RAIRO Operations Research
Will be set by the publisher

REFORMULATIONS IN MATHEMATICAL PROGRAMMING: DEFINITIONS AND SYSTEMATICS

Leo Liberti[1]

Abstract. A reformulation of a mathematical program is a formulation which shares some properties with, but is in some sense better than, the original program. Reformulations are important with respect to the choice and efficiency of the solution algorithms; furthermore, it is desirable that reformulations can be carried out automatically. Reformulation techniques are very common in mathematical programming but interestingly they have never been studied under a common framework. This paper attempts to move some steps in this direction. We define a framework for storing and manipulating mathematical programming formulations, give several fundamental definitions categorizing reformulations in essentially four types (opt-reformulations, narrowings, relaxations and approximations). We establish some theoretical results and give reformulation examples for each type.

Keywords: reformulation, formulation, model, linearization, mathematical program

Mathematics Subject Classification. 90C11, 90C26, 90C27, 90C30, 90C99

February 22, 2008.

[1] LIX, Ecole Polytechnique, F-91128 Palaiseau, France liberti@lix.polytechnique.fr

Résumé. Une reformulation d'un programme mathématique est une formulation qui a des propriétés communes avec la formulation originelle, mais qui apporte des améliorations. Les reformulations sont importantes pour le meilleur choix d'algorithme de résolution, surtout quand il est possible de les accomplir automatiquement. Les techniques de reformulation sont très communes dans la programmation mathématique, et néanmoins elles n'ont jamais étées etudiées de façon unifiée. Le but de cet article est de definir des structures de données pour stocker et manipuler les formulations de programmation mathématique, de donner des definitions fondamentales qui categorisent les reformulations dans essentiellement quatre types (opt-reformulations, rétrécissements, relaxations et approximations), d'établir des résultats théoriques généraux, et de donner des examples de reformulations de chaque type.

1. Introduction

Mathematical programming is a descriptive language used to formalize optimization problems by means of parameters, decision variables, objective functions and constraints. Such diverse settings as combinatorial, integer, continuous, linear and nonlinear optimization problems can be defined precisely by their corresponding mathematical programming formulations. Its power is not limited to its expressiveness; it also often allows hassle-free solution of the problem. Most general-purpose solution algorithms solve optimization problems cast in their mathematical programming formulation, and the corresponding implementations are typically hooked into modelling environments which allow the user to input and solve complex optimization problems easily. It is well known that several different formulations may share the same numerical properties (feasible region, optima) though some of them are easier to solve than others with respect to the most efficient available algorithms. Being able to cast the problem in the best possible formulation is therefore a crucial aspect of any solution process.

When a problem with a given formulation P is cast into a different formulation Q, we say that Q is a reformulation of P. Curiously, the term "reformulation" appears in conjunction with "mathematical programming" over 400,000 times on Google; and yet there are surprisingly few attempts to formally define what a reformulation in mathematical programming actually is [10,71]. Further motives in support of a unified study of reformulations in mathematical programming are that there is a remarkable lack of literature reviews on the topic of reformulations [46] and that modelling languages such as AMPL [24] or GAMS [17] offer very limited automatic reformulation capabilities.

The importance that reformulations have in mathematical programming (specially automatic reformulations) cannot be underestimated. Solution algorithms usually target problems belonging to different classes, such as Linear Programming

(LP), Mixed-Integer Linear Programming (MILP), convex Nonlinear Programming (cNLP), Nonlinear Programming (NLP), convex Mixed-Integer Nonlinear Programming (cMINLP), Mixed-Integer Nonlinear Programming (MINLP). Typically, solution algorithms require their input problems to be cast in a particular form, called the *standard form* with respect to the algorithm. For example, the simplex algorithm [18] requires problems cast in LP standard form (that is, subject linear equality constraints only). Being able to cast a problem to a standard form for which there is an efficient algorithm is a a pre-processing step, but it is fundamental part of the solution process.

Furthermore, there often exist different formulations of a given problem that can all be cast to the same standard form, whilst the corresponding solution algorithm yields remarkably different solution performances depending on which formulation is used. One example of this is provided by the reduced RLT constraints reformulation [48, 56], which reformulates a possibly nonconvex quadratic nonlinear programming (NLP) problem with linear equality constraints to a different quadratic NLP with more linear equality constraints and fewer quadratic terms than in the original problem. Since the reformulated problem has fewer quadratic terms, its convex relaxation is tighter: hence, any spatial Branch-and-Bound (sBB) algorithm for nonconvex quadratic programming based on finding bounds at each node through the convex relaxation of the problem [49, 79] will improve its performance when acting on the reformulation. Yet, both the problem and the reformulation belong to the same class of problems and have the same standard form.

Reformulation is also useful within commercial-grade optimization software from the end user's point of view: if the software is capable of automatically reformulating a given problem to a more convenient form, the modeller need not be aware of all the solution algorithm details (and relative standard forms) embedded in the software. Limited to linear and convex quadratic problems, this is evident both in CPLEX [41] and XPress-MP [31] at the solver level; more in general, it occurs in AMPL [24], GAMS [17] and other mathematical programming language environments at the modelling level.

Finally, some algorithms do not only employ reformulations as a pre-processing phase needed to simplify the problem or cast it in a particular standard form, but actually use reformulation in an iterative way. In the MILP world, the extremely successful Branch-and-Cut algorithm can be seen as an iterative reformulation algorithm insofar as cutting planes and tightened formulations are considered as reformulations [96]. An effective and novel NLP solution algorithm based on blending two completely different formulations for the same problem is described in [65].

The mathematical programming glossary [84] defines "reformulation" as "obtaining a new formulation of a problem that is in some sense better, but equivalent to a given formulation." The main purpose of this paper is to give a precise meaning to this definition. We propose a data structure for storing and manipulating mathematical programming formulations, give several definitions linked to the concept of reformulation (with some associated theoretical results), and describe some of the commonly known reformulations and relaxations within the proposed

framework by means of symbolic algorithms. The theory is validated by some examples.

The original content of this paper is a unified "theory of reformulations" that attempts to provide tools for the systematic study of reformulations and their algorithmic implementation. This paper also serves as the basis for a forthcoming reformulation software that will be able to automatically apply reformulations to given mathematical programming problems. Aside from the few reformulation examples given in this paper, an extended reformulation library described within the proposed framework can be found in [51] (`http://www.lix.polytechnique.fr/~liberti/hdr.pdf`).

The rest of this paper is organized as follows. In Sect. 2 we briefly review the formal definitions of reformulations found in the literature. Sect. 3 describes the theoretical framework for handling formulations and reformulations. Sect. 4 contains examples of reformulations within the given framework.

2. Existing work

2.1. Definitions

The general consensus on the term "reformulation" in the field of mathematical programming seems to be that given a formulation of an optimization problem, a *reformulation* is a different formulation having the same set of optima. Various authors make use of this (informal) definition without actually making it explicit, among which [16, 30, 42, 56, 68, 78, 83, 92]. Many of the proposed reformulations, however, stretch this implicit definition somewhat. Liftings, for example (which consist in adding variables to the problem formulation), usually yield reformulations where an optimum in the original problem is mapped to a set of optima in the reformulated problem. Furthermore, it is sometimes noted how a reformulation in this sense is overkill because the reformulation only needs to hold at global optimality [1]. Furthermore, reformulations sometimes really refer to a change of variables, as is the case in [65].

Sherali [71] proposes the following definition.

Definition 2.1. *A reformulation in the sense of Sherali of an optimization problem P (with objective function f_P) is a problem Q (with objective function f_Q) such that there is a pair (σ, τ) where σ is a bijection between the feasible region of Q and that of P, and τ is a monotonic univariate function with $f_Q = \tau(f_P)$.*

This definition imposes very strict conditions by requiring the existence of the bijection σ, resulting in many useful problem transformations, which should otherwise be described as reformulations, not to satisfy the definition. Under some regularity conditions on σ, however, this definition does present some added benefits, such as e.g. allowing easy correspondences between partitioned subspaces of the feasible regions and mapping sensitivity analysis results from reformulated to original problem.

Hansen and co-workers [10] borrow some tools from complexity theory to propose the following definition.

Definition 2.2. *Let P_A and P_B be two optimization problems. A reformulation in the sense of Hansen $B(\cdot)$ of P_A as P_B is a mapping from P_A to P_B such that, given any instance A of P_A and an optimal solution of $B(A)$, an optimal solution of A can be obtained within a polynomial amount of time.*

In [10], this definition is used to establish a complexity comparison between different problem classes (specifically, BiLevel Programming (BLP) and Mixed-Integer Programming (MIP)) based on solution via a Branch-and-Bound (BB) algorithm. It turns out that a BB algorithm applied to MIP can be mapped precisely into one applied to BLP, thus allowing the authors to conclude that BLP is practically at least as difficult as MIP, and not just from a worst-case complexity viewpoint. On the other hand, requiring a polynomial time reformulation can be just too slow practically, or might prevent non-polynomial time reformulations to belong to this class even when they might be carried out within practically reasonable amounts of time. Furthermore, a reformulation in the sense of Hansen does not necessarily preserve local optimality or the number of global optima, which might in some cases be a desirable reformulation feature.

2.2. Reformulations

The term "reformulation" in MILPs mostly refers to preprocessing simplifications (of the type implemented in most good level MILP solvers, such as e.g. CPLEX [41]) and cutting planes [92], although a considerable number of standard forms can be transformed to MILPs [10, 26, 42, 70]. Binary Quadratic Programs (BQP) have attracted a considerable amount of attentions, mostly because of the fact that there is an easy exact linearization [22] that presents some practical drawbacks. Extensive work was carried out by research teams led by A. Billionnet [13], F. Glover [1], P. Hammer [35], P. Hansen [36] (see Sect. 4.1), Ph. Michelon [32], H. Sherali [76] (see Sect. 4.4). We refer the reader to three recent papers [14, 33, 37] and the references contained therein for a more complete picture of reformulations for BQPs. Reformulations of polynomial [34, 57, 73, 93] and posynomial [75, 95] programs also attracted considerable attention. Most work in geometric programming rests on a convex reformulation [40]; a symbolic method to model problems so that the corresponding mathematical program is convex is described in [30]. Reformulations are used within algorithms [65], specially in decomposition-based ones [7]. An interesting reformulation of a robust LP to a simple LP, involving the LP dual, is given in [12]. Reformulations are very common within applications, to treat problems with certain determined structures [8, 9, 11, 19, 52, 53].

3. Reformulation theory

In this section we give a formal definition of a mathematical programming formulation in such terms that can be easily implemented on a computer. We refer to a mathematical programming problem in the most general form:

$$\left.\begin{array}{lrcl} \min & f(x) & & \\ & g(x) & \lesseqgtr & b \\ & x & \in & X, \end{array}\right\} \tag{1}$$

where f, g are function sequences of various sizes, b is an appropriately-sized real vector, and X is a cartesian product of continuous and discrete intervals.

Formulation (1) above is actually a formulation schema, i.e. it represents different formulations according to the form of f, g, x, X. We remark that (1) may represent a *structured* formulation (i.e. one where f, g involve quantifiers $\forall, \sum, \prod$) or a *flat* formulation (i.e. one where no such quantifier appears). Formulations are usually given by researchers and practitioners in structured form, and later automatically translated by modelling software such as AMPL [24] or GAMS [17] in flat form. This translation is necessary as almost all solution algorithm implementations take their input in some flat form variant.

The rest of this section is organized as follows. In Subsect. 3.1 we formally describe a data structure for storing mathematical programming formulations and give some examples. This will be used in the sequel to state symbolic reformulation algorithms unambiguously. In Subsect. 3.3 we define the fundamental notion on which our reformulation systematics is based, i.e. that of auxiliary problems. Auxiliary problems of various categories and their properties are defined in Subsections 3.4-3.8.

3.1. A data structure for formulations

Having a well-defined, unified data structure for formulations is important for two reasons: firstly, it allows us to unambiguously state symbolic reformulation algorithms acting on the formulation. Secondly, it provides a bridge between structured formulations and flat formulations. Having said that, the precise form of the data structure is not of crucial importance: there are many available alternatives, some of which are openly documented (e.g. the Optimization Services project [25] in COIN-OR [58]) and some others which are not (e.g. the internal memory representation used by AMPL [23,24,27]). The data structure we propose is the basis for the Moron optimization software framework [49] and is semantically equivalent to the `InstanceData` class within Optimization Services (see [25], p. 30).

Our definition lists the following primary formulation elements: parameters and variables (with types and bounds); expressions that depend on parameters and variables; objective functions and constraints depending on the expressions. We let $\mathbb{P}$ be the set of all mathematical programming formulations and $\mathbb{M}$ be the set of all matrices. This is used in Defn. 3.1 to define leaf nodes in mathematical expression

trees, so that the concept of a formulation can also accommodate multilevel and semidefinite programming problems.

Definition 3.1. *Given an alphabet $\mathcal{L}$ consisting of countably many alphanumeric names $N_{\mathcal{L}}$ and operator symbols $O_{\mathcal{L}}$, a mathematical programming formulation P is a 7-tuple $(\mathcal{P}, \mathcal{V}, \mathcal{E}, \mathcal{O}, \mathcal{C}, \mathcal{B}, \mathcal{T})$, where:*

- *$\mathcal{P} \subseteq N_{\mathcal{L}}$ is the sequence of parameter symbols: each element $p \in \mathcal{P}$ is a parameter name;*
- *$\mathcal{V} \subseteq N_{\mathcal{L}}$ is the sequence of variable symbols: each element $v \in \mathcal{V}$ is a variable name;*
- *$\mathcal{E}$ is the set of expressions: each element $e \in \mathcal{E}$ is a directed tree $e = (V_e, A_e)$ such that:*
  (a) *$V_e \subseteq \mathcal{L}$ is a finite set*
  (b) *there is a unique vertex $r_e \in V_e$ such that $\delta^-(r_e) = \emptyset$ (such a vertex is called the root vertex)*
  (c) *all vertices $v \in V_e$ such that $\delta^+(v) = \emptyset$ (called leaf vertices — their set is denoted by $\lambda(e)$) are such that $v \in \mathcal{P} \cup \mathcal{V} \cup \mathbb{R} \cup \mathbb{P} \cup \mathbb{M}$*
  (d) *for all $v \in V_e$ such that $\delta^+(v) \neq \emptyset$, $v \in O_{\mathcal{L}}$*
  (e) *two weightings $\chi, \xi : V_e \to \mathbb{R}$ are defined on V_e: $\chi(v)$ is the node coefficient and $\xi(v)$ is the node exponent of the node v.*

  Elements of $\mathcal{E}$ are sometimes called expression trees; nodes $v \in O_{\mathcal{L}}$ represent an operation on the nodes in $\delta^+(v)$, denoted by $v(\delta^+(v))$, with output in $\mathbb{R}$;
- *$\mathcal{O} \subseteq \{-1, 1\} \times \mathcal{E}$ is the sequence of objective functions; each objective function $o \in \mathcal{O}$ has the form (d_o, f_o) where $d_o \in \{-1, 1\}$ is the optimization direction (-1 stands for minimization, $+1$ for maximization) and $f_o \in \mathcal{E}$;*
- *$\mathcal{C} \subseteq \mathcal{E} \times \mathcal{S} \times \mathbb{R}$ (where $\mathcal{S} = \{-1, 0, 1\}$) is the sequence of constraints c of the form (e_c, s_c, b_c) with $e_c \in \mathcal{E}, s_c \in \mathcal{S}, b_c \in \mathbb{R}$:*

$$c \equiv \begin{cases} e_c \leq b_c & \text{if} \quad s_c = -1 \\ e_c = b_c & \text{if} \quad s_c = 0 \\ e_c \geq b_c & \text{if} \quad s_c = 1; \end{cases}$$

- *$\mathcal{B} \subseteq \mathbb{R}^{|\mathcal{V}|} \times \mathbb{R}^{|\mathcal{V}|}$ is the sequence of variable bounds: for all $v \in \mathcal{V}$ let $\mathcal{B}(v) = [L_v, U_v]$ with $L_v, U_v \in \mathbb{R}$;*
- *$\mathcal{T} \subseteq \{0, 1, 2\}^{|\mathcal{V}|}$ is the sequence of variable types: for all $v \in \mathcal{V}$, v is called a continuous variable if $\mathcal{T}(v) = 0$, an integer variable if $\mathcal{T}(v) = 1$ and a binary variable if $\mathcal{T}(v) = 2$.*

We remark that for a sequence of variables $z \subseteq \mathcal{V}$ we write $\mathcal{T}(z)$ and respectively $\mathcal{B}(z)$ to mean the corresponding sequences of types and respectively bound intervals of the variables in z. Given a formulation $P = (\mathcal{P}, \mathcal{V}, \mathcal{E}, \mathcal{O}, \mathcal{C}, \mathcal{B}, \mathcal{T})$, the *cardinality* of P is $|P| = |\mathcal{V}|$. We sometimes refer to a formulation by calling it an *optimization problem* or simply a *problem*.

Consider a mathematical programming formulation $P = (\mathcal{P}, \mathcal{V}, \mathcal{E}, \mathcal{O}, \mathcal{C}, \mathcal{B}, \mathcal{T})$ and a function $x : \mathcal{V} \to \mathbb{R}^{|\mathcal{V}|}$ (called *point*) which assigns values to the variables.

A point x is *type feasible* if: $x(v) \in \mathbb{R}$ when $\mathcal{T}(v) = 0$, $x(v) \in \mathbb{Z}$ when $\mathcal{T}(v) = 1$, $x(v) \in \{L_v, U_v\}$ when $\mathcal{T}(v) = 2$, for all $v \in \mathcal{V}$; x is *bound feasible* if $x(v) \in \mathcal{B}(v)$ for all $v \in \mathcal{V}$; x is *constraint feasible* if for all $c \in \mathcal{C}$ we have: $e_c(x) \leq b_c$ if $s_c = -1$, $e_c(x) = b_c$ if $s_c = 0$, and $e_c(x) \geq b_c$ if $s_c = 1$. A point x is *feasible in* P if it is type, bound and constraint feasible. A point x feasible in P is also called a *feasible solution* of P. A point which is not feasible is called *infeasible*. Denote by $\mathcal{F}(P)$ the feasible points of P. A feasible point x is a *local optimum* of P with respect to the objective $o \in \mathcal{O}$ if there is a non-empty neighbourhood N of x such that for all feasible points $y \neq x$ in N we have $d_o f_o(x) \geq d_o f_o(y)$. A local optimum is *strict* if $d_o f_o(x) > d_o f_o(y)$. A feasible point x is a *global optimum* of P with respect to the objective $o \in \mathcal{O}$ if $d_o f_o(x) \geq d_o f_o(y)$ for all feasible points $y \neq x$. A global optimum is *strict* if $d_o f_o(x) > d_o f_o(y)$. Denote the set of local optima of P by $\mathcal{L}(P)$ and the set of global optima of P by $\mathcal{G}(P)$. If $\mathcal{O}(P) = \emptyset$, we define $\mathcal{L}(P) = \mathcal{G}(P) = \mathcal{F}(P)$.

Example 3.2. A quadratic optimization problem. *This example illustrates how flat form formulations can be embedded in the proposed data structure. Consider the problem of minimizing the quadratic form* $3x_1^2 + 2x_2^2 + 2x_3^2 + 3x_4^2 + 2x_5^2 + 2x_6^2 - 2x_1x_2 - 2x_1x_3 - 2x_1x_4 - 2x_2x_3 - 2x_4x_5 - 2x_4x_6 - 2x_5x_6$ *subject to* $x_1 + x_2 + x_3 + x_4 + x_5 + x_6 = 0$ *and* $x_1, \ldots, x_6 \in \{-1, 1\}$. *For this problem,*

- $\mathcal{P} = \emptyset$*;*
- $\mathcal{V} = (x_1, x_2, x_3, x_4, x_5, x_6)$*;*
- $\mathcal{E} = (e_1, e_2)$ *where* e_1, e_2 *are the graphs shown in Fig. 1;*
- $\mathcal{O} = (-1, e_1)$*;*
- $\mathcal{C} = ((e_2, 0, 0))$*;*
- $\mathcal{B} = ([-1, 1], [-1, 1], [-1, 1], [-1, 1], [-1, 1], [-1, 1])$*;*
- $\mathcal{T} = (2, 2, 2, 2, 2, 2)$.

Example 3.3. Balanced Graph Bisection Problem. *In this example we consider a structured formulation. Example 3.2 is the (scaled) mathematical programming formulation of a balanced graph bisection problem instance. This problem is defined as follows.*

> Balanced Graph Bisection Problem *(BGBP). Given an undirected graph* $G = (V, E)$ *without loops or parallel edges such that* $|V|$ *is even, find a subset* $U \subset V$ *such that* $|U| = \frac{|V|}{2}$ *and the set of edges* $C = \{\{u, v\} \in E \mid u \in U, v \notin U\}$ *is as small as possible.*

The problem instance considered in Example 3.2 is shown in Fig. 2. To all vertices $i \in V$ *we associate variables* $x_i = \begin{cases} 1 & i \in U \\ 0 & i \notin U \end{cases}$. *The number of edges in* C *is counted by* $\frac{1}{4} \sum_{\{i,j\} \in E} (x_i - x_j)^2$. *The fact that* $|U| = \frac{|V|}{2}$ *is expressed by requiring an equal number of variables at 1 and -1, i.e.* $\sum_{i=1}^{6} x_i = 0$. *We can also express the problem*

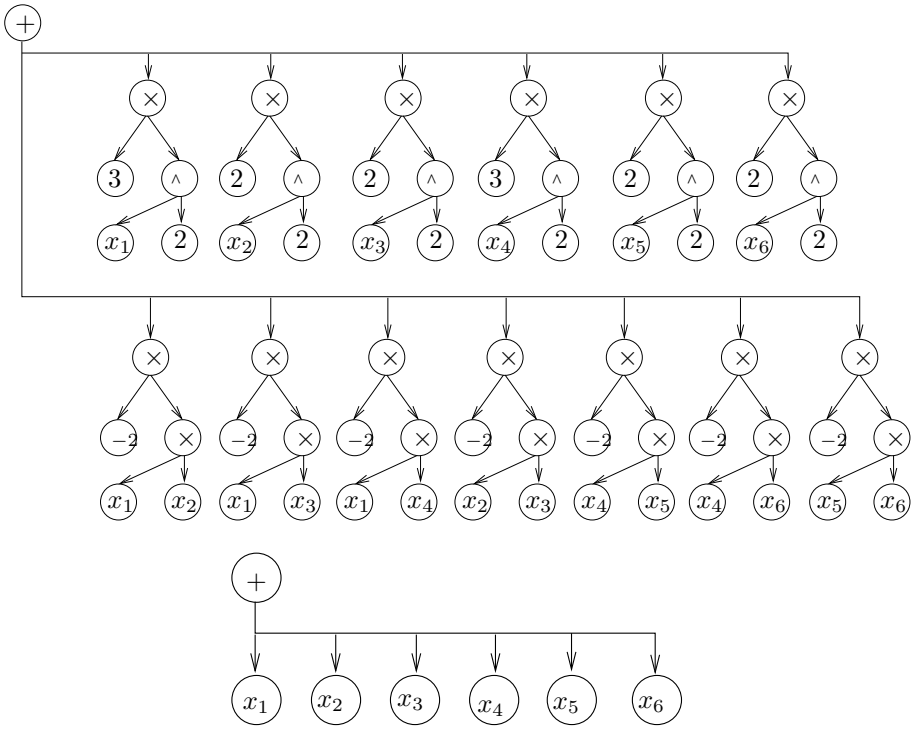

FIGURE 1. The graphs e_1 (above) and e_2 (below) from Example 3.2.

in Example 3.2 as a particular case of the more general optimization problem:

$$\left.\begin{array}{rrcl} \min_x & x^\top Lx & & \\ s.t. & x\mathbf{1} & = & 0 \\ & x & \in & \{-1,1\}^6, \end{array}\right\}$$

where

$$L = \begin{pmatrix} 3 & -1 & -1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ -1 & -1 & 2 & 0 & 0 & 0 \\ -1 & 0 & 0 & 3 & -1 & -1 \\ 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & -1 & 2 \end{pmatrix}$$

and $\mathbf{1} = (1,1,1,1,1,1)^\top$. *We represent this class of problems by the following mathematical programming formulation:*

- $\mathcal{P} = (L_{ij} \mid 1 \le i,j \le 6)$*;*
- $\mathcal{V} = (x_1, x_2, x_3, x_4, x_5, x_6)$*;*
- $\mathcal{E} = (e_1', e_2)$ *where* e_1' *is shown in Fig. 3 and* e_2 *is shown in Fig. 1 (below);*

- $\mathcal{O} = (-1, e_1')$;
- $\mathcal{C} = ((e_2, 0, 0))$;
- $\mathcal{B} = ([-1, 1], [-1, 1], [-1, 1], [-1, 1], [-1, 1], [-1, 1])$;
- $\mathcal{T} = (2, 2, 2, 2, 2, 2)$.

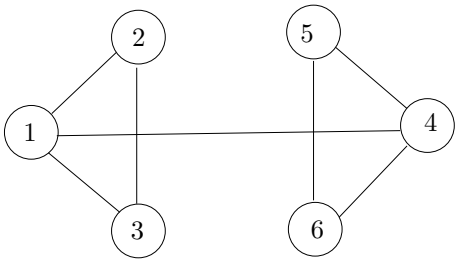

FIGURE 2. The BGBP instance in Example 3.2.

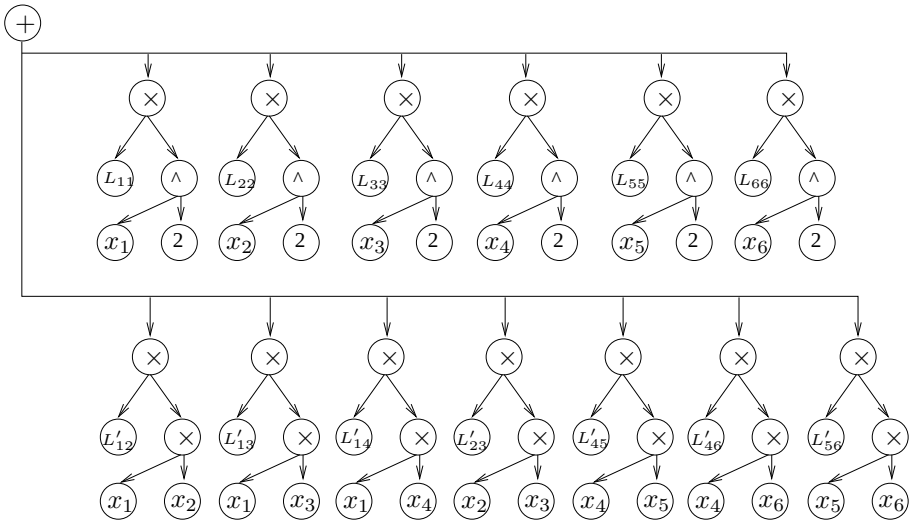

FIGURE 3. The graph e_1' from Example 3.3. $L'_{ij} = L_{ij} + L_{ji}$ for all i, j.

3.2. STANDARD FORMS

Solution algorithms for mathematical programming problems read a formulation as input and attempt to compute an optimal feasible solution as output. Naturally, algorithms that exploit problem structure are usually more efficient than those that do not. In order to be able to exploit the structure of the problem,

solution algorithms solve problems that are cast in a *standard form* that emphasizes the useful structure. A good reformulation framework should be aware of the available solution algorithms and attempt to reformulate given problems into the most appropriate standard form. In this section we review the most common standard forms.

3.2.1. *Linear Programming*

A mathematical programming problem P is a Linear Programming (LP) problem if (a) $|\mathcal{O}| = 1$ (i.e. the problem only has a single objective function); (b) e is a linear form for all $e \in \mathcal{E}$; and (c) $\mathcal{T}(v) = 0$ (i.e. v is a continuous variable) for all $v \in \mathcal{V}$.

3.2.2. *Mixed Integer Linear Programming*

A mathematical programming problem P is a Mixed Integer Linear Programming (MILP) problem if (a) $|\mathcal{O}| = 1$; and (b) e is a linear form for all $e \in \mathcal{E}$.

3.2.3. *Nonlinear Programming*

A mathematical programming problem P is a Nonlinear Programming (NLP) problem if (a) $|\mathcal{O}| = 1$ and (b) $\mathcal{T}(v) = 0$ for all $v \in \mathcal{V}$. Many fundamentally different solution algorithms are available for solving NLPs, and most of them require different standard forms. One of the most widely used is Sequential Quadratic Programming (SQP) [21, 29], which requires problem constraints $c \in \mathcal{C}$ to be expressed in the form $l_c \leq c \leq u_c$ with $l_c, u_c \in \mathbb{R} \cup \{-\infty, +\infty\}$.

3.2.4. *Mixed Integer Nonlinear Programming*

A mathematical programming problem P is a Mixed Integer Nonlinear Programming (MINLP) problem if $|\mathcal{O}| = 1$. The situation as regards MINLP standard forms is generally the same as for NLPs, save that a few more works have appeared in the literature about standard forms for MINLPs [49, 67, 82, 83]. In particular, the Smith standard form [83] is purposefully constructed so as to make symbolic manipulation algorithms easy to carry out on the formulation. A MINLP is in Smith standard form if:

- $\mathcal{O} = \{d_o, e_o\}$ where e_o is a linear form;
- $\mathcal{C}$ can be partitioned into two sets of constraints $\mathcal{C}_1, \mathcal{C}_2$ such that c is a linear form for all $c \in \mathcal{C}_1$ and $c = (e_c, 0, 0)$ for $c \in \mathcal{C}_2$ where e_c is as follows:
  (1) $r(e_c)$ is the sum operator
  (2) $\delta^+(r(e_c)) = \{\otimes, v\}$ where (a) $\otimes$ is a nonlinear operator where all subnodes are leaf nodes, (b) $\chi(v) = -1$ and (c) $v \in \mathcal{V}$.

Essentially, the Smith standard form consists of a linear part comprising objective functions and a set of constraints; the rest of the constraints have a special form $\otimes(x, y) - v = 0$, with $v, x, y \in \mathcal{V}(P)$ and $\otimes$ a nonlinear operator in $O_{\mathcal{L}}$. By grouping all nonlinearities in a set of equality constraints of the form "variable = operator(variables)" (called *defining constraints*) the Smith standard form makes it easy to construct auxiliary problems. The Smith standard form can be constructed

by recursing on the expression trees of a given MINLP [81]. Solution algorithms for solving MINLPs are usually extensions of BB type algorithms [47,49,66,83,89].

3.2.5. *Separable problems*

A problem P is in separable form if (a) $\mathcal{O}(P) = \{(d_o, e_o)\}$, (b) $\mathcal{C}(P) = \emptyset$ and (c) e_o is such that:

- $r(e_o)$ is the sum operator
- for all distinct $u, v \in \delta^+(r(e_o))$, $\lambda(u) \cap \lambda(v) \cap \mathcal{V}(P) = \emptyset$.

The separable form is useful because it allows a very easy problem decomposition: for all $u \in \delta^+(r(e_o))$ it suffices to solve the smaller problems Q_u with $\mathcal{V}(Q) = \lambda(v) \cap \mathcal{V}(P)$, $\mathcal{O}(Q) = \{(d_o, u)\}$ and $\mathcal{B}(Q) = \{\mathcal{B}(P)(v) \mid v \in \mathcal{V}(Q)\}$. Then $\bigcup_{u \in \delta^+(r(e_o))} x(\mathcal{V}(Q_u))$ is a solution for P.

3.2.6. *Factorable problems*

A problem P is in factorable form [63, 80, 89, 94] if:

(1) $\mathcal{O} = \{(d_o, e_o)\}$
(2) $r(e_o) \in \mathcal{V}$ (consequently, the vertex set of e_o is simply $\{r(e_o)\}$)
(3) for all $c \in \mathcal{C}$:
   - $s_c = 0$
   - $r(e_c)$ is the sum operator
   - for all $t \in \delta^+(r(e_c))$, either (a) t is a unary operator and $\delta^+(t) \in \lambda(e_c)$ (i.e. the only subnode of t is a leaf node) or (b) t is a product operator with $\delta^+(t) = \{u, v\}$ such that u, v are both unary operators with only one leaf subnodes.

The factorable form is useful because it is easy to construct many auxiliary problems (including convex relaxations, [3, 63, 80]) from problems cast in this form. In particular, factorable problems can be reformulated to separable problems [63, 66, 89].

3.2.7. *D.C. problems*

The acronym "d.c." stands for "difference of convex". Given a set $\Omega \subseteq \mathbb{R}^n$, a function $f : \Omega \to \mathbb{R}$ is a *d.c. function* if it is a difference of convex functions, i.e. there exist convex functions $g, h : \Omega \to \mathbb{R}$ such that, for all $x \in \Omega$, we have $f(x) = g(x) - h(x)$. Let C, D be convex sets; then the set $C \backslash D$ is a *d.c. set*. An optimization problem is *d.c.* if the objective function is d.c. and Ω is a d.c. set. In most of the d.c. literature, however [39, 86, 91], a mathematical programming problem is d.c. if:

- $\mathcal{O} = \{(d_o, e_o)\}$;
- e_o is a d.c. function;
- c is a linear form for all $c \in \mathcal{C}$.

D.C. programming problems have two fundamental properties. The first is that the space of all d.c. functions is dense in the space of all continuous functions. This

implies that any continuous optimization problem can be approximated as closely as desired, in the uniform convergence topology, by a d.c. optimization problem [39, 91]. The second property is that it is possible to give explicit necessary and sufficient global optimality conditions for certain types of d.c. problems [86, 91]. Some formulations of these global optimality conditions [85] also exhibit a very useful algorithmic property: if at a feasible point x the optimality conditions do not hold, then the optimality conditions themselves can be used to construct an improved feasible point x'.

3.2.8. *Linear Complementarity problems*

Linear complementarity problems (LCP) are nonlinear feasibility problems with only one nonlinear constraint. A mathematical programming problem is defined as follows [28], p. 50:

- $\mathcal{O} = \emptyset$;
- there is a constraint $c' = (e, 0, 0) \in \mathcal{C}$ such that (a) $t = r(e)$ is a sum operator; (b) for all $u \in \delta^+(t)$, u is a product of two terms v, f such that $v \in \mathcal{V}$ and $(f, 1, 0) \in \mathcal{C}$;
- for all $c \in \mathcal{C} \setminus \{c'\}$, e_c is a linear form.

Essentially, an LCP is a feasibility problem of the form:

$$\left.\begin{array}{rcl} Ax & \geq & b \\ x & \geq & 0 \\ x^\top(Ax - b) & = & 0, \end{array}\right\}$$

where $x \in \mathbb{R}^n$, A is an $m \times n$ matrix and $b \in \mathbb{R}^m$. Many types of mathematical programming problems (including MILPs with binary variables [28, 42]) can be recast as LCPs or small extensions of LCP problems [42]. Furthermore, some types of LCPs can be reformulated to LPs [59] and as separable bilinear programs [60]. Certain types of LCPs can be solved by an interior point method [28, 43].

3.2.9. *Bilevel Programming problems*

The bilevel programming (BLP) problem consists of two nested mathematical programming problems named the *leader* and the *follower* problem. Formally, a BLP is a pair of formulations (L, F) (leader and follower) and a subset $\ell \neq \emptyset$ of the set of all leaf nodes of the expressions in $\mathcal{E}(L)$ such that any leaf node $v \in \ell$ has the form (x, F) where $x \in \mathcal{V}(F)$.

The usual mathematical notation is as follows [10, 20]:

$$\left.\begin{array}{rrlll} \min_y & F(x(y), y) & & & \\ & \min_x & f(x, y) & & \\ & \text{s.t.} & x \in X, & y \in Y, & \end{array}\right\} \tag{2}$$

where X, Y are arbitrary sets. This type of problem arises in economic applications. The leader knows the cost function of the follower, who may or may not know that of the leader; but the follower knows the optimal strategy selected by

the leader (i.e. the optimal values of the decision variables of L) and takes this into account to compute his/her own optimal strategy.

BLPs can be reformulated exactly to MILPs with binary variables and vice-versa [10], where the reformulation is as in Defn. 2.2. Furthermore, two typical Branch-and-Bound (BB) algorithms for the considered MILPs and BLPs have the property that the the MILP BB can be "embedded" in the BLP BB (this roughly means that the BB tree of the MILP is a subtree of the BB tree of the BLP); however, the contrary does not hold. This seems to hint at a practical solution difficulty ranking in problems with the same degree of worst-case complexity (both MILPs and BLPs are **NP**-hard).

3.2.10. *Semidefinite Programming problems*

Consider known symmetric $n \times n$ matrices C, A_k for $k \leq m$, a vector $b \in \mathbb{R}^m$ and a symmetric $n \times n$ matrix $X = (x_{ij})$ where x_{ij} is a problem variable for all $i, j \leq n$. The following is a *semidefinite programming problem* (SDP) in primal form:

$$\left.\begin{array}{rrcl} \min_X & C \bullet X & & \\ \forall k \leq m & A_k \bullet X & = & b_i \\ & X & \succeq & 0, \end{array}\right\} \tag{3}$$

where $X \succeq 0$ is a constraint that indicates that X should be positive semidefinite, and $A \bullet B = \mathrm{tr}(A^\top B)$. We also consider the SDP in dual form:

$$\left.\begin{array}{rrcl} \max_{y,S} & b^\top y & & \\ & \sum_{k \leq m} y_k A_k + S & = & C \\ & S & \succeq & 0, \end{array}\right\} \tag{4}$$

where S is a symmetric $n \times n$ matrix and $y \in \mathbb{R}^m$. Both forms of the SDP problem are convex NLPs, so the duality gap is zero. Both forms can be solved by a particular type of polynomial-time interior point method (IPM), which implies that solving SDPs is practically efficient [6, 90]. SDPs are important because they provide tight relaxations to (nonconvex) quadratically constrained quadratic programming problems (QCQP), i.e. problems with a quadratic objective and quadratic constraints.

SDPs can be easily modelled with the data structure described in Defn. 3.1, for their expression trees are linear forms where each leaf node contains a symmetric matrix. There is no need to explicitly write the semidefinite constraints $X \succeq 0, S \succeq 0$ because the IPM solution algorithms will automatically find optimal X, S matrices that are semidefinite.

3.3. Auxiliary problems

If two problems are related, we say that one is an auxiliary problem of the other. We sometimes refer to auxiliary problems with the term "problem transformation" or the generic term of "reformulation".

Definition 3.4. *Any problem Q that is related to a given problem P by a formula $f(Q, P) = 0$ is called an auxiliary problem with respect to P.*

Among the several possible auxiliary problem types, four are specially interesting and used quite commonly: transformations preserving all optimality properties (opt-reformulations); transformations preserving at least one global optimum (narrowings); transformations based on dropping constraints, variable bounds or types (relaxations); transformations that are one of the above types "in the limit" (approximations).

3.4. Opt-reformulations

Opt-reformulations are auxiliary problems that preserve all optimality information. We define them by considering local and global optima.

Definition 3.5. *Q is a local reformulation of P if there is a function $\varphi : \mathcal{F}(Q) \to \mathcal{F}(P)$ such that (a) $\varphi(y) \in \mathcal{L}(P)$ for all $y \in \mathcal{L}(Q)$, (b) φ restricted to $\mathcal{L}(Q)$ is surjective. This relation is denoted by $P \prec_\varphi Q$.*

A local reformulation transforms all optima of the original problem into optima of the reformulated problem, although more than one reformulated optimum may correspond to the same original optimum. A local reformulation does not lose any local optimality information and makes it possible to map reformulated optima back to the original ones; on the other hand, a local reformulation does not keep track of globality: some global optima in the original problem may be mapped to local optima in the reformulated problem, or vice-versa.

Example 3.6. *Consider the problem P, Q as follows:*

$$\begin{aligned} P &\equiv \min_{x \in [-2\pi, 2\pi]} x + \sin(x) \\ Q &\equiv \min_{x \in [-2\pi, 2\pi]} \sin(x). \end{aligned}$$

It is easy to verify that there is a bijection between the local optima of P and those of Q. However, although P has a unique global optimum, every local optimum in Q is global.

Definition 3.7. *Q is a global reformulation of P if there is a function $\varphi : \mathcal{F}(Q) \to \mathcal{F}(P)$ such that (a) $\varphi(y) \in \mathcal{G}(P)$ for all $y \in \mathcal{G}(Q)$, (b) φ restricted to $\mathcal{G}(Q)$ is surjective. This relation is denoted by $P \lhd_\varphi Q$.*

A global reformulation transforms all global optima of the original problem into global optima of the reformulated problem, although more than one reformulated global optimum may correspond to the same original global optimum. Global reformulations are desirable, in the sense that they make it possible to retain the useful information about the global optima whilst ignoring local optimality. At best, given a difficult problem P with many local minima, we would like to find a global reformulation Q where $\mathcal{L}(Q) = \mathcal{G}(Q)$.

Example 3.8. *Consider a problem P with $\mathcal{O}(P) = \{f\}$. Let Q be a problem such that $\mathcal{O}(Q) = \{\breve{f}\}$ and $\mathcal{F}(Q) = conv(\mathcal{F}(P))$, where $conv(\mathcal{F}(P))$ is the convex hull of the points of $\mathcal{F}(P)$ and $\breve{f}$ is the convex envelope of f over the convex hull of $\mathcal{F}(P)$ (in other words, f is the greatest convex function underestimating f on $\mathcal{F}(P)$). Since the set of global optima of P is contained in the set of global optima of Q [38], the convex envelope is a global reformulation.*

Unfortunately, finding convex envelopes in explicit form is not easy. A considerable amount of work exists in this area: e.g. for bilinear terms [5, 63], trilinear terms [64], fractional terms [88], monomials of odd degree [44, 55] the envelope is known in explicit form (this list is not exhaustive). See [87] for recent theoretical results and further references.

We write $P \prec Q$ (resp. $P \lhd Q$) if there is a φ such that $P \prec_\varphi Q$ (resp. $P \lhd_\varphi Q$).

Definition 3.9. *Q is an opt-reformulation of P (denoted by $P < Q$) if $P \prec Q$ and $P \lhd Q$.*

This type of reformulation preserves both local and global optimality information. It turns out that several well-known reformulations in the literature are opt-reformulations.

Definition 3.10. *An exact linearization of a problem P is an opt-reformulation Q of P where all expressions $e \in \mathcal{E}(P)$ are linear forms.*

Lemma 3.11. *The relations $\prec, \lhd, <$ are reflexive and transitive, but in general not symmetric.*

Proof. For reflexivity, simply take φ as the identity. For transitivity, let $P \prec Q \prec R$ with functions $\varphi : \mathcal{F}(Q) \to \mathcal{F}(P)$ and $\psi : \mathcal{F}(R) \to \mathcal{F}(Q)$. Then $\vartheta = \varphi \circ \psi$ has the desired properties. In order to show that $\prec$ is not symmetric, consider a problem P with variables x and a unique minimum x^* and a problem Q which is exactly like P but has one added variable $w \in [0, 1]$ not appearing in objective functions or constraints. It is easy to show that $P \prec Q$ (take φ as the projection of (x, w) on x). However, since for all $w \in [0, 1]$ (x^*, w) is an optimum of Q, there is no function of a singleton to a continuously infinite set that is surjective, so $Q \not\prec P$. □

Given a pair of problems P, Q where $\prec, \lhd, <$ are symmetric on the pair, we call Q a *symmetric reformulation* of P.

The most important consequence of Lemma 3.11 is that we can compose elementary opt-reformulations together to create more complex opt-reformulations.

3.5. Change of variables

Continuous reformulations are based on a continuous map τ (invertible on the variable domains) acting on the continuous relaxation of the feasible space of the two problems.

Definition 3.12. *For P, Q having the following properties:*

(a) $|P| = n, |Q| = m$,
(b) $\mathcal{V}(P) = x, \mathcal{V}(Q) = y$,
(c) $\mathcal{O}(P) = (f, d), \mathcal{O}(Q) = (f', d')$ *where* f *is a sequence of expressions in* $\mathcal{E}(P)$ *and* d *is a vector with elements in* $\{-1, 1\}$ *(and similarly for* f', d'*),*
(d) $\mathcal{C}(P) = (g, -\mathbf{1}, \mathbf{0}), \mathcal{C}(Q) = (g', -\mathbf{1}, \mathbf{0})$ *where* g *is a sequence of expressions in* $\mathcal{E}(P)$*,* $\mathbf{0}$ *(resp.* $\mathbf{1}$*) is a vector of 0s (resp. 1s) of appropriate size (and similarly for* g'*),*
(e) f, f' *are continuous functions and* g, g' *are sequences of continuous functions,*

Q *is a continuous reformulation of* P *with respect to a reformulating bijection* τ *(denoted by* $P \approx_\tau Q$*) if* $\tau : \mathbb{R}^n \to \mathbb{R}^m$ *is a continuous map, invertible on the variable domains* $\prod_{x_i \in x} \mathcal{B}(x_i)$*, such that* $f' \circ \tau = f$*,* $g' \circ \tau = g$ *and* $\mathcal{B}(y) = \tau(\mathcal{B}(x))$*, and such that* τ^{-1} *is also continuous.*

It is easy to show that τ is an invertible map $\mathcal{F}(P) \to \mathcal{F}(Q)$. Changes of variables usually provide continuous reformulations [65]. Continuous reformulations are similar to reformulations in the sense of Sherali: they are stronger, in that they require the invertible mapping to be continuous; and they are weaker, in that they impose no additional condition on the way the objective functions are reformulated. We remark that $\approx_\tau$ is an equivalence relation.

Lemma 3.13. *If* $P \approx_\tau Q$ *with* $|P| = n, |Q| = m$*, for all* $x \in \mathbb{R}^n$ *which is bound and constraint feasible in* P*,* $\tau(x)$ *is bound and constraint feasible in* Q*.*

Proof. Suppose without loss of generality that the constraints and bounds for P can be expressed as $g(x) \leq 0$ for $x \in \mathbb{R}^n$ and those for Q can be expressed as $g'(y) \leq 0$ for $y \in \mathbb{R}^m$. Then $g'(y) = g'(\tau(x)) = (g' \circ \tau)(x) = g(x) \leq 0$. □

Proposition 3.14. *If* $P \approx_\tau Q$ *with* $\mathcal{V}(P) = x, \mathcal{V}(Q) = y$*,* $|P| = n, |Q| = m$*,* $|\mathcal{O}(P)| = |\mathcal{O}(Q)| = 1$ *such that* (f, d) *is the objective function of* P *and* (f', d') *is that of* Q*,* $d = d'$*,* $\mathcal{T}(x) = 0, \mathcal{T}(y) = 0$*, then* τ *is a bijection* $\mathcal{L}(P) \to \mathcal{L}(Q)$ *and* $\mathcal{G}(P) \to \mathcal{G}(Q)$*.*

Proof. Let $x \in \mathcal{L}(P)$. Then there is a neigbourhood $N(P)$ of x such that for all $x' \in N(P)$ with $x' \in \mathcal{F}(P)$ we have $df(x') \leq df(x)$. Since τ is a continuous invertible map, $N(Q) = \tau(N(P))$ is a neighbourhood of $y = \tau(x)$ (so $\tau^{-1}(N(Q)) = N(P)$). For all $y' \in \mathcal{F}(Q)$, by Lemma 3.13 and because all problem variable are continuous, $\tau^{-1}(y') \in \mathcal{F}(P)$. Hence for all $y' \in N(Q) \cap \mathcal{F}(Q)$, $x' = \tau^{-1}(y') \in N(P) \cap \mathcal{F}(P)$. Thus, $d'f'(y') = df'(\tau(x')) = d(f' \circ \tau)(x') = df(x') \leq df(x) = d(f \circ \tau^{-1})(y) = d'f'(y)$. Thus for all $x \in \mathcal{L}(P)$, $\tau(x) \in \mathcal{L}(Q)$. The same argument applied to τ^{-1} shows that for all $y \in \mathcal{L}(Q)$, $\tau^{-1}(y) \in \mathcal{L}(P)$; so τ restricted to $\mathcal{L}(P)$ is a bijection. As concerns global optima, let $x^* \in \mathcal{G}(P)$ and $y^* = \tau(x^*)$; then for all $y \in \mathcal{F}(Q)$ with $y = \tau(x)$, we have $d'f'(y) = d'f'(\tau(x)) = d(f \circ \tau)(x) = df(x) \leq df(x^*) = d'(f \circ \tau^{-1})(y^*) = d'f'(y^*)$, which shows that $y^* \in \mathcal{G}(Q)$. The same argument applied to τ^{-1} shows that τ restricted to $\mathcal{G}(P)$ is a bijection. □

Theorem 3.15. *If $P \approx_\tau Q$ with $\mathcal{V}(P) = x, \mathcal{V}(Q) = y$, $|P| = n, |Q| = m$, $|\mathcal{O}(P)| = |\mathcal{O}(Q)| = 1$ such that (f, d) is the objective function of P and (f', d') is that of Q, $d = d'$, $\mathcal{T}(x) = 0, \mathcal{T}(y) = 0$, then $P < Q$ and $Q < P$.*

Proof. The fact that $P < Q$ follows from Prop. 3.14. The reverse follows by considering τ^{-1}. □

Proposition 3.16. *Let P, Q be two problems with $\mathcal{V}(P) = x, \mathcal{V}(Q) = y$, $|P| = n, |Q| = m$, $|\mathcal{O}(P)| = |\mathcal{O}(Q)| = 1$ such that (f, d) is the objective function of P and (f', d') is that of Q, $d = d'$, $\mathcal{L}(P)$ and $\mathcal{L}(Q)$ both consist of isolated points in the respective Euclidean topologies, and assume $P \prec Q$ and $Q \prec P$. Then there is a continuous invertible map $\tau : \mathcal{F}(P) \to \mathcal{F}(Q)$.*

Proof. Since $P \prec Q$ there is a surjective function $\varphi : \mathcal{L}(Q) \to \mathcal{L}(P)$, which implies $|\mathcal{L}(Q)| \geq |\mathcal{L}(P)|$. Likewise, since $Q \prec P$ there is a surjective function $\psi : \mathcal{L}(P) \to \mathcal{L}(Q)$, which implies $|\mathcal{L}(P)| \geq |\mathcal{L}(Q)|$. This yields $|\mathcal{L}(P)| = |\mathcal{L}(Q)|$, which means that there is a bijection $\tau : \mathcal{L}(P) \to \mathcal{L}(Q)$. Because $\mathcal{L}(P) \subseteq \mathbb{R}^n$ and $\mathcal{L}(Q) \subseteq \mathbb{R}^m$ only contain isolated points, there is a way to extend τ to $\mathbb{R}^n$ so that it is continuous and invertible on the x variable domains, and so that τ^{-1} enjoys the same properties. □

In summary, continuous reformulations of continuous problems are symmetric reformulations, whereas symmetric reformulations may not necessarily be continuous reformulations, unless the set of local optima consists of isolated points. We also remark that continuous reformulations applied to discrete problems may fail to be opt-reformulations. This happens because integrality constraints do not transform with the map τ along with the rest of the problem constraints.

3.6. Narrowings

Narrowings are auxiliary problems that preserve at least one global optimum.

Definition 3.17. *Q is a narrowing of P if there is a function $\varphi : \mathcal{F}(Q) \to \mathcal{F}(P)$ such that (a) $\varphi(y) \in \mathcal{G}(P)$ for all $y \in \mathcal{G}(Q)$.*

Narrowings come in specially useful in presence of problems exhibiting many symmetries: it may then be the huge amount of global optima that is preventing a search from being successful. An example of narrowing is given by the local cuts obtained from the symmetry group of the problem, presented in [61] (see Sect. 4.3).

The fact that all opt-reformulations are a special case of narrowings follows directly from the definition. By a similar argument to Lemma 3.11, it is easy to show that narrowings can be chained to obtain more complex narrowings. Likewise, the chaining of an opt-reformulation and a narrowing is a narrowing.

3.7. Relaxations

Loosely speaking, a relaxation of a problem P is an auxiliary problem of P with fewer constraints. Relaxations are useful because they often yield problems

which are simpler to solve yet they provide a bound on the objective function value at the optimum. Such bounds are mainly used in Branch-and-Bound type algorithms, which are the most common exact or ε-approximate (for a given $\varepsilon > 0$) solution algorithms for MILPs, nonconvex NLPs and MINLPs. A further use of bounds provided by mathematical programming formulations is to evaluate the performance of heuristic algorithms without an approximation guarantee [19]. Bounds are sometimes also used to guide heuristics [69].

Definition 3.18. *Q is a relaxation of P if $\mathcal{F}(P) \subsetneq \mathcal{F}(Q)$.*

The fundamental theorem of relaxations states that relaxations provide bounds to the objective function.

Theorem 3.19. *Let Q be a relaxation of P and let (f, d) be an objective function of P, $x \in \mathcal{G}(P)$ and $y \in \mathcal{G}(Q)$. Then $df(y) \geq df(x)$.*

Proof. Since $\mathcal{F}(Q) \supseteq \mathcal{F}(P)$, for all $x \in \mathcal{G}(P)$, $x \in \mathcal{F}(Q)$, which implies the result. □

Defn. 3.18 is not used very often in practice because it does not say anything on how to construct Q. The following elementary relaxations are more useful.

Definition 3.20. *Q is a:*

- *constraint relaxation of P if $\mathcal{C}(Q) \subsetneq \mathcal{C}(Q)$;*
- *bound relaxation of P if $\mathcal{B}(Q) \subsetneq \mathcal{B}(Q)$;*
- *a continuous relaxation of P if $\exists v \in \mathcal{V}(P)$ $(\mathcal{T}(v) > 0)$ and $\mathcal{T}(Q) = 0$.*

It is easy to show that opt-reformulations and narrowings are special types of relaxations, that relaxations can be chained to obtain other relaxations, and that chains of relaxations with opt-reformulations and narrowings are themselves relaxations.

3.8. Approximations

Definition 3.21. *Q is an approximation of P if there is a countable sequence of problems Q_k (for $k \in \mathbb{N}$), a positive integer k' and an auxiliary problem Q^* of P such that: (a) $Q = Q_{k'}$; (b) for all expression trees $f^* \in \mathcal{O}(Q^*)$ there is a sequence of expression trees $f_k \in \mathcal{O}(Q_k)$ that represent functions converging uniformly to the function represented by f^* (c) for all $c^* = (e^*, s^*, b^*) \in \mathcal{C}(Q^*)$ there is a sequence of constraints $c_k = (e_k, s_k, b_k) \in \mathcal{C}(Q_k)$ such that: (i) the functions represented by e_k converge uniformly to the function represented by e^*; (ii) $s_k = s^*$ for all k; (iii) b_k converges to b.*

Since approximations can be defined for all types of auxiliary problems, we can have approximations to opt-reformulations, narrowings, relaxations and approximations themselves. Approximations are very useful to reformulate MINLPs into MILPs. In general, approximations have no guarantee of optimality, i.e. solving an approximation may give results that are arbitrarily far from the optimum. In practice, however, approximations manage to provide solutions of good quality.

Opt-reformulations, narrowings and relaxations are special types of approximations, since they are all auxiliary problems and one can take the trivial sequence $Q_k = Q^*$ for all k. Chaining approximations and other auxiliary problems yields an approximation.

4. Reformulation examples

In this section we provide some examples for each type of reformulation (opt-reformulation, narrowing, relaxation, approximation) proposed above. For a more complete reformulation library, see [51].

4.1. Hansen's Fixing Criterion as an opt-reformulation

This method applies to unconstrained quadratic 0-1 problems of the form

$$\min_{x \in \{0,1\}^n} x^\top Q x$$

where Q is an $n \times n$ matrix [36], and relies on fixing some of the variables to values guaranteed to provide a global optimum.

Let P be a problem with $\mathcal{P} = \{n \in \mathbb{N}, \{q_{ij} \in \mathbb{R} \mid 1 \leq i, j \leq n\}\}$, $\mathcal{V} = \{x_i \mid 1 \leq i \leq n\}$, $\mathcal{E} = \{f = \sum_{i,j \leq n} q_{ij} x_i x_j\}$, $\mathcal{O} = \{(f, -1)\}$, $\mathcal{C} = \emptyset$, $\mathcal{B} = [0,1]^n$, $\mathcal{T} = \mathbf{2}$. This can be restricted as follows:

- initialize two sequences $V = \emptyset, A = \emptyset$;
- for all $i \leq n$:
  (1) if $q_{ii} + \sum_{j<i} \min(0, q_{ij}) + \sum_{j>i} \min(0, q_{ij}) > 0$ then append x_i to V and 0 to A;
  (2) (else) if $q_{ii} + \sum_{j<i} \max(0, q_{ij}) + \sum_{j>i} \max(0, q_{ij}) < 0$ then append x_i to V and 1 to A;
- apply Restrict(P, V, A).

This opt-reformulation is denoted by HansenFix(P).

Essentially, any time a binary variable consistently decreases the objective function value when fixed, independently of the values of other variables, it is fixed.

4.2. The reduced RLT constraints opt-reformulation

This reformulation concerns a problem P with quadratic terms and linear equality constraints. More precisely, we require P to exhibit the following properties:

- there is a subset $x \subseteq \mathcal{V}$ with $|x| = n$ and a set $E = \{(i,j) \mid 1 \leq i \leq j \leq n\}$ in $\mathcal{P}$ such that the terms $x_i x_j$ appear as sub-expressions in the expressions $\mathcal{E}$ for all $(i,j) \in E$;
- there is a number $m \leq n$, an $m \times n$ matrix $A = (a_{ij})$ and an m-vector b in $\mathcal{P}$ such that $(\sum_{j \leq n} a_{ij} x_j, 0, b_i) \in \mathcal{C}$ for all $i \leq m$.

Let $F = \{(i,j) \mid (i,j) \in E \vee \exists k \leq m (a_{kj} \neq 0\}$. Under these conditions, P can be reformulated as follows:

- for all $(i,j) \in F$ add continuous variables w_{ij} with $\mathcal{T}(w_{ij}) = 0$ and $\mathcal{B}(w_{ij}) = [-\infty, +\infty]$;
- for all $(i,j) \in E$ replace sub-expression $x_i x_j$ with w_{ij} in the expressions $\mathcal{E}$;
- for all $i \leq n, k \leq m$ add the constraints $(\sum_{j \leq n} a_{kj} w_{ij} - b_k x_i, 0, 0)$ to $\mathcal{C}$: we call this linear system the *Reduced RLT Constraint System* (RRCS) and $(\sum_{j \leq n} a_{kj} w_{ij}, 0, 0)$ the *companion* system;
- let $B = \{(i,j) \in F \mid w_{ij}$ is basic in the companion$\}$;
- let $N = \{(i,j) \in F \mid w_{ij}$ is non-basic in the companion$\}$;
- add the constraints $(w_{ij} - x_i x_j, 0, 0)$ for all $(i,j) \in N$.

This opt-reformulation is denoted by $\mathrm{RRLT}(P)$, and its validity was shown in [48]. It is important because it effectively reduces the number of quadratic terms in the problem (only those corresponding to the set N are added). This reformulation can be extended to work with sparse sets E [56], namely sets E whose cardinality is small with respect to $\frac{1}{2}n(n+1)$.

Essentially, the constraints $w_{ij} = x_i x_j$ for $(i,j) \in B$ are replaced by the RRCS $\forall i \leq n\ (Aw_i = x_i)$, where $w_i = (w_{i1}, \ldots, w_{in})$.

4.3. The symmetry group narrowing

This section extends the material in [61, 62] to 0-1 MINLPs. Consider a formulation P in the form (1) where $X = \{0,1\}^n$ and a constraint set $\mathcal{C}(P)$ which we suppose in the form $g(x) \geq b \in \mathbb{R}^m$. Consider also the symmetric groups S^n and S^m of permutations acting on sets of n and respectively m objects. For $\pi \in S^n$, we denote by $x\pi$ the vector obtained by permuting the elements of x, and by $f(x)\pi$ the function $f(x\pi)$ where the variables x were permuted according to π. For $\sigma \in S^m$, we denote by $\sigma g(x)$ the vector-valued function g where the elements are permuted according to σ. The *symmetry group* of P is the group:

$$\bar{G} = \{\pi \in S^n | \exists \sigma \in S^m (\forall f \in \mathcal{O}(P) f(x)\pi = f(x) \wedge \sigma b = b \wedge \sigma g(x)\pi = g(x))\}. \quad (5)$$

We remark that the equalities $f(x)\pi = f(x)$ and $\sigma g(x)\pi = g(x)$ refer to a recursive comparison procedure applied to the expression tree graphs of f, g, and this is why the x variable is unquantified in (5). It is easy to show that $\bar{G}$ is indeed a group.

Assume f, g are linear forms, $\mathcal{T}(x_i) = 2$ for all $i \leq n$ (i.e. x is a vector of binary variables) and $|\mathcal{O}(P)| = 1$; then $f(x) = cx$ for some parameter vector $c \in \mathbb{R}^n$ and $g(x) \geq b$ can be written as $Ax \geq b$. We recover Margot's original definition:

$$G = \{\pi \in S^n | \exists \sigma \in S^m (c\pi = c \wedge \sigma b = b \wedge \sigma A\pi = A)\}.$$

Assume P has a large symmetry group G. When solving P by means of a BB-type algorithm, few nodes can ever be fathomed because of the large number of symmetric globally optimal solutions. The symmetry group G is used in [61, 62] to derive techniques that help the BB algorithm avoid taking symmetric optima into consideration. Let $I^n = \{1, \ldots, n\}$; for a node a of the BB tree, let $F_k^a = \{i \leq$

$n \mid x_i$ fixed at $k\}$ for $k \in \{0,1\}$. For two nodes a, b of the BB tree, a is isomorphic to b if there is $\pi \in G$ such that $F_k^a\pi = F_k^b$ for $k \in \{0,1\}$. For all $S \subseteq I^n$ let $G_S = \{\pi \in G \mid S\pi = S\}$ be the *stabilizer* and $GS = \{T \subseteq I^n \mid \exists \pi \in G\ (S\pi = T)\}$ be the *orbit* of S in G. Margot suggests a BB branching strategy based on grouping BB nodes by isomorphism equivalence classes only only consider one representative per class (this is implemented by considering $S \subseteq I^n$ a representative if S is the lexicographically smallest element of GS). Using this branching strategy, linear inequalities are derived (locally to a given BB node) that cut away some of the left-over symmetric optima. Let a be a node of the BB tree and $H^a = I^n \setminus F_0^a$. For all $J \subseteq H^a$ having representative J^* in $G_{F_1^a}J$ and lexicographically smaller than F_1^a, if some descendant BB node b of a is such that F_1^b contains J it can be pruned immediately. This can be obtained by adding a cut:

$$\sum_{j \in J} x_j \leq |J| - 1$$

local to the subproblem at node a. The above cuts provide a narrowing denoted by SymmCutLin(P, a) valid at a and all its descendants. As the above ideas are not based on the fact that f, g are linear but only on the symmetries of variables and constraints with the respect to which the mathematical program is invariant, they extend naturally to the group $\bar{G}$ of any MINLP involving binary variables.

4.4. Reformulation-Linearization Technique based relaxation

The Reformulation-Linearization Technique (RLT) is a relaxation method for mathematical programming problems with quadratic terms. The RLT linearizes all quadratic terms in the problem and generates valid linear equation and inequality constraints by considering multiplications of bound factors (terms like $x_i - x_i^L$ and $x_i^U - x_i$) and constraint factors (the left hand side of a constraint such as $\sum_{j=1}^n a_j x_j - b \geq 0$ or $\sum_{j=1}^n a_j x_j - b = 0$). Since bound and constraint factors are always non-negative, so are their products: this way one can generate sets of valid problem constraints. In a sequence of papers published from the 1980s onwards (see e.g. [72,74–76,78–80]), RLT-based relaxations were derived for many different classes of problems, including IPs, NLPs, MINLPs in general formulation, and several real-life applications. It was shown that the RLT can be used in a lift-and-project fashion to generate the convex envelope of binary and general discrete problems [2,77].

4.4.1. *Basic RLT*

The RLT consists of two symbolic manipulation steps: reformulation and linearization. The reformulation step is a reformulation in the sense of Defn. 3.9. Given a problem P, the reformulation step produces a reformulation Q' where:

- $\mathcal{P}(Q') = \mathcal{P}(P)$;
- $\mathcal{V}(Q') = \mathcal{V}(P)$;
- $\mathcal{E}(Q') \supseteq \mathcal{E}(P)$;

- $\mathcal{C}(Q') \supseteq \mathcal{C}(P)$;
- $\mathcal{O}(Q') = \mathcal{O}(P)$;
- $\mathcal{B}(Q') = \mathcal{B}(P)$;
- $\mathcal{T}(Q') = \mathcal{T}(P)$;
- $\forall x, y \in \mathcal{V}(P)$, add the following constraints to $\mathcal{C}(Q')$:

$$\begin{aligned}(x - L_x)(y - L_y) &\geq 0 &(6)\\ (x - L_x)(U_y - y) &\geq 0 &(7)\\ (U_x - x)(y - L_y) &\geq 0 &(8)\\ (U_x - x)(L_y - y) &\geq 0; &(9)\end{aligned}$$

- $\forall x \in \mathcal{V}(P), c = (e_c, s_c, b_c) \in \mathcal{C}(P)$ such that e_c is an affine form, $s_c = 1$ and $b_c = 0$ (we remark that all linear inequality constraints can be easily reformulated to this form), add the following constraints to $\mathcal{C}(Q')$:

$$\begin{aligned}e_c(x - L_x) &\geq 0 &(10)\\ e_c(U_x - x) &\geq 0; &(11)\end{aligned}$$

- $\forall x \in \mathcal{V}(P), c = (e_c, s_c, b_c) \in \mathcal{C}(P)$ such that e_c is an affine form, $s_c = 0$ and $b_c = 0$ (we remark that all linear equality constraints can be trivially reformulated to this form), add the following constraint to $\mathcal{C}(Q')$:

$$e_c x = 0. \quad (12)$$

Having obtained Q', we proceed to linearize all the quadratic products engendered by (6)-(12). We derive the auxiliary problem Q from Q' by reformulating Q' in Smith's standard form (see Sect. 3.2.4) and then performing a constraint relaxation with respect to all defining constraints; we denote the resulting relaxation by RLT(P). Smith's standard form is a reformulation of the lifting type, and the obtained constraint relaxation Q is a MILP whose optimal objective function value $\bar{f}$ is a bound to the optimal objective function value f^* of the original problem P. The bound obtained in this way is shown to dominate, or be equivalent to, several other bounds in the literature [2].

We remark in passing that (6)-(9), when linearized by replacing the bilinear term xy with an added variable w, are also known in the literature as McCormick relaxation, as they were first proposed as a convex relaxation of the nonconvex constraint $w = xy$ [63], shown to be the convex envelope [5], and widely used in spatiala Branch-and-Bound (sBB) algorithms for global optimization [3, 4, 49, 83, 89]. RLT constraints of type (12) have been the object of further research showing their reformulating power [45, 47, 48, 50, 56] (also see Sect. 4.2).

4.4.2. *RLT Hierarchy*

The basic RLT method can be extended to provide a hierarchy of relaxations, by noticing that we can form valid RLT constraints by multiplying sets of bound and

constraint factors of cardinality higher than 2, and then projecting the obtained constraints back to the original variable space. In [2, 77] it is shown that this fact can be used to construct the convex hull of an arbitrary MILP P. For simplicity, we only report the procedure for MILP in standard canonical form (see Sect. 3.2.2) where all discrete variables are binary, i.e. $\mathcal{T}(v) = 2$ for all $v \in \mathcal{V}(P)$. Let $|\mathcal{V}(P)| = n$. For all integer $d \leq n$, let P_d be the relaxation of P obtained as follows:

- for all linear constraint $c = (e_c, 1, 0) \in \mathcal{C}(P)$, subset $V \subseteq \mathcal{V}(P)$ and finite binary sequence B with $|V| = |B| = d$ such that B_x is the x-th term of the sequence for $x \in V$, add the valid constraint:

$$e_c \left(\prod_{\substack{x \in V \\ B_x = 0}} x \right) \left(\prod_{\substack{x \in V \\ B_x = 1}} (1 - x) \right) \geq 0; \tag{13}$$

  we remark that (13) is a multivariate polynomial inequality;
- for all monomials of the form

$$a \prod_{x \in J \subseteq \mathcal{V}(P)} x$$

  with $a \in \mathbb{R}$ in a constraint (13), replace $\prod_{x \in J} x$ with an added variable w_J (this is equivalent to relaxing a defining constraint $w_J = \prod_{x \in J}$ in the Smith's standard form restricted to (13).

Now consider the projection X_d of P_d in the $\mathcal{V}(P)$ variable space. It can be shown that

$$\text{conv}(\mathcal{F}(P)) \subseteq \mathcal{F}(X_n) \subseteq \mathcal{F}(X_{n-1}) \ldots \subseteq \mathcal{F}(X_1) \subseteq \mathcal{F}(P).$$

We recall that for a set $Y \subseteq \mathbb{R}^n$, $\text{conv}(Y)$ is defined as the smallest convex subset of $\mathbb{R}^n$ containing Y.

A natural practical application of the RLT hierarchy is to generate relaxations for polynomial programming problems [75], where the various multivariate monomials generated by the RLT hierarchy might already be present in the problem formulation. We denote the relaxation P_d by $\text{RLT}(P, d)$.

4.5. Signomial programming based relaxation

A signomial programming problem is an optimization problem where every objective function is a signomial function and every constraint is of the form $c = (g, s, 0)$ where g is a signomial function of the problem variables, and $s \neq 0$

(so signomial equality constraints must be reformulated to pairs of inequality constraints). A *signomial* is a term of the form:

$$a \prod_{k=1}^{K} x_k^{r_k}, \tag{14}$$

where $a, r_k \in \mathbb{R}$ for all $k \in K$, and the r_k exponents are assumed ordered so that $r_k > 0$ for all $k \leq m$ and $r_k < 0$ for $m \leq k \leq K$. Because the exponents of the variables are real constants, this is a generalization of a multivariate monomial term. A *signomial function* is a sum of signomial terms. In [15], a set of transformations of the form $x_k = f_k(z_k)$ are proposed, where x_k is a problem variable, z_k is a variable in the reformulated problem and f_k is suitable function that can be either exponential or power. This yields an opt-reformulation where all the inequality constraints are convex, and the variables z and the associated (inverse) defining constraints $x_k = f_k(z_k)$ are added to the reformulation for all $k \in K$ (over each signomial term of each signomial constraint).

We distinguish the following cases:

- If $a > 0$, the transformation functions f_k are exponential univariate, i.e. $x_k = e^{z_k}$. This reformulates (14) as follows:

$$\left.\begin{array}{c} a \frac{e^{\sum_{k \leq m} r_k z_k}}{\prod_{k=m+1}^{K} x_k^{|r_k|}} \\ \forall k \leq K \quad x_k = e^{z_k}. \end{array}\right\}$$

- If $a < 0$, the transformation functions are power univariate, i.e. $x_k = z_k^{\frac{1}{R}}$ for $k \leq m$ and $x_k = z_k^{-\frac{1}{R}}$ for $k > m$, where $R = \sum_{k \leq K} |r_k|$. This is also called a *potential transformation*. This reformulates (14) as follows:

$$\left.\begin{array}{c} a \prod_{k \leq K} z_k^{\frac{|r_k|}{R}} \\ \forall k \leq m \quad x_k = z_k^{\frac{1}{R}} \\ \forall k > m \quad x_k = z_k^{-\frac{1}{R}} \\ R = \sum_{k \leq K} |r_k|. \end{array}\right\}$$

This opt-reformulation isolates all nonconvexities in the inverse defining constraints. These are transformed as follows:

$$\begin{array}{rcl} \forall k \leq K \; x_k = e^{z_k} & \Rightarrow & \forall k \leq K \; z_k = \log x_k \\ \forall k \leq m \quad z_k = x_k^R & & \\ \forall k > m \quad z_k = x_k^{-R}, & & \end{array}$$

and then relaxed using a piecewise linear approximation as per Fig. 4. This requires the introduction of binary variables (one per turning point).

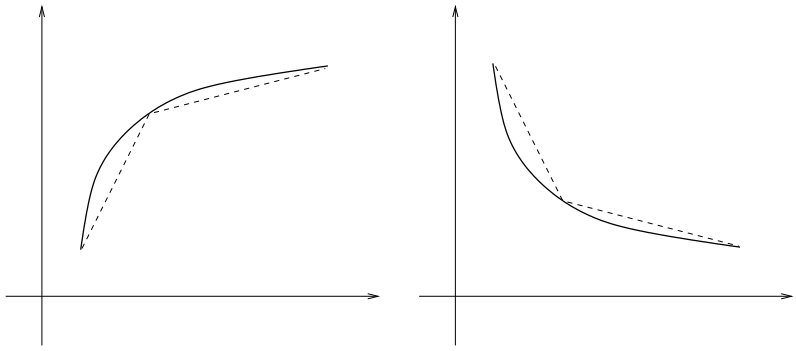

FIGURE 4. Piecewise linear underestimating approximations for concave (left) and convex (right) univariate functions.

The resulting signomial relaxation, denoted SIGRELAX(P), is a convex MINLP; it can be further relaxed to a MILP by outer approximation of the convex terms, or to a convex NLP by continuous relaxation of the discrete variables.

4.6. APPROXIMATION OF BILINEAR PRODUCTS

Consider a problem P with two continuous variables $x, y \in \mathcal{V}$ such that $\mathcal{T}(x) = \mathcal{T}(y) = 0$, $\mathcal{B}(x) = [x^L, x^U]$ and $\mathcal{B}(y) = [y^L, y^U]$ with $x^U - x^L \leq y^U - y^L$. Assume there is a bilinear product xy appearing in some expression tree (objective and/or constraint). For any positive integer k, the following is an approximation of the identity opt-reformulation of P:

- add a continuous variable w to $\mathcal{V}$ such that $\mathcal{T}(w) = 0$ and $\mathcal{B}(w) = [w^L, w^U]$ where $w^L = \min(x^L y^L, x^L y^U, x^U y^L, x^U y^U)$ and $w^U = \max(x^L y^L, x^L y^U, x^U y^L, x^U y^U)]$;
- for all $1 \leq i \leq k$ add binary variables z_i to $\mathcal{V}$ with $\mathcal{T}(z_i) = 2$;
- for all $0 \leq i \leq k$ add parameters q_i to $\mathcal{P}$ with distinct values in $[x^L, x^U]$ such that $q_0 = x^L, q_k = x^U$ and $q_i < q_j$ for all $i < j$;
- replace all occurrences of the product xy by the variable w;
- add the assignment constraint $(\sum_{i=1}^{k} z_i, 0, 1)$ to $\mathcal{C}$;
- for all $i \in \{1, \ldots, k\}$ add the (linear) constraints $(x_i - \sum_{j=1}^{k} q_j z_j, -1, 0)$ and $(x_i - \sum_{j=1}^{k} q_{j-1} z_j, 1, 0)$ to $\mathcal{C}$;
- for all $i \in \{1, \ldots, k\}$ add the (linear) constraints $(w - \frac{q_i + q_{i-1}}{2} y - (w^U - w^L) z_i, -1, 0)$ and $(w - \frac{q_i + q_{i-1}}{2} y + (w^U - w^L) z_i, 1, 0)$ to $\mathcal{C}$.

This approximation is denoted by BILINAPPROX(P, x, y, q, k). Essentially, we discretize the range of x (the variable in the product having the smallest range) by means of $k+1$ points q; for all values of x ranging in $[q_{i-1}, q_i]$ we define w as being

the straight line $\frac{q_i+q_{i-1}}{2}y$ by means of the constraints:

$$\left.\begin{array}{c} \sum_{j=1}^{k} q_{j-1}z_j \le x_i \le \sum_{j=1}^{k} q_j z_j \\ \frac{q_i+q_{i-1}}{2}y - (w^U - w^L)(1-z_i) \le w \le \frac{q_i+q_{i-1}}{2}y + (w^U - w^L)(1-z_i), \end{array}\right\} \quad (15)$$

for all $i \in \{1, \ldots, k\}$. The fact that BilinApprox is an approximation follows because by (15) we have that for $k \to \infty$, if $x = q \in [x^L, x^U]$ then $w \to qy$.

Although different approximations of the term xy are possible, the one presented in this section employs a reasonably small number of variables and is not likely to restrict the feasible region of the problem. Geometrically, this approximation is depicted in Fig. 5.

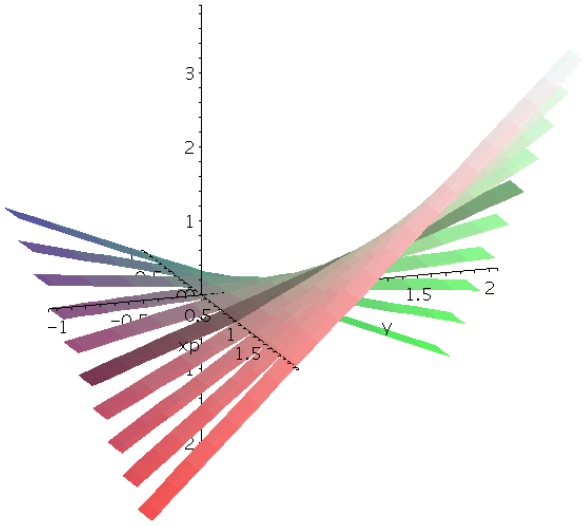

Figure 5. Linearizing approximation of a bilinear term.

5. Conclusion

This paper describes a theoretical framework for the analysis and classification of reformulations for mathematical programs that can be carried out automatically by means of a symbolic/numerical algorithm acting on the formulation. This fundamental study will serve as the basis for a software that can carry out reformulations of mathematical programs automatically.

Acknowledgments

The financial support of ANR JCJC grant "ARS" and of EU NEST grant "Morphex" is gratefully acknowledged.

References

[1] W.P. Adams, R.J. Forrester, and F.W. Glover. Comparisons and enhancement strategies for linearizing mixed 0-1 quadratic programs. *Discrete Optimization*, 1:99–120, 2004.

[2] W.P. Adams and H.D. Sherali. A hierarchy of relaxations leading to the convex hull representation for general discrete optimization problems. *Annals of Operations Research*, 140:21–47, 2005.

[3] C.S. Adjiman, S. Dallwig, C.A. Floudas, and A. Neumaier. A global optimization method, αBB, for general twice-differentiable constrained NLPs: I. Theoretical advances. *Computers & Chemical Engineering*, 22(9):1137–1158, 1998.

[4] C. S. Adjiman, I. P. Androulakis, and C. A. Floudas. A global optimization method, αBB, for general twice-differentiable constrained NLPs: II. Implementation and computational results. *Computers & Chemical Engineering*, 22(9):1159–1179, 1998.

[5] F.A. Al-Khayyal and J.E. Falk. Jointly constrained biconvex programming. *Mathematics of Operations Research*, 8(2):273–286, 1983.

[6] F. Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM Journal on Optimization*, 5(1):13–51, 1995.

[7] H.M.T. Ben Amor, J. Desrosiers, and A. Frangioni. Stabilization in column generation. *Discrete Applied Mathematics*, accepted for publication.

[8] K.M. Anstreicher. Recent advances in the solution of quadratic assignment problems. *Mathematical Programming B*, 97:27–42, 2003.

[9] C. Audet, J. Brimberg, P. Hansen, S. Le Digabel, and N. Mladenović. Pooling problem: Alternate formulations and solution methods. *Management Science*, 50(6):761–776, 2004.

[10] C. Audet, P. Hansen, B. Jaumard, and G. Savard. Links between linear bilevel and mixed 0-1 programming problems. *Journal of Optimization Theory and Applications*, 93(2):273–300, 1997.

[11] W. Ben-Ameur and H. Kerivin. Routing of uncertain demands. *Optimization and Engineering*, 3:283–313, 2005.

[12] D. Bertsimas and M. Sym. The price of robustness. *Operations Research*, 52(1):35–53, 2004.

[13] A. Billionnet and S. Elloumi. Using a mixed-integer quadratic programming solver for the unconstrained quadratic 0-1 problem. *Mathematical Programming*, 109:55–68, 2007.

[14] A. Billionnet, S. Elloumi, and M.-C. Plateau. Improving the performance of standard solvers via a tighter convex reformulation of constrained quadratic 0-1 programs: the QCR method. *Discrete Applied Mathematics*, to appear.

[15] K.-M. Björk, P.O. Lindberg, and T. Westerlund. Some convexifications in global optimization of problems containing signomial terms. *Computers & Chemical Engineering*, 27:669–679, 2003.

[16] J. Bjorkqvist and T. Westerlund. Automated reformulation of disjunctive constraints in MINLP optimization. *Computers & Chemical Engineering*, 23:S11–S14, June 1999.

[17] A. Brook, D. Kendrick, and A. Meeraus. Gams, a user's guide. *ACM SIGNUM Newsletter*, 23(3-4):10–11, 1988.

[18] G.B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, Princeton, NJ, 1963.

[19] T. Davidović, L. Liberti, N. Maculan, and N. Mladenović. Towards the optimal solution of the multiprocessor scheduling problem with communication delays. In *MISTA Proceedings*, 2007.

[20] J.E. Falk and J. Liu. On bilevel programming, part i: general nonlinear cases. *Mathematical Programming*, 70:47–72, 1995.
[21] R. Fletcher and S. Leyffer. User manual for filter. Technical report, University of Dundee, UK, March 1999.
[22] R. Fortet. Applications de l'algèbre de boole en recherche opérationelle. *Revue Française de Recherche Opérationelle*, 4:17–26, 1960.
[23] R. Fourer. Personal communication. 2004.
[24] R. Fourer and D. Gay. *The AMPL Book*. Duxbury Press, Pacific Grove, 2002.
[25] R. Fourer, J. Ma, K. Martin, and W. Sheng. Optimization services 1.0 user manual. Technical report, COIN-OR, November 2007.
[26] A. Frangioni. On a new class of bilevel programming problems and its use for reformulating mixed-integer problems. *European Journal of Operations Research*, 82(3):615–646, May 1995.
[27] S. Galli. Parsing ampl internal format for linear and non-linear expressions, 2004. Didactical project, DEI, Politecnico di Milano, Italy.
[28] L. Di Giacomo. *Mathematical programming methods in dynamical nonlinear stochastic Supply Chain management*. PhD thesis, DSPSA, Università di Roma "La Sapienza", 2007.
[29] P.E. Gill. *User's Guide for SNOPT 5.3*. Systems Optimization Laboratory, Department of EESOR, Stanford University, California, February 1999.
[30] M. Grant, S. Boyd, and Y. Ye. Disciplined convex programming. In Liberti and Maculan [54], pages 155–210.
[31] C. Guéret, C. Prins, and M. Sevaux. *Applications of optimization with Xpress-MP*. Dash Optimization, Bilsworth, 2000.
[32] S. Gueye and P. Michelon. "miniaturized" linearizations for quadratic 0/1 problems. *Annals of Operations Research*, 140:235–261, 2005.
[33] S. Gueye and Ph. Michelon. A linearization framework for unconstrained quadratic (0-1) problems. *Discrete Applied Mathematics*, to appear.
[34] K. Hägglöf, P.O. Lindberg, and L. Svensson. Computing global minima to polynomial optimization problems using gröbner bases. *Journal of Global Optimization*, 7(2):115:125, 1995.
[35] P.L. Hammer and S. Rudeanu. *Boolean Methods in Operations Research and Related Areas*. Springer, Berlin, 1968.
[36] P. Hansen. Method of non-linear 0-1 programming. *Annals of Discrete Mathematics*, 5:53–70, 1979.
[37] P. Hansen and C. Meyer. Improved compact linearizations for the unconstrained quadratic 0-1 minimization problem. *Discrete Applied Mathematics*, to appear.
[38] R. Horst. On the convexification of nonlinear programming problems: an applications-oriented approach. *European Journal of Operations Research*, 15:382–392, 1984.
[39] R. Horst and Hoang Tuy. *Global Optimization: Deterministic Approaches*. Springer-Verlag, Berlin, third edition, 1996.
[40] K.-L. Hsiung, S.-J. Kim, and S. Boyd. Tractable approximate robust geometric programming. *Optimization and Engineering*, to appear.
[41] ILOG. *ILOG CPLEX 10.0 User's Manual*. ILOG S.A., Gentilly, France, 2005.
[42] J. Judice and G. Mitra. Reformulation of mathematical programming problems as linear complementarity problems and investigation of their solution methods. *Journal of Optimization Theory and Applications*, 57(1):123–149, 1988.
[43] M. Kojima, N. Megiddo, and Y. Ye. An interior point potential reduction algorithm for the linear complementarity problem. *Mathematical Programming*, 54:267–279, 1992.
[44] L. Liberti. Comparison of convex relaxations for monomials of odd degree. In I. Tseveendorj, P.M. Pardalos, and R. Enkhbat, editors, *Optimization and Optimal Control*. World Scientific, 2003.
[45] L. Liberti. Reduction constraints for the global optimization of NLPs. *International Transactions in Operations Research*, 11(1):34–41, 2004.
[46] L. Liberti. *Reformulation and Convex Relaxation Techniques for Global Optimization*. PhD thesis, Imperial College London, UK, March 2004.

[47] L. Liberti. Reformulation and convex relaxation techniques for global optimization. *4OR*, 2:255–258, 2004.
[48] L. Liberti. Linearity embedded in nonconvex programs. *Journal of Global Optimization*, 33(2):157–196, 2005.
[49] L. Liberti. Writing global optimization software. In Liberti and Maculan [54], pages 211–262.
[50] L. Liberti. Compact linearization of binary quadratic problems. *4OR*, 5(3):231–245, 2007.
[51] L. Liberti. Reformulation techniques in mathematical programming. Thèse d'Habilitation à Diriger des Recherches, Université de Paris-Dauphine, Nov. 2007.
[52] L. Liberti, E. Amaldi, N. Maculan, and F. Maffioli. Mathematical models and a constructive heuristic for finding minimum fundamental cycle bases. *Yugoslav Journal of Operations Research*, 15(1):15–24, 2005.
[53] L. Liberti, C. Lavor, M.A. Chaer Nascimento, and N. Maculan. Reformulation in mathematical programming: an application to quantum chemistry. *Discrete Applied Mathematics*, accepted for publication.
[54] L. Liberti and N. Maculan, editors. *Global Optimization: from Theory to Implementation*. Springer, Berlin, 2006.
[55] L. Liberti and C.C. Pantelides. Convex envelopes of monomials of odd degree. *Journal of Global Optimization*, 25:157–168, 2003.
[56] L. Liberti and C.C. Pantelides. An exact reformulation algorithm for large nonconvex NLPs involving bilinear terms. *Journal of Global Optimization*, 36:161–189, 2006.
[57] J.A. De Loera, J. Lee, S. Margulies, and S. Onn. Expressing combinatorial optimization problems by systems of polynomial equations and the nullstellensatz. Technical Report RC24276(W0706-020), IBM Corporation, 2007.
[58] R. Lougee-Heimer. The common optimization interface for operations research: Promoting open-source software in the operations research community. *IBM Journal of Research and Development*, 47(1):57–66, 2003.
[59] O.L. Mangasarian. Linear complementarity problems solvable by a single linear program. *Mathematical Programming*, 10:263–270, 1976.
[60] O.L. Mangasarian. The linear complementarity problem as a separable bilinear program. *Journal of Global Optimization*, 6:153–161, 1995.
[61] F. Margot. Pruning by isomorphism in branch-and-cut. *Mathematical Programming*, 94:71–90, 2002.
[62] F. Margot. Exploiting orbits in symmetric ilp. *Mathematical Programming B*, 98:3–21, 2003.
[63] G.P. McCormick. Computability of global solutions to factorable nonconvex programs: Part i — convex underestimating problems. *Mathematical Programming*, 10:146–175, 1976.
[64] C.A. Meyer and C.A. Floudas. Convex hull of trilinear monomials with mixed sign domains. *Journal of Global Optimization*, 29:125–155, 2004.
[65] N. Mladenović, F. Plastria, and D. Urošević. Reformulation descent applied to circle packing problems. *Computers and Operations Research*, 32(9):2419–2434, 2005.
[66] I. Nowak. *Relaxation and Decomposition Methods for Mixed Integer Nonlinear Programming*. Birkhäuser, Basel, 2005.
[67] C.C. Pantelides, L. Liberti, P. Tsiakis, and T. Crombie. Mixed integer linear/nonlinear programming interface specification. *Global Cape-Open Deliverable WP2.3-04*, February 2002.
[68] M.-C. Plateau. *Reformulations quadratiques convexes pour la programmation quadratique en variables 0-1*. PhD thesis, Conservatoire National d'Arts et Métiers, 2006.
[69] J. Puchinger and G.R. Raidl. Relaxation guided variable neighbourhood search. In *Proc. of Mini Euro Conference on Variable Neighbourhood Search, Tenerife, Spain*, 2005.
[70] A. Saxena, V. Goyal, and M. Lejeune. MIP reformulations of the probabilistic set covering problem. Technical Report 2007-02-1579, Optimization Online, 2007.
[71] H. Sherali. Personal communication. June 2007.
[72] H. Sherali and L. Liberti. Reformulation-linearization methods for global optimization. In C. Floudas and P. Pardalos, editors, *Encyclopedia of Optimization*. Springer, New York, to appear.

[73] H. Sherali and C.H. Tuncbilek. A global optimization algorithm for polynomial programming problems using a reformulation-linearization technique. *Journal of Global Optimization*, 2:101–112, 1991.
[74] H. Sherali and C.H. Tuncbilek. New reformulation linearization/convexification relaxations for univariate and multivariate polynomial programming problems. *Operations Research Letters*, 21:1–9, 1997.
[75] H.D. Sherali. Global optimization of nonconvex polynomial programming problems having rational exponents. *Journal of Global Optimization*, 12:267–283, 1998.
[76] H.D. Sherali and W.P. Adams. A tight linearization and an algorithm for 0-1 quadratic programming problems. *Management Science*, 32(10):1274–1290, 1986.
[77] H.D. Sherali and W.P. Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM Journal of Discrete Mathematics*, 3:411–430, 1990.
[78] H.D. Sherali and W.P. Adams. *A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems*. Kluwer Academic Publishers, Dodrecht, 1999.
[79] H.D. Sherali and A. Alameddine. A new reformulation-linearization technique for bilinear programming problems. *Journal of Global Optimization*, 2:379–410, 1992.
[80] H.D. Sherali and H. Wang. Global optimization of nonconvex factorable programming problems. *Mathematical Programming*, 89:459–478, 2001.
[81] E.M.B. Smith. *On the Optimal Design of Continuous Processes*. PhD thesis, Imperial College of Science, Technology and Medicine, University of London, October 1996.
[82] E.M.B. Smith and C.C. Pantelides. Global optimisation of nonconvex MINLPs. *Computers & Chemical Engineering*, 21:S791–S796, 1997.
[83] E.M.B. Smith and C.C. Pantelides. A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex MINLPs. *Computers & Chemical Engineering*, 23:457–478, 1999.
[84] INFORMS Computing Society. The mathematical programming glossary. `http://glossary.computing.society.informs.org/second.php?page=R.html`.
[85] A.S. Strekalovsky. On global optimality conditions for d.c. programming problems. *Technical Paper, Irkutsk State University*, 1997.
[86] A.S. Strekalovsky. Extremal problems with d.c. constraints. *Computational Mathematics and Mathematical Physics*, 41(12):1742–1751, 2001.
[87] F. Tardella. Existence and sum decomposition of vertex polyhedral convex envelopes. Technical report, Facoltà di Economia e Commercio, Università di Roma "La Sapienza", 2007.
[88] M. Tawarmalani and N. Sahinidis. Convex extensions and envelopes of semi-continuous functions. *Mathematical Programming*, 93(2):247–263, 2002.
[89] M. Tawarmalani and N.V. Sahinidis. Global optimization of mixed integer nonlinear programs: A theoretical and computational study. *Mathematical Programming*, 99:563–591, 2004.
[90] M.J. Todd. Semidefinite optimization. *Acta Numerica*, 10:515–560, 2001.
[91] H. Tuy. D.c. optimization: Theory, methods and algorithms. In R. Horst and P.M. Pardalos, editors, *Handbook of Global Optimization*, volume 1, pages 149–216. Kluwer Academic Publishers, Dordrecht, 1995.
[92] T.J. van Roy and L.A. Wolsey. Solving mixed integer programming problems using automatic reformulation. *Operations Research*, 35(1):45–57, 1987.
[93] J.C. Vera, J.F. Pe na, and L.F. Zuluaga. Exploiting equalities in polynomial programming. Technical Report 2006-05-1338, Optimization Online, 2006.
[94] X. Wang and T.S. Change. A multivariate global optimization using linear bounding functions. *Journal of Global Optimization*, 12:383–404, 1998.
[95] T. Westerlund. Some transformation techniques in global optimization. In Liberti and Maculan [54], pages 45–74.
[96] L.A. Wolsey. *Integer Programming*. Wiley, New York, 1998.