Reformulations in Mathematical Programming: Definitions

Leo Liberti [1]

Key words: Opt-reformulation, narrowing, relaxation, approximation.

1 Introduction

The mathematical programming formulation language is a very powerful tool 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, but usually 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 can usually be hooked into language 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; yet there are surprisingly few attempts to formally define what a reformulation in mathematical programming actually is [7,1]. Furthermore, there is a remarkable lack of literature reviews on the topic of reformulations in mathematical programming [3]; and even more importantly, very few solution methods consider reformulation-based algorithmic steps (usually, the reformulation is taken to be a pre-processing step) [6]. Although some automatic relaxation software exists [2], there is no equivalent for general reformulations.

In this paper we propose a data structure for storing and manipulation mathematical programming formulations, and several definitions of different types

Email address: liberti@lix.polytechnique.fr (Leo Liberti).

[1] LIX, École Polytechnique, F-91128 Palaiseau, France

of reformulations, all based on transformations carried out on the proposed data structure. A (non-exhaustive) list of known reformulations based on these definitions can be found in [4].

2 A data structure for mathematical programs

We refer to a mathematical programming problem in the most general form:

$$\min\{f(x) \mid g(x) \lesseqgtr b \wedge x \in X\}, \tag{1}$$

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. We let $\mathbb{P}$ be the set of all mathematical programming formulations and $\mathbb{M}$ be the set of all matrices.

Definition 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 Acyclic Graph (DAG) $e = (V_e, A_e)$ 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) vertices $v \in V_e$ such that $\delta^+(v) = \emptyset$ are called leaf vertices and their set is denoted by $\lambda(e)$; all leaf vertices v 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; for any vertex $v \in V_e$, we let $\tau(v)$ be the symbolic term of v: namely, $v = \chi(v)\tau(v)^{\xi(v)}$.

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 $e_c \in \mathcal{E}, s_c \in \mathcal{S}, b_c \in \mathbb{R}$: $c \equiv \begin{cases} e_c \leq b_c \text{ if } s_c = -1 \\ e_c = b_c \text{ if } s_c = 0 \\ e_c \geq b_c \text{ if } s_c = 1; \end{cases}$*
- *$\mathcal{B} \subseteq \mathbb{R}^{|\mathcal{V}|} \times \mathbb{R}^{|\mathcal{V}|}$ 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 write $\mathcal{T}(z)$ and respectively $\mathcal{B}(z)$ to mean the sequences of types and respectively bound intervals of the sequence of variables in $z \subseteq \mathcal{V}$. We sometimes refer to a formulation by calling it an *optimization problem* or simply a *problem*. Consider 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. 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 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 $y \neq x$. 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 $\mathcal{L}(P) = \mathcal{G}(P) = \mathcal{F}(P)$.

3 Reformulations

The generic term we employ for a problem Q related to a given problem P by some form of transformation carried out on the formulation of P as defined in Defn. 1 is *auxiliary problem*. 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).

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

Definition 2 *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 $P \prec_\varphi Q$. 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 $P \lhd_\varphi Q$. We write $P \prec Q$ (resp. $P \lhd Q$) if there is a φ such that $P \prec_\varphi Q$*

(resp. $P \lhd_\varphi Q$). Q is an opt-reformulation of P (denoted by $P < Q$) if $P \prec Q$ and $P \lhd Q$.

Opt-reformulations can be chained (i.e. applied in sequence) to obtain other opt-reformulations.

Lemma 3 *The relations $\prec, \lhd, <$ are reflexive and transitive.*

Narrowings are auxiliary problems that preserve at least one global optimum. They 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 [5]. All opt-reformulations are a special case of narrowings; narrowings can be chained to obtain more complex narrowings. Chaining an opt-reformulation and a narrowing results in a narrowing.

Definition 4 *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 $y \in \mathcal{G}(Q)$.*

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. The "fundamental theorem" of relaxations states that relaxations provide bounds to the objective function. Opt-reformulations and narrowings are special types of relaxations. Relaxations can be chained to obtain other relaxations; chaining of relaxations with opt-reformulations and narrowings results in other relaxations.

Definition 5 *Q is a relaxation of P if $\mathcal{F}(P) \subsetneq \mathcal{F}(Q)$.*

Approximations are auxiliary problems dependent on a numerical parameter, which approximate as closely as desired other auxiliary problems for some limiting value of the parameter. Since approximations can be defined for all types of auxiliary problems, we can have approximations to opt-reformulations, narrowings, relaxations and approximations themselves. 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. Chaining approximations and other auxiliary problems yields an approximation.

Definition 6 *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.

References

[1] 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.

[2] E.P. Gatzke, J.E. Tolsma, and P.I. Barton. Construction of convex relaxations using automated code generation techniques. *Optimization and Engineering*, 3:305–326, 2002.

[3] L. Liberti. *Reformulation and Convex Relaxation Techniques for Global Optimization*. PhD thesis, Imperial College London, UK, March 2004.

[4] L. Liberti. Reformulation techniques in mathematical programming, November 2007. Thèse d'Habilitation à Diriger des Recherches.

[5] F. Margot. Pruning by isomorphism in branch-and-cut. *Mathematical Programming*, 94:71–90, 2002.

[6] N. Mladenović, F. Plastria, and D. Urošević. Reformulation descent applied to circle packing problems. *Computers and Operations Research*, 32(9):2419–2434, 2005.

[7] H. Sherali. Personal communication. June 2007.