^2$, $\delta = 0.03$



(g) Func. 7: $x_1 \sin(x_1) \sin(x_2)$, $\delta = 0.50$



(h) Func. 8: $\left(x_1^2 - x_2^2\right)^2$, $\delta = 0.50$



(i) Func. 9: $\exp\left(-10(x_1^2 - x_2^2)^2\right)$, $\delta = 0.25$

Fig. 2: Triangulations (2-D approach) obtained for functions of Table 2.

rectangle and such that the product of the length and the width of the rectangle is minimized. The non-overlapping constraints of the circles lead to non-convex constraints. This problem has been studied by Kallrath (2009, [10]), and more recently also by Misener and Floudas (2012, [14]).

6.1 Nomenclature

We use the following nomenclature in this section. All lower case symbols represent decision variables or indices, while upper case symbols are for input or derived data.

Indices:

$d \in \{1,2\}$ width ($d=1$) and length ($d=2$) of two-dimensional space.
$i \in \mathbb{I} := \{i_1, \ldots, i_I\}$ circles to be packed.
We use also index j to refer to a circle when we compare two circles i and j.

Input data: (the dimensions are provided in brackets "[]")

A_i [L^2] the areas of all circles i; quantity derived as $A_i = \pi R_i^2$.
D [L] the maximum possible length of the diagonal of the design rectangle.
K_p [-] the number of vertices of polygon p.
L [L] maximum size (upper bound) of the length of the design rectangle.
R_i [L] the radius of circle i.
S_{jd} [L] the extension of rectangle j, *i.e.*, width S_{j1} and length S_{j2}.
S_d^+ [L] maximum size (upper bound) of the extension of the design rectangles in dimension d.
S_d^- [L] minimum size (lower bound) of the extension of the design rectangles in dimension d.
W [L] maximum size (upper bound) of the width of the design rectangle.

Variables:

$a \in [S_1^- S_2^-, S_1^+ S_2^+]$ [L^2] the area of the design rectangle. a^- and a^+ are lower and upper bounds on a obtained during the computation.
$x_{id} \in [0, S_d^+]$ [L] the coordinates of the center vector, $\mathbf{x}_i$, of circle i.
For circles i with radius $2R_i \leq \min_d\{S_d^+\}$ the bounds can be refined to $[R_i, S_d^+ - R_i]$.
$x_d^{\mathrm{P}} \in [0, S_d^+]$ [L] the extension of the design rectangle in dimension d.
$z \in [0, S_1^+ S_2^+]$ [L^2] the objective function, area, or, alternatively, trimloss or waste, resp., associated to the optimal solution.
This variable is defined as $z = a$, or $z = a - \sum_i A_i$.

We provide lower and upper bounds in the model wherever possible and as tight as possible.

6.2 Model Formulation as a Non-convex NLP

The objective function to be minimized is the area, a, of the design rectangle

$$\min\ a := \prod_{d=1}^{2} x_d^{\mathrm{P}} \quad , \tag{34}$$

where x_d^{P} represents the extension of the design rectangle in dimension d. Equivalent to this is to minimize waste, *i.e.*,

$$\min\ z := a - \mathbf{A}_{\mathrm{circ}} \quad , \quad \mathbf{A}_{\mathrm{circ}} := \sum_i A_i \quad , \tag{35}$$

where A_i denotes the known areas of the circles.

The extensions x_d^P and of the design rectangle are subject to the bounds

$$S_d^- \le x_d^{\mathrm{P}} \le S_d^+ \quad , \quad \forall d \quad . \tag{36}$$

Note that x_1^{P} is width w, and x_2^{P} is length ℓ. With the known total area $\mathbf{A}_{\mathrm{circ}}$ occupied by the circles and the upper bounds on the sides of the rectangle, we can sometimes improve the lower bounds S_d^- and replace them by

$$S_d^- = max\{S_d^-, \mathbf{A}_{\mathrm{circ}}/S_d^+\}, \quad \forall d \quad . \tag{37}$$

For all circles we have to guarantee that they do not overlap with each other, *i.e.*,

$$(\mathbf{x}_i - \mathbf{x}_j)^2 := \sum_{d\in\mathscr{D}} \left(x_{id} - x_{jd}\right)^2 \ge (R_i + R_j)^2 =: D_{ij}^2 \quad , \quad \forall\{\ (i,j) \mid i < j\} \quad . \tag{38}$$

Note that for I circles we have $I(I-1)/2$ inequalities of type (38).

Fitting the circles inside the enclosing rectangles requires

$$x_{id} \ge R_i \quad , \quad \forall\{i,d\} \quad .$$

and

$$x_{id} + R_i \le x_d^{\mathrm{P}} \le S_d^+ \quad , \quad \forall\{i,d\} \quad .$$

We destroy symmetry by putting the first circle into the first quadrants of the rectangle, *i.e.*, $x_{1d} \le x_d^{\mathrm{P}}/2$, and also by enforcing that the rectangle's length is not smaller than its width, *i.e.*, $x_1^{\mathrm{P}} \ge x_2^{\mathrm{P}}$. A valid lower bound, X_{id}^-, on the center coordinates is also given by the largest radius available.

6.3 Applying Under- and Overestimators

The problem contains the bilinear term $a = w\ell = x_1^{\mathrm{P}} x_2^{\mathrm{P}}$ and the square terms $s_{ijd}^2 = \left(x_{id} - x_{jd}\right)^2$. For a we use the triangle approach with the set $\mathscr{T}$ of triangles t each of them with coordinates $\mathbf{x}_{vt}$, function values F_{vt}, for vertices $v \in \mathscr{V} := \{1,2,3\}$. The binary variable δ_t indicates whether t is selected for the convex combination. For each t, we introduce non-negative variables λ_{tv} subject to

$$\sum_{v \in \mathscr{V}} \lambda_{tv} = \delta \quad , \quad \forall t \quad .$$

In addition we request that exactly one triangle is selected, *i.e.*,

$$\sum_{t \in \mathscr{T}} \delta_t = 1 \quad .$$

The coordinates $\mathbf{x} = (x_1^{\mathrm{P}}, x_2^{\mathrm{P}})$ are given by

$$\mathbf{x} = \mathbf{x}_0 + \sum_{t \in \mathscr{T}} \sum_{v \in \mathscr{V}} \lambda_{tv} \mathbf{x}_{vt} \quad ,$$

with an upper bound $\mathbf{X}$ on $\mathbf{x}_0$, and

$$\mathbf{x}_0 \leq \mathbf{X} \delta_t \quad , \quad \forall t \quad .$$

while the function values $a = f = f(\mathbf{x})$ follow as

$$f = \sum_{t \in \mathscr{T}} \sum_{v \in \mathscr{V}} \lambda_{tv} f_{vt} \quad .$$

For the free variables $s_{ijd} = x_{id} - x_{jd}$, or s_{ijd}^2, respectively, we apply the SOS-2 approach with optimal breakpoints computed as described in Rebennack & Kallrath (2012, [15]). Due to $s_{ii'd}^2 = s_{jid}^2$ this gives $I(I-1)$ special order sets. Furthermore, we exploit that

$$0 \leq \left| s_{ijd} \right| \leq S_d^+ - \min\{R_i, R_j\} \quad , \quad \forall \{ijd\}$$

with $S_1^+ = L$ and $S_2^+ = W$. As the nonlinear function is a square function, a given accuracy of δ is obtained by equidistant breakpoints separated by $\Delta := 2\sqrt{\delta}$. To see this, we consider the quadratic function $f(x) := x^2$ over the interval $[a,b]$ and neglect the shift variables for a moment. The maximal difference between the secant function $s(x)$

$$s(x) := f(a) + \frac{f(b) - f(a)}{b - a}(x - a) \tag{39}$$

and $f(x)$ is realized at

$$x = \frac{b + a}{2} \tag{40}$$

with a maximal difference of

$$\delta_{\max} := \left(\frac{b - a}{2} \right)^2 \quad . \tag{41}$$

Therefore, if we want to limit the maximal difference, $\delta_{\max}$, to δ, we obtain the equidistant distances between breakpoints of $\Delta = 2\sqrt{\delta}$. The accuracy δ is maintained, or improved, if we use shift variables as additional degrees of freedom.

6.4 Implementation Tweaks

As the square function $s^2_{ii'd}$ is convex, its SOS-2 representation with argument breakpoints, X_{ijdb}, and function breakpoints, F_{ijdb}, is in argument

$$s_{ijd} = \sum_{b \in \mathscr{B}} X_{ijdb} \lambda_{ijdb} \quad , \tag{42}$$

and in function value

$$s^2_{ijd} = \sum_{b \in \mathscr{B}} F_{ijdb} \lambda_{ijdb} \quad ,$$

or

$$s^2_{ij} = \sum_{d \in \mathscr{D}} s^2_{ijd} = \sum_{d \in \mathscr{D}} \sum_{b \in \mathscr{B}} F_{ijdb} \lambda_{ii'db} \quad ,$$

respectively, is an overestimator of the real problem. This implies, that the circles may slightly overlap.

For triangle accuracy $\delta = 0.125$ and SOS-2 accuracy $\delta = \frac{1}{64} \approx 0.016$ a solution near to the global optimum, or, to be more precise, a lower bound on the objective function value, resp., is obtained within seconds – much faster than in Kallrath (2009, [10]). However, it is difficult to move up the lower bound. The reason is related to the fact that we explicitly need the unsquared distance s_{ijd} in our model. While the straightforward NLP formulation depends only on the indices i and j, *i.e.*, scales with $I(I-1)/2$, the SOS-2 approach scales with $BI(I-1)$, where B is the number of breakpoints.

We have also reduced the number of triangles involved once we had found a reasonable value for the area a by eliminating all triangles which had function values at their vertices which exceeded a significantly. Once we obtained an integrality gap of less than 20%, we also eliminated all triangles with function values smaller than the lower bound on a.

To improve the numerical behavior of this approach, we have applied various modifications. At first, we have normalized the coordinate difference (42) by $D_{ij} := R_i + R_j$. The advantages is that we can reduce the number of breakpoints significantly. Due to the normalization and as we are working with overestimators we need to have a fine grid only for arguments in the range of $[-1,+1]$. Arguments outside $[-1,+1]$ lead to center differences greater or equal than one and are thus feasible w.r.t. the non-overlap condition. In practice, to compute the distance between the centers of circle i and j the breakpoints

$$-D_{ijd}, -1, -0.75, -0.5, -0.25, 0, +0.25, +0.50, +0.75, +1, +D_{ijd}$$

with

$$D_{ijd} := (S^+_d - D_{ij})/D_{ij} \tag{43}$$

are sufficient. With this improvement, the first feasible point is found always instantaneously, but we still need to do something on moving up the lower bound.

If we knew the absolute coordinate differences $|s_{ii'd}|$, we need only 6 breakpoints

$$0, +0.25, +0.50, +0.75, +1, +D_{ijd}$$

and the normalization (43) allows us to improve the lower bound by adding the cut

$$\sum_{d\in\mathscr{D}} \left|s_{ijd}\right| \geq 1-\varepsilon \quad , \quad \forall\{ijd\} \quad , \tag{44}$$

as for points with $\sum_{d\in\mathscr{D}} |s_{ii'd}| < 1$ we have $\sum_{d\in\mathscr{D}} s_{ijd}^2 < 1$, *i.e.*, overlaps. As distance arguments in the interval $[0, 0.5]$ lead, despite the fact that the secant overestimates s_{ijd}^2, to overlaps, we use the breakpoints

$$0, +0.50, +0.75, +0.90, +1, +D_{ijd} \quad ,$$

as they give better accuracy for near-contact circles.

To compute $\left|s_{ijd}\right|$ we introduce another set of SOS-2 variables, v_{ijdb} with only three breakpoints

$$-D_{ijd}, 0, +D_{ijd} \quad . \tag{45}$$

We use priorities on these SOS-2 variables v_{ijdb} first. The lower bound increases, but it is harder to find integer feasible points.

An attempt to close the integrality gaps for some of the complicated cases involving six congruent circles, was to introduce additional binary variables σ_{ij} which take the value 1 if circle i is positioned left of circle j, *i.e.*, $x_{i1} \leq x_{j1}$. This is enforced by the inequalities

$$\begin{array}{l} x_{i1} \leq x_{j1} + L(1-\sigma_{ij}) \\ x_{i1} \geq x_{j1} - L\sigma_{ij} \end{array} \tag{46}$$

Branching on σ_{ij} has given highest priority. In addition, these binary variables are used to represent the absolute values terms $\left|s_{ijd}\right|$ by

$$\left|s_{ijd}\right| = s_{ijd}^+ + s_{ijd}^-$$

involving the nonnegative variables s_{ijd}^- and s_{ijd}^+ subject to

$$\begin{array}{l} s_{ijd}^- \leq s_{ijd}^{\mathrm{UB}} \sigma_{ijd} \\ s_{ijd}^+ \leq s_{ijd}^{\mathrm{UB}} \left(1-\sigma_{ijd}\right) \end{array} \quad . \tag{47}$$

where s_{ijd}^{UB} denotes an upper bounds on the dimensionless center distance. This avoids the need of extra breakpoints to model $\left|s_{ijd}\right|$. Unfortunately, the binary variables σ_{ijd} did not help as expected.

6.5 Computational Results

In four numerical examples taken from Kallrath (2009) we want to place N_{circ} circles of radii R_i into a rectangle limited by maximal length of $S_1^+ = L = 8$ and a maximum width of $S_2^+ = W = 4$ for cases c6-0 and c6-c, and $W = 2.9$ for cases c6-a and c6-b. We consider the four examples listed in Table 6. $\mathbf{A}_{\text{circ}}$ is the sum of the areas of the circles. The last four columns of the table state the computational results, published in two papers, for the area a together with an absolute gap ("GAP-A"), *i.e.*, GAP-A is the difference of the upper bound and the global lower bound. Misener & Floudas (2012, [14]) publish in their paper the waste $z = a - \mathbf{A}_{\text{circ}}$ instead of the rectangle's area a to quantify their solutions; for Table 6, we obtained their a indirectly. Note that the results for c6-c in Kallrath (2009, [10]) is either premature, or just an error in print. The time limits 45,000 and 7,200 seconds, resp., used by Kallrath (2009) and Misener & Floudas (2012) were reached for case c6-c leaving the gap unclosed.

Table 6: Instances tested together with computational results from the literature.

Case $\mathbf{N}_{\text{circ}}$	**Circle Radii**	$\mathbf{A}_{\text{circ}}$	**Kallrath** (2009, [10])			**Misener & Floudas** (2012, [14])		
			z	a	**GAP-A**	z	a	**GAP-A**
c6-0 6	5 6 8 12 13 17	22.839	7.788	30.627	10^{-8}	7.788	30.627	10^{-4}
c6-a 7	$7 + 6 \times 5$	6.252	2.112	8.363	10^{-9}	2.112	8.363	10^{-4}
c6-b 7	$9 + 6 \times 5$	7.257	1.974	9.231	10^{-9}	1.974	9.231	10^{-4}
c6-c 8	$9 + 7 + 6 \times 5$	11.341	4.098	20.621	10^{-9}	2.798	14.139	2.798

We modeled the problem with GAMS using LindoGlobal to solve the NLP problems to global optimality. The resulting MILP problems are solved using GAMS / CPLEX while the NLPs are locally solved with CONOPT (*cf.* Drud 1994, [4]).

The computational results when using δ-approximator systems with 5 breakpoints are summarized in Table 7. The columns labeled "MILP" present the results for the MILP model, resulting from the usage of the approximator systems; t_a is the time in seconds when the global optimum has been computed (without the necessary proof of optimality) and t_f is the solution time in seconds (with a time limit of 14,000 seconds). The column "NLP" states the results when using an NLP solver to compute a local minimum in the vicinity of the circle center points provided by the MILP solver (note that this initial solution may slightly violate the non-overlap constraints). Column "GLB" reports on the global optimum computed by using GloMIQO developed by Misener & Floudas (2012, [14]). "GAP-A" and "GAP-R" is the absolute and relative gap, respectively, of the MILP solution with respect to the global minimum, a_*.

The minimal area rectangle for the case c6-c is $a = 11.594$, while the MILP solver underestimates this slightly with $a = 11.132$ because circles have small overlaps as displayed in Fig. 3 (c).

While the original NLP problem as described in Section 6.2 consists of 14 Variables, 45 constraints, 60 non-linearities with 62 non-zeros, the MILP models (con-

Table 7: Computational results using δ-approximator systems for the instances of Table 6 based on 1+5 breakpoints.

Case	MILP				NLP	GLB	GAP-A	GAP-R
	a	t_a	GAP-A	t_f	a	a_*		
c6-0	30.258	84.69	10^{-7}	84.69	30.627	30.627	0.369	1.20%
c6-a	8.267	637.01	1.390	14000.00	8.363	8.363	0.096	1.15%
c6-b	9.074	5.34	0.723	14000.00	9.231	9.231	0.157	1.70%
c6-c	11.132	1631.38	1.456	14000.00	**11.757**	11.594	0.462	3.98%

gruent cases) involved, after GAMS/CPLEX's presolve, 270 rows, 510 columns, and 1657 nonzeros. The reduced MILP has 78 binaries, and 30 SOS-2s.

Our MILP approach produces solutions with objective function values which are within a gap of a few percent when compared to the global optimum. To compare the solution (the centers of the circles) to those of global solver we used the center coordinates x_i^*, y_i^* obtained by the MILP solver as an initial starting points to solve the real NLP problem with a local NLP solver such as CONOPT or IPOPT, a free NLP solver available in GAMS. To support this procedure we allowed the center coordinates to vary only slightly, *i.e.*,

$$(x_i - x_i)^2 + (y_i - y_i^*)^2 \leq \Delta^2 \quad , \quad \forall i \quad ,$$

where $\Delta > 0$ is a small number of the order 0.20. We then checked whether this locally optimal solution gave us a solution identical or close to that of the global solver, LindoGlobal; when solving the problem to global optimality, we let out the vicinity inequality (6.5). For the cases c6-a to c6-c the MILP solver did not close the integrality gap within 14,000 seconds. For case c6-c the local NLP solution did not agree to the global solution, a_*. The reason is most probably that the MILP solver missed the optimal solution.

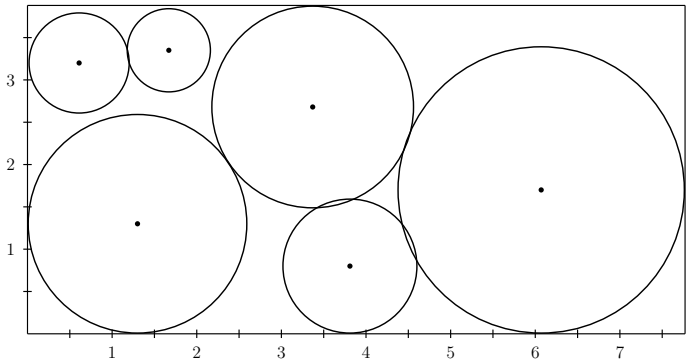
Due to the long running times (especially, for the congruent cases c6-a to c6-c) and for curiosity, we have also implemented the logarithmic SOS-2 approach by Vielma & Nemhauser (2011,[19],Theorem 4) for 1+8 breakpoints

$$0, +0.50, +0.75, +0.80, +0.90, +0.95, +0.98, +1, +D_{ijd} \quad ,$$

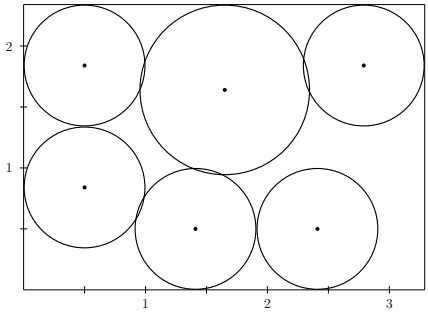
and the 6 inequalities

$$\begin{aligned}
\lambda_0 + \lambda_1 + \lambda_7 + \lambda_8 &\leq x_1 \\
\lambda_2 + \lambda_6 &\leq x_2 \\
\lambda_0 + \lambda_1 + \lambda_2 + \lambda_3 &\leq x_3 \\
\lambda_3 + \lambda_4 + \lambda_5 &\leq 1 - x_1 \\
\lambda_0 + \lambda_4 + \lambda_8 &\leq 1 - x_2 \\
\lambda_5 + \lambda_6 + \lambda_7 + \lambda_8 &\leq 1 - x_3 \quad .
\end{aligned}$$

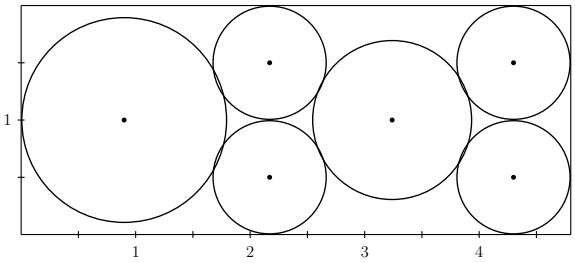
These inequalities have to be generated for each pair of circles and for each of the two coordinate axis. To compare the standard and logarithmic formulation, we also used the standard SOS-2 formulation with 1+8 breakpoints, *cf.* Tables 8 and 9.



(a) c6-0



(b) c6-a



(c) c6-c

Fig. 3: Solutions obtained by the MILP using the δ-approximator system.

First, we observe in the solution of the standard approach that the slight increase of the number of breakpoints from 1+5 to 1+8 leads to a smaller gap between the global solution of the NLP and the MILP problem decreases slightly, as expected. Second, it takes significantly longer to solve the MILP problems (*e.g.*, 900 seconds versus 2423 seconds, for c6-0). Using the logarithmic approach makes a great difference in time (*e.g.*, 2423 seconds versus 14 seconds, for c6-0). It is remarkable how fast the optimal value was reached (though not yet proven), namely in less than 70 seconds, while it took much longer when the standard approach was used with 8

Table 8: Computational results using δ-approximator systems for the instances of Table 6 based on 1+8 breakpoints using the standard SOS-2 formulation.

Case	MILP				NLP	GLB	GAP-A	GAP-R
	a	t_a	GAP-A	t_f	a	a_*		
c6-0	30.296	2423.32	10^{-7}	2423.32	30.627	30.627	0.331	1.09%
c6-a	8.268	1067.53	1.244	14000.00	8.363	8.363	0.095	1.13%
c6-b	9.075	2812.85	0.678	14000.00	9.231	9.231	0.156	1.68%
c6-c	11.172	21.05	1.496	14000.00	**11.758**	11.594	0.422	3.64%

Table 9: Computational results using δ-approximator systems for the instances of Table 6 based on 1+8 breakpoints using the logarithmic SOS-2 formulation by Vielma and Nemhauser.

Case	MILP				NLP	GLB	GAP-A	GAP-R
	a	t_a	GAP-A	t_f	a	a_*		
c6-0	30.296	14	$< 5{\cdot}10^{-7}$	14.43	30.627	30.627	0.332	1.09%
c6-a	8.268	52	0.549	14000.00	8.363	8.363	0.095	1.13%
c6-b	9.075	51	$9{\cdot}10^{-5}$	11820.00	9.231	9.231	0.156	1.68%
c6-c	11.100	67	1.424	14000.00	**11.757**	11.594	0.494	4.26%

breakpoints. Closing the gap for the cases c6-a, c7-b, and c6-c remains also a problem for the logarithmic approach although the gap was closed for c6-b.

7 Future Research Directions

Let us discuss (potentially) promising future research directions and open problems for piecewise linear approximator systems for MINLP problems involving two- or higher-dimensional functions:

Two dimensional specifics:

- Develop an algorithm which can compute *provable* optimal triangulations to any accuracy specified by δ.

- Rectangles and specific convex polygons could be an alternative to triangles.

Higher (≥ 2) dimensional:

- Generalize the concepts of triangulations to higher dimensions $n, n \geq 3$.

- An unsolved, probably problem-specific question to analyze and to answer is whether it is more efficient to approximate separable n-dimensional functions (*i.e.*, functions which can be transformed using the rules in Table 1) directly with an n-dimensional approach or to separate the function into lower-dimensional

functions applying appropriate lower-dimensional approaches.

– When using the transformations of Table 1, then one faces the problem of choosing the individual approximation errors δ_i. For our computations, we have chosen them equally. An optimal selection of δ_i's leading to a piecewise linear function requiring the least number of breakpoints for a given accuracy δ is an interesting problem in this context.

– Finally, we want to development explicit, piecewise linear formulations of functions $f(x)$: $\mathbb{R}^n \to \mathbb{R}$, $n \geq 2$, that are defined only defined on orthogonal or irregular grids of vertex points, but are not available in a closed algebraic form. This is an interesting problem relevant to various situations and industries. Such situations occur if the functions are evaluated by complex black box models involving, for instance, differential equations, or if the functions have been established only by experiments or observations leading to look-up tables, which in turn lead to non-smooth, piecewise-affine functions defined by linear interpolation between vertex points. An important subtask is also to reduce the system of grid points, *i.e.*, to replace is by a coarser grid which, relative to the system of given grid points, preserves δ-accuracy.

8 Conclusions

For bivariate nonlinear, convex and nonconvex functions, we automatically generate triangulations for continuous piecewise linear approximations as well as over- and underestimators satisfying a specified accuracy. Unlike in paper I (Rebennack & Kallrath, 2012, [15]), in this two-dimensional case we are not able to compute optimal triangulations, *i.e.*, we cannot make any safe statement whether the number of triangles is minimal or not. However, the refinement techniques we exploit in the construction of the triangles gives us some hope that we are not too far away from the minimal number of triangles.

The methods we have developed involve solving nonlinear problems to global optimality. They are useful to replace nonlinear terms in large MINLP problems which are mostly dominated by mixed-integer linear terms. This allows to solve approximately a significant subset of NLP or MINLP just by MILP solvers. Note that the computation of the triangulation systems is not restricted by available CPU time as they are computed only once a priori to their usage in the large scale MILP which are very well restricted in the available CPU time.

The automatic refinement triangulation provides an alternative to apply separation or transformation techniques applied to bivariate functions followed by one-dimensional piecewise linear approximation. We discuss and analyze the trade-off between one-dimensional and two-dimensional approaches, and also provide transformations for functions depending on more than two variables.

Acknowledgements We thank Timo Lohmann and Greg Steeger for their proof-reading, and GAMS GmbH (Cologne), Linus Schrage (Chicago) for providing an evaluation license of the global solver Lin-

doGlobal, and Christodoulos A. Floudas (Princeton) for providing an evaluation license of the global solver GloMIQO.

References

1. A. Brooke, D. Kendrick, and A. Meeraus. *GAMS – A User's Guide (Release 2.25)*. Boyd & Fraser Publishing Company, Danvers, Massachusetts, 1992.
2. M. R. Bussieck and A. Meeraus. General Algebraic Modeling System (GAMS). In J. Kallrath, editor, *Modeling Languages in Mathematical Optimization*, pages 137–157. Kluwer Academic Publishers, Norwell, MA, 2004.
3. C. D'Ambrosio, A. Lodi, and S. Martello. Piecewise Linear Approximation of Functions of Two Variables in MILP Models. *Operations Research Letters*, 38:39–46, 2010.
4. A. S. Drud. CONOPT - A Large-Scale GRG Code. *ORSA Journal of Computing*, 6(2):207–218, 1994.
5. B. Geißler. *Towards Globally Optimal Solutions for MINLPs by Discretization Techniques with Applications in Gas Network Optimization*. Dissertation, Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen-Nürnberg, Germany, 2011.
6. B. Geißler, A. Martin, A. Morsi, and L. Schewe. Using Piecewise Linear Functions for Solving MINLPs. In J. Lee and S. Leyffer, editors, *Mixed Integer Nonlinear Programming*, volume 154 of *The IMA Volumes in Mathematics and its Applications*, pages 287–314. Springer, 2012.
7. R. Horst, P. M. Pardalos, and N. V. Thoai. *Introduction to Global Optimization*. Kluwer Academic Publishers, 2nd edition, 2000.
8. J. Kallrath. Solving Planning and Design Problems in the Process Industry Using Mixed Integer and Global Optimization. *Annals of Operations Research*, 140:339–373, 2005.
9. J. Kallrath. Combining Strategic Design and Operative Planning Problems in the Process Industry. In L. G. Papageorgiou and M. C. Georgiadis, editors, *Supply Chain Optimizations - Part I*, pages 219–243. Wiley-VCH, Wiesbaden, Germany, 2007.
10. J. Kallrath. Cutting Circles and Polygons from Area-Minimizing Rectangles. *Journal of Global Optimization*, 43:299–328, 2009.
11. J. Kallrath and T. I. Maindl. *Real Optimization with SAP-APO*. Springer, Heidelberg, Germany, 2006.
12. J. Linderoth. A Simplicial Branch-and-Bound Algorithm for Solving Quadratically Constrained Quadratic Programs. *Mathematical Programming Ser. B*, 103:251–282, 2005.
13. R. Misener and C. A. Floudas. Piecewise-Linear Approximations of Multidimensional Functions. *Journal of Optimization Theory and Applications*, 145:120–147, 2010.
14. R. Misener and C. A. Floudas. GloMIQO: Global Mixed-Integer Quadratic Optimizer. *Journal of Global Optimization*, in print, 2012.
15. S. Rebennack and J. Kallrath. Continuous Piecewise Linear δ-Approximations for MINLP Problems. I. Minimal Breakpoint Systems for Univariate Functions. submitted, 2012.
16. S. Rebennack, J. Kallrath, and P. M. Pardalos. Energy Portfolio Optimization for Electric Utilities: Case Study for Germany. In E. Bjorndal, M. Bjorndal, P. M. Pardalos, and M. Rönnqvist, editors, *Energy, Natural Resources and Environmental Economics*, Energy Systems, pages 221–246. Springer, Berlin, 2010.
17. L. Schrage. LindoSystems: LindoAPI, 2004.
18. C. Timpe and J. Kallrath. Optimal Planning in Large Multi-Site Production Networks. *European Journal of Operational Research*, 126(2):422–435, 2000.
19. J. P. Vielma and G. Nemhauser. Modeling Disjunctive Constraints with a Logarithmic Number of Binary Variables and Constraints. *Mathematical Programming*, 128:49–72, 2011.